UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

Estudo Comparativo de Técnicas de Virtualização

Autor:

Carlo Fragni

Orientador:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Examinadores:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Miguel Elias Mitre Campista, D.Sc.

DEL
Fevereiro de 2009

Dedico este trabalho a todos que me ajudaram e apoiaram a chegar até aqui, em especial à minha família e à minha namorada Lívia.

Agradecimentos

Desejo agradecer a todos que me possibilitaram este momento.

Em primeiro lugar, aos meus pais, que sempre me deram amor, me passaram valores e me criaram dando sempre o melhor que podiam.

À minha namorada, que sempre conseguiu me arrancar um sorriso, me presenteou com inúmeros momentos felizes e com quem passarei muitos outros.

À minha família, que sempre me deu atenção e me motivou a seguir em frente sempre buscando o melhor.

Aos meus amigos, que sempre estiveram ao meu lado e de quem tenho diversas boas lembranças.

Aos meus professores, que dedicaram suas vidas ao saber e me ensinaram desde as coisas mais simples até as mais complexas, contribuindo para que eu pudesse me tornar um engenheiro.

Ao meu orientador e a todo o grupo do GTA, que rapidamente me acolheram, tornaram possível a confecção deste trabalho e com os quais certamente compartilharei os avanços futuros.

Resumo do Projeto Final apresentado ao DEL/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

Estudo Comparativo de Técnicas de Virtualização

Carlo Fragni

Fevereiro de 2009

Orientador: Otto Carlos Muniz Bandeira Duarte
Departamento: Engenharia Eletrônica e de Computação

O presente trabalho trata do tema de Virtualização. Virtualização é uma área do conhecimento, muitas vezes associada a Sistemas Operacionais, que estuda técnicas para particionar um sistema computacional físico em sistemas computacionais virtuais, os quais oferecem um ambiente extremamente similar a um sistema computacional real.

Atualmente, a virtualização é uma área que recebe investimentos em larga escala e é considerada uma tecnologia extremamente promissora. Segundo *The Green Grid*, grupo formado por grandes empresas com o intuito de melhorar a eficiência das tecnologias de informação do ponto de vista ambiental, a virtualização é um fator chave para obter a sustentabilidade ambiental das infraestruturas de tecnologia da informação. Conforme *Gartner*, empresa mais respeitada para consultoria estratégica em tecnologia da informação, virtualização é uma das dez tecnologias estratégicas para as empresas hoje.

Os usos correntes da virtualização incluem consolidar servidores (como os servidores *web* ou de *e-mail*) em um único servidor físico, desenvolver novas técnicas de simulação, criar novas estratégias de segurança e robustez, facilitar o desenvolvimento de *software* e muitas outras aplicações.

Neste trabalho são abordados aspectos fundamentais de virtualização, bem como alguns fundamentos de sistemas operacionais necessários para a compreensão do tema deste trabalho. Em seguida são detalhadas as técnicas de virtualização e apresentadas diversas

ferramentas que as implementam. Por fim, são indicadas algumas aplicações para virtualização, um estudo de desempenho de algumas das ferramentas existentes e as conclusões deste trabalho.

Palavras-Chave

Virtualização

Sistemas Operacionais

Máquinas Virtuais

Hipervisores

Sistemas Computacionais

Lista de Acrônimos

VM : *Virtual Machine*;

VMM : *Virtual Machine Monitor*;

ISA : *Instruction Set Architecture*;

API : *Application Programming Interface*;

RAM : *Random Access Memory*;

IVT : *Intel Virtualization Technology*;

IBM : *International Business Machines*;

TI : Tecnologia de Informação;

Sumário

Lista de Figuras

Capítulo I

Introdução

O presente trabalho cuida do tema de Virtualização. Virtualização é uma área do conhecimento, muitas vezes associada a Sistemas Operacionais, que estuda técnicas para particionar um sistema computacional físico em sistemas computacionais virtuais, os quais oferecem um ambiente extremamente similar a um sistema computacional real. A associação à área de sistemas operacionais remete à sua origem, a virtualização foi desenvolvida nos anos 60 pela *IBM* para possibilitar a compatibilização de *software* legado com seus novos *mainframes*.

I.1 Escopo

Este trabalho é um estudo sobre as técnicas de virtualização de computadores, dando ênfase às ferramentas de virtualização *VMWare* e *Xen* que atualmente são amplamente utilizadas comercial e academicamente.

O estudo possui duas partes, uma de fundamentos e outra de aplicação. A parte de fundamentos tem como escopo detalhar os conceitos básicos de virtualização, sistemas operacionais e técnicas de virtualização. A parte de aplicação enfatiza as ferramentas de virtualização, compara o desempenho de algumas ferramentas entre si e mostra algumas aplicações de virtualização.

I.2 Motivação

A virtualização apresenta inúmeros aspectos vantajosos que justificam o interesse em seu estudo. Por possibilitar a utilização de aplicações em sistemas operacionais, ou até em plataformas computacionais, diferentes dos para os quais as aplicações foram concebidas, a virtualização apresenta um enorme apelo de compatibilidade e retrocompatibilidade de sistemas, fornecendo grande economia de tempo e dinheiro ao evitar a necessidade de se refazer sistemas legado.

Para o desenvolvimento de sistemas, a virtualização também possibilita uma grande economia de tempo e dinheiro. Isso ocorre devido à extrema simplificação da instalação de sistemas no cliente, visto que o ambiente computacional utilizado em desenvolvimento e que já está configurado para executar a aplicação é portado junto à aplicação, o que resume a instalação da aplicação a uma simples instanciação da máquina virtual que a contém.

Para a infra-estrutura de tecnologia de informação (TI) de uma empresa, as vantagens do uso de virtualização também são claras. Os serviços de TI de uma empresa são tradicionalmente desenvolvidos para serem executados em servidores dedicados, por razões que variam desde as peculiaridades de configuração do sistema que o serviço necessita, até as questões de disponibilidade de serviço e segurança. Utilizar um servidor físico por serviço encarece a manutenção, pois demanda maior equipe e tempo, além de potencial desperdício de equipamentos, visto que a maioria dos servidores fica ociosa durante a maior parte do tempo. Ao unificar os serviços de TI em um servidor físico executando um servidor virtual para cada serviço, consegue-se obter o tão vantajoso isolamento de serviços combinado a um melhor aproveitamento dos equipamentos e à necessidade de uma equipe menor de manutenção de TI, economizando significativamente tempo e dinheiro.

A virtualização apresenta diversas outras aplicações e motivações para seu estudo, como por exemplo, a possibilidade de desenvolver *softwares* cuja portabilidade independe da plataforma computacional, as acima descritas são apenas alguns exemplos que ilustram a importância desta área do conhecimento.

I.3 Objetivos deste trabalho

Este trabalho tem como objetivo estudar os fundamentos necessários para compreender o que é virtualização, como funciona, quais são as técnicas existentes, quem as programa, qual é o escopo do uso dessas ferramentas, quais têm melhor desempenho e o porquê.

Para exibir as características das ferramentas de virtualização aqui descritas, foram considerados documentos extraídos dos sites oficiais das ferramentas (manuais de usuário, manuais de desenvolvedor, *white papers*, etc), bem como artigos de análise de desempenho, características e aplicações das mesmas.

No capítulo 2 são apresentados alguns aspectos históricos da virtualização, algumas aplicações e motivações para seu estudo, bem como conceitos básicos e alguns conceitos de sistemas operacionais para facilitar o entendimento deste trabalho.

O capítulo 3 apresenta técnicas de virtualização buscando mostrar não somente as diferenças arquiteturais entre as diversas técnicas, mas também as diferenças de desempenho em que estas acarretam.

O capítulo 4 detalha algumas ferramentas de virtualização, tornando possível uma melhor compreensão das diferenças existentes e de quais são as ferramentas mais adequadas para cada cenário de uso da virtualização.

O capítulo 5 apresenta um estudo de desempenho de ferramentas de virtualização utilizadas no mercado e no meio acadêmico.

O capítulo 6 apresenta algumas aplicações para virtualização tanto para utilizações em grande quanto em pequena escala.

O capítulo 7 apresenta as conclusões deste trabalho e observações finais.

Capítulo II

Virtualização

II.1 Introdução à Virtualização

O conceito de máquina virtual não é recente. As primeiras máquinas virtuais foram desenvolvidas pela *IBM* na década de 60 para um sistema operacional experimental denominado M44/44X. Baseado neste sistema operacional experimental, a IBM desenvolveu diversos sistemas comerciais com suporte à virtualização, e dentre estes o mais famoso foi o OS/370 [3]. Naquela época era comum a utilização de *mainframes* e, para fornecer a cada usuário um ambiente individual completo, onde ele pudesse ter sistema operacional, aplicações e dados isolados dos ambientes de outros usuários, lançava-se mão da virtualização, disponibilizando uma máquina virtual para cada usuário e obtendo assim o nível de isolamento e independência desejados.

Além disso, a virtualização era frequentemente utilizada para portar sistemas legados para *mainframes* mais novos. A utilização da técnica de virtualização era imprescindível, visto que, naquela época, diferentes modelos de *mainframes* de um mesmo fabricante costumavam possuir seu próprio sistema operacional e arquitetura[4]. Deve-se ressaltar ainda que, até a década de 80, o *hardware* era o responsável da maior parte do custo dos sistemas computacionais o que tornava muito interessante para os fabricantes de *hardware* como *IBM* o desenvolvimento dessas plataformas virtualizadas para viabilizar a venda de

seus *mainframes* mais novos.

Na década de 80 houve o barateamento do *hardware* e os usuários passaram a possuir estações de trabalho individuais, fazendo com que a virtualização se tornasse um tema menos relevante. A popularização dos computadores também fez com que os sistemas operacionais convergissem para poucas plataformas (*Macintosh*, *Unix* e *Microsoft*) cada uma com um público-alvo e aplicações específicas, o que também fez com que o problema da virtualização perdesse importância.

Com o aumento do poder computacional dos processadores mais recentes, a disseminação de sistemas distribuídos, a proliferação de redes de computadores e o surgimento e sucesso da linguagem de programação *Java*, o problema da virtualização voltou a estar em foco.

Por diversas razões, como o uso de sistemas operacionais distintos e configurações específicas de ambiente, a maioria das aplicações empresariais são feitas para rodarem em um servidor dedicado, o que, até hoje, apresentava-se como a melhor opção. Atualmente, entretanto, devido a esse aumento da capacidade de processamento dos processadores atuais, grande parte dos servidores empresarias ficam ociosos fazendo com que parte do investimento em *hardware* seja desperdiçado. Dessa forma, existe hoje uma tendência a virtualizar esses servidores com capacidade ociosa consolidando-os em um único ou poucos servidores, aproveitando melhor o uso do *hardware*, evitando assim o desperdício de recursos. Além da economia em equipamentos, a consolidação de servidores também possibilita o corte de outros custos como o custo do espaço físico que os servidores físicos eliminados ocupariam, o gasto energético e o custo com refrigeração.

Outra aplicação para a virtualização é agilizar o desenvolvimento de *softwares*, pois pode-se utilizar uma máquina virtual para o ambiente de desenvolvimento e outra para o ambiente de produção. Isto evita erros de configuração no ambiente, versões diferentes de algum componente e agiliza o processo de instalação da aplicação.

A seguir, serão explicados alguns conceitos básicos sobre Sistemas Operacionais, imediatamente seguidos por alguns conceitos básicos sobre virtualização.

II.2 Conceitos Básicos de Sistemas Operacionais

O Sistema Operacional permite e controla o acesso das aplicações aos recursos de *hardware*. Segundo Silberchatz e Galvin [5], a definição clássica de Sistema Operacional é de uma camada de *software* inserida entre o *hardware* e as aplicações que executam tarefas para os usuários e cujo objetivo é tornar a utilização do computador, ao mesmo tempo, mais eficiente e conveniente.

Essa utilização mais eficiente do *hardware* é obtida através de diversos mecanismos para otimizar o aproveitamento dos recursos como processador, memória *RAM*, disco rígido, etc.

Um dos principais mecanismos para obter esse uso eficiente do *hardware* é o mecanismo de execução em multi-tarefas.

Antigamente, os computadores possuíam Sistemas Operacionais que possibilitavam a execução de apenas uma tarefa por vez. Dessa maneira, se uma tarefa estivesse esperando algum evento, como entrada de dados por parte do usuário ou algum dado de outro periférico, por mais que houvesse outras tarefas para ser executadas em seguida, o sistema ficava ocioso aguardando a ocorrência do evento para prosseguir a execução da tarefa corrente, resultando em grande desperdício de recursos.

Com a introdução da execução em multi-tarefas, cada tarefa possui a ilusão de estar executando sozinha no computador, quando na realidade, está compartilhando os recursos com as demais. Ao ser criada, a tarefa recebe do Sistema Operacional um processador lógico associado a ela e memória virtual para seu uso. Para passar a ilusão de paralelismo entre as tarefas, o núcleo do sistema operacional conta com um escalonador de tarefas que divide o tempo de execução do processador em fatias e é responsável por alternar a execução das tarefas chaveando rapidamente os processadores lógicos sobre o processador físico a cada uma dessas fatias de tempo.

A utilização mais conveniente do *hardware* é obtida através de camadas de abstração que escondem do usuário os detalhes dos recursos utilizados, em especial, dos dispositivos

de entrada e saída.

Para exemplificar o uso dessas camadas de abstração pode-se utilizar o conceito de *driver*. *Driver* é uma camada de software que disponibiliza interfaces lógicas para o uso de dispositivos de *hardware*. Com o uso de *drivers*, pode-se abstrair detalhes de implementação do *hardware* acessado e utilizar a interface lógica do *driver* para efetuar as tarefas desejadas. Isso é possível porque as diversas formas de interação com os *hardwares* de diferentes modelos e fabricantes são tratadas pelos *drivers*.

Outro exemplo de uso mais conveniente do *hardware* é o conceito de arquivos. Arquivos não existem nos dispositivos de armazenamento como discos rígidos, *CDs*, *DVDs* ou *pen-drives*, eles são uma abstração que facilita imensamente o uso da capacidade de armazenagem oferecida pelo *hardware*.

Um conceito de suma importância para a compreensão de Sistemas Operacionais é o conceito de processo. O processo é uma abstração que representa um programa em execução, a unidade básica de execução tratada pelo Sistema Operacional. Cada processo é definido por uma região de endereçamento lógico da memória que é composta pelas regiões de código, dados, pilha e *heap*. A região de código é a região de memória onde estão armazenadas as instruções do programa. A região de dados é a região onde estão armazenadas as variáveis globais do programa. A região de pilha armazena o endereço de retorno das chamadas a funções, é utilizada para a passagem de parâmetros das funções e serve também para guardar as variáveis locais de uma função. Por último, mas não menos importante, a região de *heap* é a zona utilizada para armazenar as variáveis alocadas dinamicamente, ou seja, variáveis alocadas em tempo de execução. A figura II.1 representa a organização da memória utilizada por um processo.

Explicando de maneira simplificada, mas suficiente para o escopo deste trabalho, a execução de um processo é basicamente definida por dois registradores lógicos associados a ele. O primeiro registrador lógico é o contador de programa (*Program Counter - PC*) e o segundo registrador lógico é o apontador de pilha (*Stack Pointer - SP*). O contador de programa armazena o endereço de memória da próxima instrução a ser executada en-

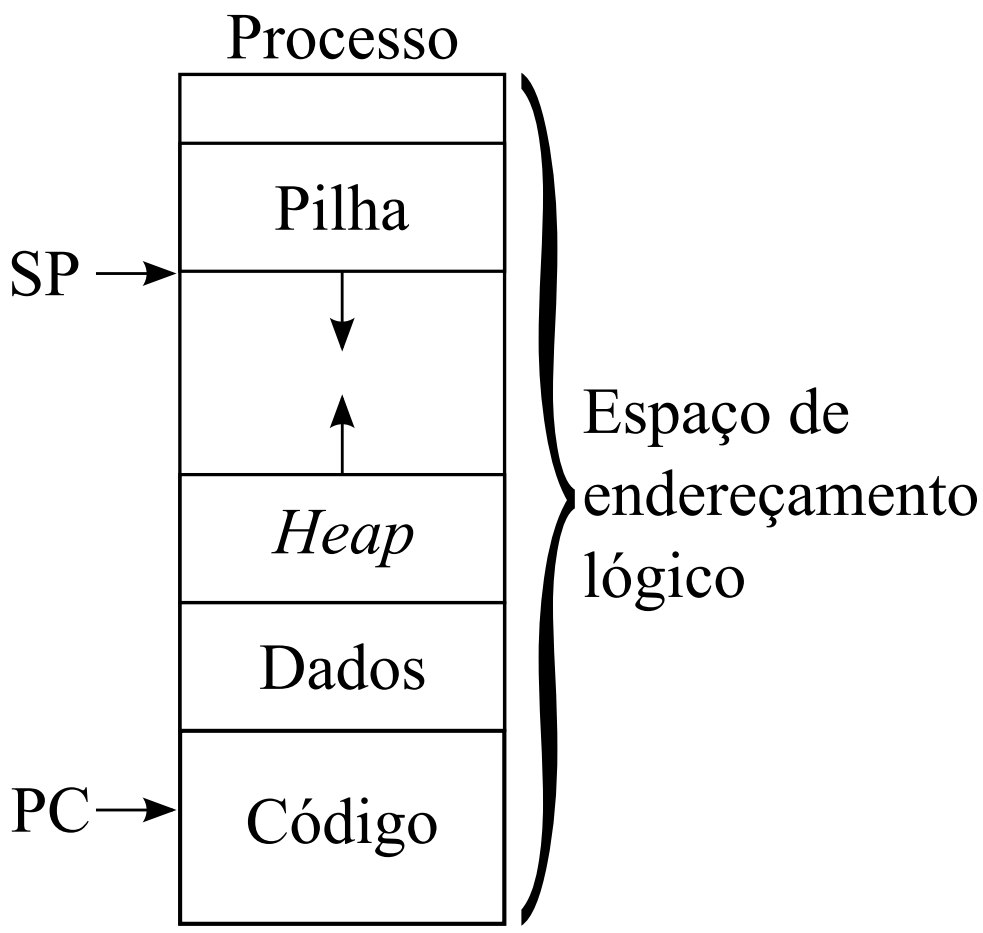

Figura II.1: Organização da memória alocada por um processo.

quanto o apontador de pilha indica onde estão armazenados os endereços de retorno de uma chamada de função, bem como seus parâmetros e variáveis locais. O sistema operacional então, através de seu escalonador de processos e *dispatcher*, mapeia os registradores lógicos PC e SP de cada processo para os registradores PC e SP do processador real, constituindo o chaveamento de contexto. É este chaveamento de contextos (ilustrado na figura II.2) que possibilita a execução de múltiplas tarefas, dando a impressão de paralelismo das execuções.

Outro conceito de suma importância da arquitetura de Sistemas Operacionais é sua construção em camadas hierárquicas com diferentes níveis de abstração e de interfaces. Graças a esta arquitetura em camadas, a evolução dos sistemas computacionais continua, apesar da grande complexidade inerente devida aos diversos componentes de *hardware*

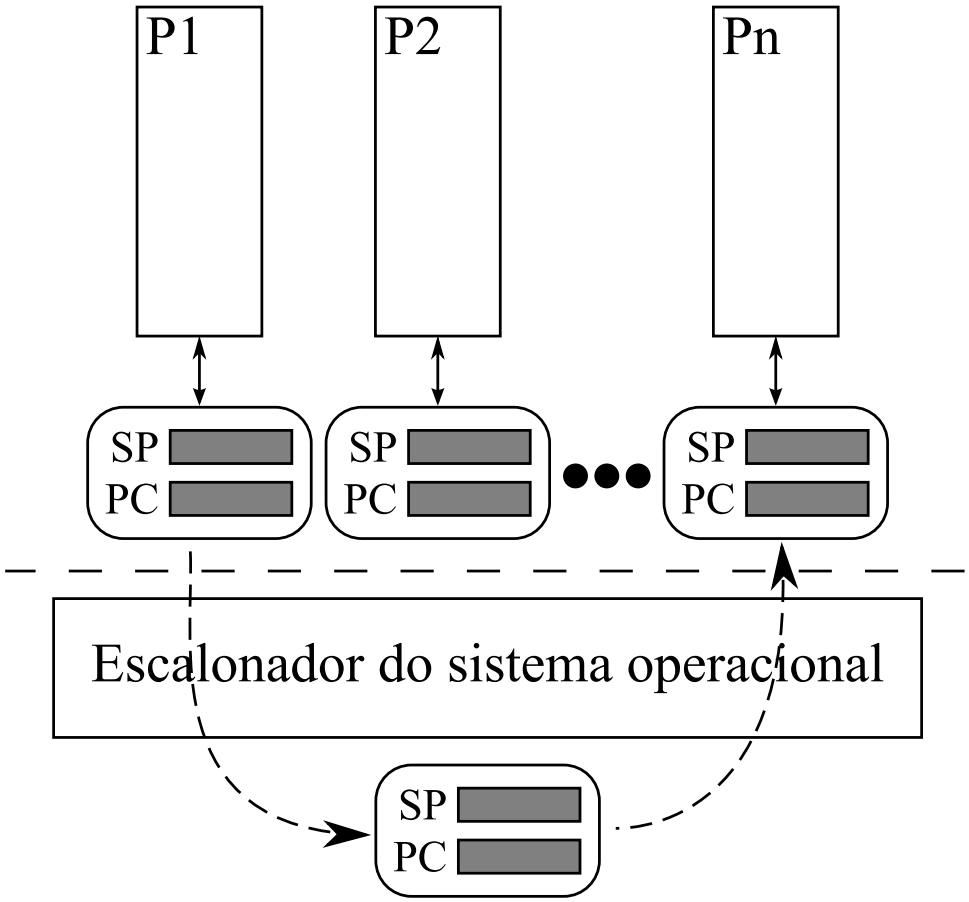

Figura II.2: Exemplo simplificado do chaveamento de contexto.

e *software* que os formam e suas relações. Essa estrutura hierárquica permite diversos níveis de abstração e interfaces bem definidas entre cada componente. Assim, os desenvolvedores de novos componentes de *hardware* e *software* podem se abstrair dos detalhes de tudo que se encontra ao redor do componente desenvolvido, sendo necessário apenas preocupar-se com as interfaces dos componentes que fornecem serviços para ou que usam serviços oferecidos pelo componente em desenvolvimento. Essa característica arquitetural dos sistemas operacionais é fundamental também para que as equipes que desenvolvem cada componente possam trabalhar de forma independente e garantir o funcionamento do sistema final como um todo.

Em geral, um sistema computacional possui quatro tipos de interfaces:

1. instruções de máquina privilegiadas. Este subconjunto das instruções de máquina

também é conhecido como *System Instruction Set Architecture* ou *system ISA*;

2. instruções de máquina não-privilegiadas. Este subconjunto das instruções de máquina também é conhecido como *User Instruction Set Architecture* ou *user ISA*;

3. chamadas de sistema, também conhecidas como chamadas ao sistema operacional;

4. interface aplicativa de programação, também conhecida como *Application Programming Interface - API*).

Como o próprio nome *CPU* diz, *Central Processing Unit*, o principal componente de um computador é o seu processador. Existem diversos processadores e cada um deles possui o seu próprio conjunto de instruções de máquina que pode seguir um padrão. A plataforma mais utilizada hoje é a plataforma *Intel* IA-32, também conhecida como x86, que é implementada por diversos processadores de vários fabricantes como *Intel*, *AMD* e *Via*. Os *softwares* desenvolvidos são compilados para utilizar um determinado conjunto de instruções de máquina e, por esse motivo, um *software* desenvolvido para a plataforma *Intel* IA-64 não irá funcionar na plataforma *Intel* IA-32, por exemplo. O conjunto de instruções de máquina é uma interface entre o *hardware* e *software* sendo dividida em dois grupos: o das instruções privilegiadas, que só podem ser executadas no nível de privilégios mais alto da *CPU*, e o das instruções não-privilegiadas, que podem ser executadas com níveis mais baixos de privilégios da *CPU*. As instruções privilegiadas do conjunto de instruções de máquina são executadas somente pelo sistema operacional e as não-privilegiadas são executadas tanto pelo sistema operacional quanto pelas aplicações do usuário.

As chamadas de sistema são disponibilizadas pelo sistema operacional para acessar recursos que requerem nível mais alto de privilégio. Através delas, de forma indireta e controlada pelo sistema operacional, as aplicações de usuário conseguem acessar os recursos que demandam o uso de instruções privilegiadas, como, por exemplo, acesso aos dispositivos de entrada e saída. Este intermédio do sistema operacional para acessar os recursos de *hardware* através das chamadas de sistema é fundamental para garantir que uma aplicação não acesse recursos de *hardware* alocados para outra aplicação e possibilita

que haja a execução de múltiplas tarefas mesmo com os programas sendo escritos com a filosofia de que vão rodar sozinhos no computador.

A interface aplicativa de programação (*API*) é composta pelas interfaces disponibilizadas por bibliotecas. As bibliotecas costumam fornecer o acesso indireto e de mais alto nível às chamadas de sistema além de fornecer funcionalidades de alto nível comumente utilizadas por desenvolvedores de *software*.

Após esta rápida revisão de alguns conceitos básicos de sistemas operacionais, são apresentados na seção seguinte alguns conceitos básicos de virtualização que farão uso desses conceitos.

II.3 Conceitos Básicos de Virtualização

Nesta seção são apresentados alguns conceitos básicos importantes para a compreensão de o que é e como funciona a virtualização.

É importante expor algumas definições sobre o que é virtualização. Virtualização é um conceito muito amplo e abstrato e possui diversas definições coerentes e que se aplicam de forma mais adequada ou menos dependendo do contexto em que é aplicada. Dentro do escopo deste trabalho, a melhor definição de virtualização é a de que virtualização é uma técnica que permite particionar um sistema computacional físico em diversos sistemas computacionais virtuais denominados máquinas virtuais. As máquinas virtuais fornecem um ambiente computacional completo extremamente similar ao de um sistema computacional físico. Dessa maneira, cada máquina virtual pode possuir seu próprio sistema operacional, aplicações, dispositivos periféricos, etc [3].

Segundo Popek e Goldberg [6], uma máquina virtual é uma cópia isolada e eficiente do *hardware* que se encontra abaixo dela. Ainda segundo Popek e Goldberg, essa definição impõe três propriedades que um programa responsável pelo controle das máquinas virtuais deve satisfazer para ser considerado um monitor de máquina virtual (*Virtual Machine Monitor - VMM*, também conhecido como hipervisor). Estas propriedades são:

1. a eficiência, que impõe que um subconjunto estatisticamente dominante das instruções do processador virtual seja executado diretamente sobre o processador real, sem requerer nenhuma intervenção do monitor de máquina virtual;

2. controle de recursos, que impõe que o monitor de máquina virtual tenha controle completo dos recursos do sistema. Isto requer que seja impossível que um programa arbitrário rodando dentro de uma máquina virtual localizada acima do monitor de máquina virtual afete recursos do sistema, como por exemplo memória *RAM* ou periféricos, que estejam alocados para outra máquina virtual ou para o próprio monitor de máquina virtual.

3. equivalência, que impõe que um monitor de máquina virtual disponibilize aos programas um ambiente que seja essencialmente idêntico ao provido pela máquina original. Formalmente, qualquer programa "P"executando com um monitor de máquina virtual residente na memória, com somente duas exceções possíveis, deve agir de maneira indistinguível da maneira como agiria no caso do monitor de máquina virtual não existir, em que "P"teria acesso às instruções privilegiadas previstas pelo programador. As duas exceções possíveis para o princípio da equivalência resultam da disponibilidade de recursos e de dependências temporais.

Um exemplo de configuração de máquina virtual encontra-se na figura II.3.

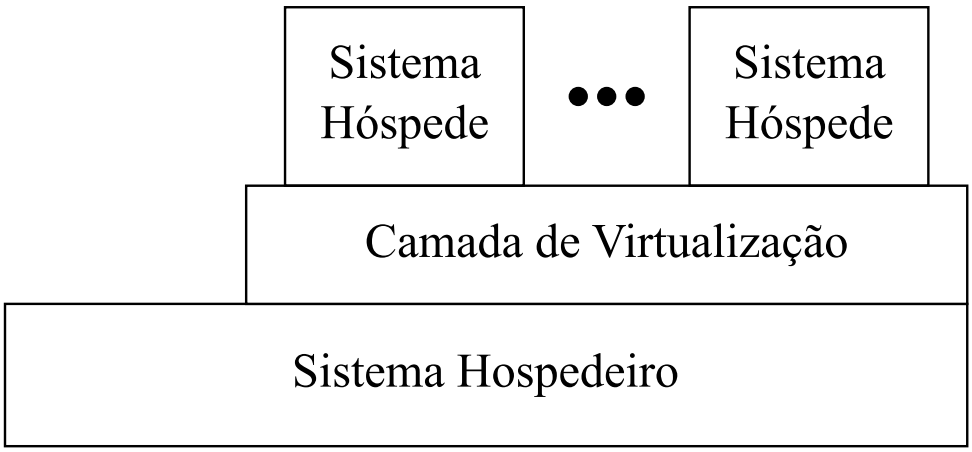

Figura II.3: Exemplo de configuração de Máquina Virtual.

II.3 Conceitos Básicos de Virtualização

Definições mais recentes de virtualização compreendem também a virtualização de recursos de software e não somente os de hardware previstos nas definições clássicas de virtualização.

Segundo Singh[7], virtualização é um *framework* ou metodologia para dividir os recursos de um computador em múltiplos ambientes de execução, utilizando, para isso, um ou mais conceitos e tecnologias como particionamento de *software* e *hardware*, compartilhamento de tempo, simulação completa ou parcial da máquina, emulação e qualidade de serviços. De forma mais prática, virtualização pode ser definida como um conjunto de técnicas que permitem particionar um único sistema computacional em vários outros isolados e independentes. Cada um desses sistemas computacionais originados do particionamento do original fornece um ambiente, denominado máquina virtual, que estende ou substitui as interfaces existentes de forma a imitar as características e comportamento de um outro sistema computacional.

É importante ressaltar que, dentro do contexto de virtualização, existe uma classificação para os sistemas computacionais quanto aos recursos fornecidos/utilizados. Sob esta ótica, um ambiente de máquina virtual possui três partes distintas:

1. o sistema computacional que fornece os recursos a serem virtualizados denominado sistema real, nativo, hospedeiro ou ainda *host system*;
2. a camada de virtualização que constrói as interfaces virtuais a partir das interfaces reais, efetuando a conversão de instruções quando necessário, denominada *hipervisor* ou monitor de máquina virtual (*Virtual Machine Monitor - VMM*);
3. o sistema computacional que utiliza os recursos no ambiente virtualizado denominado sistema convidado, hóspede ou ainda *guest system*.

A figura II.4 ilustra as três partes dessa classificação:

Existem diversas técnicas de virtualização, conforme é detalhado no próximo capítulo, e os recursos virtualizados dependem de qual ponto que a virtualização será inserida dentro da arquitetura de um sistema computacional.

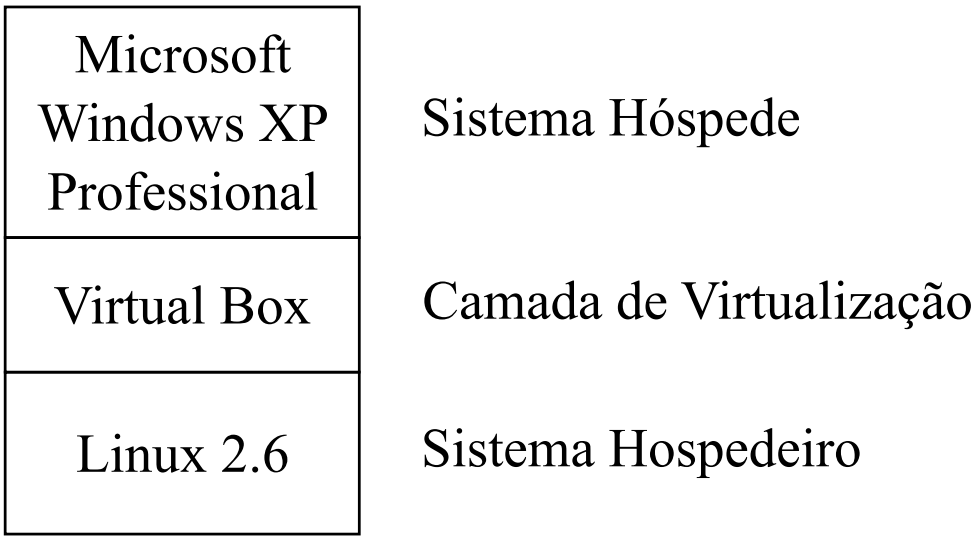

Figura II.4: Camadas de um ambiente com Máquina Virtual.

Para gerar uma máquina virtual capaz de suportar um sistema operacional, por exemplo, é necessário virtualizar o conjunto de instruções de máquina do processador. O sistema operacional é feito especialmente para utilizar a *system ISA* e *user ISA* de uma plataforma específica de *hardware*, como por exemplo, a *Intel* IA-32 (x86). Por isso, a máquina virtual que fornecer a ISA ao sistema operacional convidado deverá disponibilizar a ISA do processador físico tomando o cuidado de lidar com as requisições de acesso ao *hardware* que utilizem recursos compartilhados pelos sistemas convidados e pelo monitor de máquina virtual.

Um outro ponto da arquitetura de um sistema computacional que é interessante virtualizar é o topo de um sistema operacional, que fornece às aplicações as interfaces de programação aplicativa (*Application Programming Interface - API*) e chamadas de sistema. Um exemplo de ferramenta de virtualização que utiliza esta abordagem é o *Wine*, que permite executar aplicações feitas para *Windows* no ambiente *Linux*.

Os dois exemplos de pontos da arquitetura de um sistema computacional onde pode-se inserir uma camada de virtualização servem só para demonstrar algumas das diversas possibilidades de técnicas de virtualização.

Capítulo III

Técnicas de Virtualização

III.1 Introdução às Técnicas de Virtualização

Dependendo de onde se encontra dentro da arquitetura do sistema computacional, um *software* precisa de distintas interfaces para seu correto funcionamento. Em decorrência desses diversos pontos da arquitetura onde o *software* pode estar localizado e das interfaces disponíveis nesses pontos, existem diferentes técnicas de virtualização de forma a suprir os recursos esperados.

Um sistema operacional é desenvolvido para uma determinada plataforma de *hardware*, como *Intel* IA-32 (x86) ou *Intel* IA-64. Dessa forma, para seu correto funcionamento, o sistema operacional precisa utilizar o conjunto de instruções de máquina (privilegiadas e não-privilegiadas) para o qual foi desenvolvido. Por isso, quando a virtualização é aplicada entre o *hardware* e o sistema operacional, a máquina virtual deve prover as interfaces da plataforma de hardware esperadas pelo sistema operacional.

As técnicas de virtualização podem ser aplicadas para particionar o *hardware* físico em diversos *hardwares* lógicos, para emular um *hardware* diferente do verdadeiro, como disponibilizar um *ARM* virtual sobre uma plataforma x86 por exemplo, ou abstrair o *hardware* e sistema operacional ao desenvolver um software, como ocorre nas linguagens

de programação *Java* e *C#*.

III.2 Tipos de Máquinas Virtuais

Existem diversas camadas nos sistemas computacionais, o que gera diversas possibilidades de implementação de máquinas virtuais com características individuais de isolamento, eficiência, robustez, etc. As máquinas virtuais possuem diversas classificações propostas, fato justificado pelos numerosos aspectos relevantes das máquinas virtuais que podem ser utilizados como parâmetro de classificação. Dependendo da complexidade dos sistemas convidados suportados, as máquinas virtuais podem ser divididas em dois grupos [3]:

- Máquinas Virtuais de Aplicação (*Process Virtual Machines*: são ambientes de máquinas virtuais feitos para suportar apenas processos ou aplicações hóspedes específicos. A máquina virtual Java (*Java Virtual Machine - JVM*), capaz de executar somente aplicações *Java*, é um exemplo desse tipo de máquina virtual;
- Máquinas Virtuais de Sistema (*System Virtual Machines*): são ambientes de máquinas virtuais feitos para suportar sistemas operacionais hóspedes completos, com aplicações hóspedes rodando sobre elas. A máquina virtual da ferramenta *VirtualBox*, capaz de suportar, entre outros, o sistema operacional *Windows*, é um exemplo deste tipo de máquina virtual.

As figuras III.1 e III.2 exemplificam os grupos desta classificação.

Outra classificação que pode ser efetuada para máquinas virtuais utiliza como parâmetro de classificação o grau de semelhança entre o conjunto de instruções de máquina (*ISA*) do sistema hóspede e do sistema hospedeiro [3]:

- interfaces equivalentes: o conjunto de instruções de máquina virtualizado oferecido ao ambiente hóspede reproduz o conjunto de instruções de máquina do sistema real,

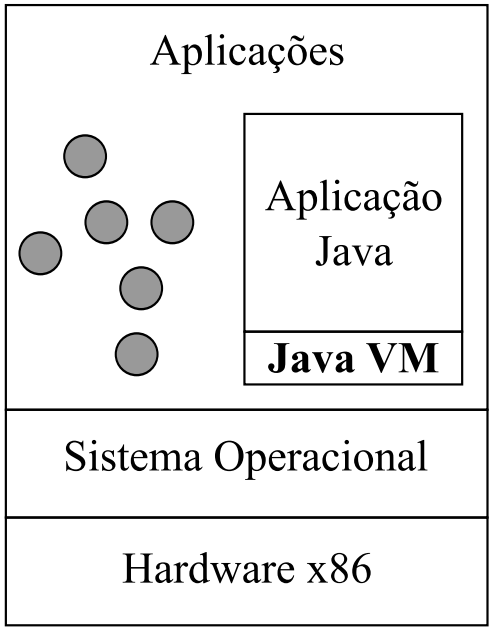

Figura III.1: Exemplo de máquina virtual de aplicação.

tornando possível que os softwares desenvolvidos para o ambiente real sejam executados dentro do ambiente virtual. Como grande parte das instruções de máquina do sistema hóspede pode ser diretamente executada sobre processador real, o desempenho atingido pelas aplicações hóspedes pode chegar muito perto do desempenho atingido ao executar as mesmas aplicações diretamente sobre o sistema real. As instruções que não podem ser executadas diretamente são as instruções sensíveis, que precisam ser monitoradas e tratadas pelo hipervisor para que não ocorram problemas de uso de recursos alocados para outra máquina virtual ou, até mesmo, para o próprio hipervisor;

- Interfaces distintas: o conjunto de instruções de máquina virtualizado oferecido ao ambiente hóspede não possui relação com o conjunto de instruções de máquina do sistema real, ou seja, o conjunto de instruções de máquina virtualizado deve ser interpretado pelo monitor de máquina virtual e convertido em instruções para a plataforma real. Essa interpretação e conversão de instruções por parte do hipervisor possibilitam, por exemplo, executar programas para uma plataforma *ARM, Advanced RISC Machine*, sobre uma plataforma x86 mas, onera muito a eficiência

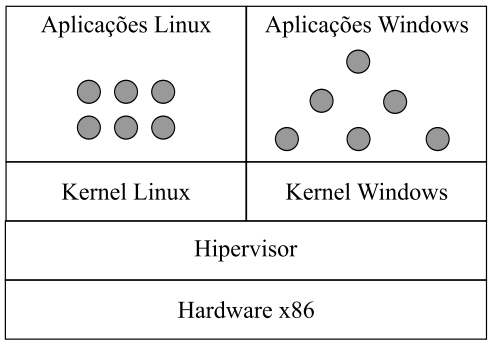

Figura III.2: Exemplo de duas máquinas virtuais de sistema sobre um mesmo hipervisor, uma hospedando *Windows* e outra hospedando *Linux*.

do hipervisor impactando no desempenho do sistema hóspede.

A figura III.3 exemplifica a estrutura dos sistemas computacionais dos dois grupos desta classificação.

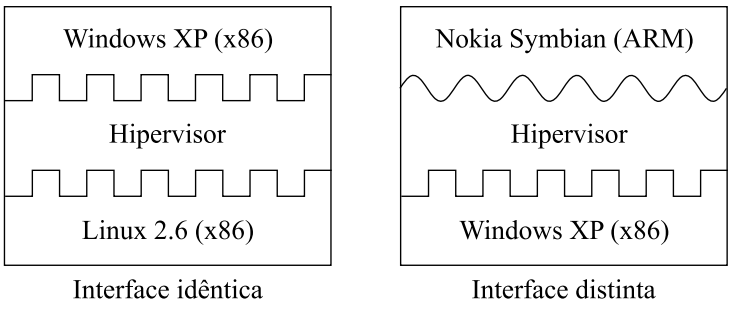

Figura III.3: Exemplo de máquinas virtuais com interfaces equivalentes e com interfaces distintas.

Segundo Rosenblum[8], a virtualização pode ser classificada como:

- Virtualização do *Hardware*: quando o sistema físico é exportado pela camada de virtualização como *hardware* abstrato e semelhante a algum real. Qualquer *software* escrito para a arquitetura real funciona no sistema convidado;
- Virtualização do Sistema Operacional: quando o sistema operacional é exportado pela camada de virtualização como um sistema operacional abstrato semelhante a algum real. A máquina virtual executa aplicações ou um conjunto de aplicações de um sistema operacional específico;

- Virtualização de Linguagens de Programação: quando a camada de virtualização está localizada no topo do sistema operacional. As máquinas virtuais desse tipo são feitas para simular computadores fictícios que servem para um propósito específico. Nessa abordagem, a camada de virtualização fornece as interfaces desse computador fictício para que *softwares* escritos para ele possam ser executados.

O trabalho de Nanda e Chiueh[9] apresenta mais uma classificação para virtualização:

- Abstração da *ISA*: quando a virtualização é implementada através da emulação completa da *ISA*. O emulador executa as instruções desejadas pelo sistema hóspede traduzindo as instruções da *ISA* virtual em instruções que possam ser executadas sobre o processador real. Esta modalidade de virtualização possibilita o uso de *softwares* em plataformas de *hardware* diferentes daquela para qual o *software* foi desenvolvido mas, apresenta baixa eficiência devido ao custo de tradução das instruções embutido pelo emulador;

- *Hardware Abstraction Layer (HAL)*: quando o hipervisor disponibiliza uma arquitetura completa para o sistema hóspede, que acredita estar executando sobre uma plataforma de *hardware* real;

- *OS Level*: quando virtualização é obtida através de uma chamada de sistema específica. A principal vantagem desta técnica é obter uma camada para isolamento de processos, virtualizando cada sistema com *IP* e recursos de *hardware* próprios, mesmo que limitados. O isolamento ocorre a partir de um diretório ou sistema de arquivos separados e previamente preparados para este uso;

- Nível de aplicação ou virtualização de linguagens de programação: esse tipo de virtualização é feito por meio de uma abstração: a camada de execução. Aplicações fazem uso dessa camada para executar as instruções desejadas. Este tipo de solução garante a execução da aplicação independente da plataforma de *software* ou *hardware*, bastando simplesmente que haja a presença de uma máquina virtual, específica para cada plataforma, que implemente as interfaces desta camada de execução;

- *User Level Library Interface*: grande parte das aplicações de um sistema computacional é escrita para utilizar um conjunto de *APIs* fornecidas pelo sistema operacional. A virtualização desta categoria é obtida através da abstração do topo do sistema operacional, sendo que a máquina virtual fornece as interfaces das *APIs* esperadas à aplicação hóspede.

Nas duas subseções a seguir são formalizados os conceitos de máquinas virtuais de aplicação e máquinas virtuais de sistema.

III.2.1 Máquinas Virtuais de Aplicação

Uma máquina virtual de aplicação, também conhecida como *Process Virtual Machine*, tem a capacidade de executar um processo ou aplicação individualmente. Este tipo de máquina virtual é criado sob demanda, no momento que a aplicação hóspede é iniciada, e é destruído no momento que a aplicação convidada é terminada. Dentro do sistema hospedeiro, a camada de virtualização (hipervisor) juntamente com a aplicação convidada são vistos como um único processo, tratado da mesma maneira que os outros processos do sistema hospedeiro e sob as mesmas restrições impostas aos outros processos.

As máquinas virtuais de aplicação costumam ser implementadas por hipervisores que permitem interações entre a aplicação hóspede e os outros processos do sistema hospedeiro. Além disso, não é incomum que este tipo de hipervisor permita que as aplicações convidadas acessem o sistema de arquivos e outros recursos do sistema hospedeiro. Este tipo de hipervisor geralmente disponibiliza a mesma *ISA* do hardware real, permitindo que a aplicação hóspede execute instruções não sensíveis diretamente sobre o processador real, há também, entretanto, os hipervisores que disponibilizam uma *ISA* diferente da *ISA* nativa, traduzindo as instruções oferecidas para instruções nativas. Alguns exemplos de máquinas virtuais de aplicação que disponibilizam a mesma *ISA* do sistema real são os sistemas operacionais multi-tarefas, os tradutores dinâmicos e alguns depuradores de memória.

As máquinas virtuais de aplicação mais comuns atualmente são as que oferecem à aplicação uma interface binária de aplicação (*Aplication Binary Interface* - *ABI*, ou seja, a soma da *user ISA* e chamadas de sistema) diferente da utilizada pelo sistema real. Um exemplo de ferramenta que fornece esse tipo de máquina virtual é o *Wine*, que permite executar aplicações desenvolvidas para *Windows* de forma transparente sobre o *Linux* (utilizando, inclusive, recursos como a *GPU, Graphics Processing Unit*). Como o *Wine* oferece a mesma *API* do *Windows* mas reimplementada, em certas circunstâncias, o desempenho de aplicações executadas em *Linux* sobre o *Wine* chega a ser superior ao desempenho obtido nativamente no *Windows* [10].

Também são comuns hoje as máquinas virtuais de aplicação que fornecem uma *ABI* que não corresponde a nenhum sistema real. Este é o caso da *JVM* (*Java Virtual Machine*) que fornece a *ABI* de um sistema computacional fictício com o intuito de uniformizar a compilação de código das aplicações escritas em *Java*, favorecendo assim sua portabilidade. O código de máquina fictício gerado pela linguagem *Java* é denominado *bytecode* e pode ser executado em qualquer plataforma que possua uma *JVM* implementada capaz de traduzir as instruções da *ABI* deste sistema computacional fictício em instruções e chamadas de sistema do sistema computacional real. Embora do ponto de vista de portabilidade esta abordagem possa parecer excelente, do ponto de vista de desempenho esta abordagem introduz grande sobrecarga devido à necessidade de interpretação do *bytecode* e tradução para instruções e chamadas de sistema do sistema computacional real.

III.2.2 Máquinas Virtuais de Sistema

Máquinas virtuais de sistema são ambientes completos com capacidade para suportar um ou mais sistemas operacionais completos com suas aplicações. Cada sistema operacional convidado possui a impressão de estar executando sozinho sobre a máquina virtual e cada um deles está isolado dos outros, rodando de forma independente.

O isolamento entre as máquinas virtuais desse tipo é tão forte que, normalmente, os sistemas convidados podem interagir uns com os outros somente através dos métodos

utilizados para comunicação entre computadores físicos distintos, como os mecanismos de rede. Alguns sistemas de virtualização, como o *VirtualBox* e *VMWare*, entretanto, possuem mecanismos para compartilhamento controlado de determinados recursos, como, por exemplo, um diretório virtual.

Esse tipo de máquina virtual era utilizado na década de 70 para o particionamento dos *mainframes* em ambientes virtuais individuais para cada usuário. Hoje, este tipo de virtualização volta a ter importância para situações como consolidação de servidores quando há capacidade de processamento ociosa dos servidores físicos.

As classificações desse tipo de máquinas virtuais levam em conta o grau de virtualização do *hardware* e a arquitetura das máquinas virtuais.

Quanto à arquitetura dos hipervisores, as classificações são:

- Hipervisores Nativos (ou de tipo I): quando o hipervisor executa diretamente sobre o *hardware* real. Nesse caso, sua função é multiplexar os recursos de *hardware* para as diversas máquinas virtuais que se encontram sobre ele. Cada máquina virtual tem a impressão de possuir recursos próprios e exclusivos de *hardware*, quando na verdade possui um controle sobre um conjunto de *hardwares* virtuais gerados a partir do particionamento de recursos do *hardware* verdadeiro. Este é o tipo mais antigo de virtualização, era encontrada nos *mainframes* dos anos 70. *VMWare* e *Xen* são alguns dos que implementam esta técnica;

- Hipervisores Convidados (ou de tipo II): nessa categoria, o hipervisor executa como um processo comum dentro de um sistema operacional. Os recursos utilizados pelo hipervisor são providos e controlados pelo sistema operacional subjacente. Exemplos de ferramentas que utilizam essa abordagem são o *VirtualBox* e *QEMU*.

Quanto ao nível de virtualização, as classificações são:

- Virtualização de recursos: quando a *user ISA* é mantida e pode ser acessada diretamente pelo sistema convidado. A *system ISA* e os recursos como disco e periféricos

são virtualizados. Esta é a técnica que permite desempenho mais próximo ao obtido executando-se o sistema convidado diretamente sobre o *hardware* real. *VirtualBox*, *Xen* e *VMWare* pertencem a esta categoria;

- Virtualização completa: nessa categoria toda a interface de *hardware* é virtualizada, podendo, inclusive, haver emulação de uma plataforma de *hardware* distinta disponibilizada para as máquinas virtuais. Esta técnica de virtualização apresenta pior desempenho devido ao custo de processamento introduzido pela tradução das instruções virtuais disponibilizadas em instruções da plataforma real. O *QEMU* é um exemplo dessa categoria.

III.3 Técnicas de Virtualização

Nesta seção serão definidas as técnicas utilizadas atualmente para a implementação de ferramentas de virtualização. A primeira técnica apresentada é a para-virtualização, utilizada na ferramenta de virtualização *Xen*. A segunda técnica, virtualização total, é utilizada por grande parte das ferramentas de virtualização, como por exemplo *VMWare* e *VirtualBox*.

III.3.1 Para-Virtualização

Esta técnica de virtualização consiste na modificação dos sistemas operacionais convidados de forma que eles tenham "consciência"de estarem virtualizados. Este grau de "consciência"é obtido através da modificação do acesso às instruções da *system ISA* de forma que ao querer executar uma instrução sensível, o sistema operacional convidado efetua uma chamada ao hipervisor requisitando a operação desejada.

Dessa forma, a sobrecarga do hipervisor monitorar a execução das máquinas virtuais em busca de instruções sensíveis é eliminada, resultando em grandes ganhos de desempenho. O lado negativo do uso desta técnica é que o sistema operacional convidado deve

ser modificado, o que limita a portabilidade dos sistemas operacionais para ambientes virtuais que implementem esta técnica.

Exemplos de ferramentas que utilizam esta abordagem são o *Denali* e o *Xen*.

III.3.2 Virtualização Total

Esta técnica de virtualização consiste na criação de um ambiente virtual que provê todas as interfaces disponibilizadas por um *hardware* real. As interfaces virtuais fornecem desde uma *ISA* virtual até dispositivos periféricos virtuais, todos similares a dispositivos reis. O ambiente de máquina virtual criado por esta técnica suporta sistemas operacionais convidados completos sem que estes sofram nenhuma modificação, visto que todas as interfaces esperadas são providas pela máquina virtual.

Essa transparência do ambiente virtual oferecida aos sistemas convidados traz alto custo de desempenho. Por não saberem que estão em um ambiente virtualizado, as aplicações e sistema operacional hóspedes executam instruções sensíveis como se todo o *hardware* pertencesse ao sistema convidado. Para não causar problemas de uso dos mesmos recursos por máquinas virtuais diferentes, o hipervisor é sobrecarregado com a tarefa de monitorar a ocorrência destas instruções sensíveis e tratá-las de forma que o sistema convidado acredite que sua requisição foi efetuada e que ao mesmo tempo não cause conflitos em acesso ao *hardware*.

Esta técnica de virtualização é a mais utilizada e é implementada por diversas soluções de virtualização, como *VMWare, VirtualBox* e *VirtualPC*.

III.4 Suporte de *Hardware* à Virtualização

Com a recente volta de interesse pelas técnicas de virtualização, os fabricantes de *hardware* vem desenvolvendo tecnologias incorporadas aos computadores para melhorar o suporte à virtualização.

III.4 Suporte de *Hardware* à Virtualização

Por volta de 2005, os fabricantes de *hardware Intel* e *AMD* passaram a incluir em seus processadores suporte básico à virtualização [3]. A *Intel* desenvolveu a *IVT (Intel Virtualization Technology)* e a *AMD* desenvolveu a *AMD-V (AMD Virtualization)*, ambas com a idéia principal de criar dois modos de operação no processador: o modo *root* e o modo *non-root*.

O modo *root* foi feito para uso do hipervisor e é equivalente ao funcionamento de um processador convencional. O modo *non-root* foi feito para uso das máquinas virtuais. Os dois modos possuem os quatro *rings* de proteção dos processadores normais.

Os *rings* de privilégio dos processadores servem para evitar que código malicioso executado por um usuário consiga acessar diretamente todos os recursos de *hardware*. Dessa maneira, quando um código estiver sendo executado com um nível de privilégio baixo e tentar executar uma instrução que requer nível de privilégio mais alto, o processador não executa a instrução e lança uma exceção, tratada normalmente pelo sistema operacional.

Os sistemas operacionais rodam suas tarefas do núcleo com privilégio máximo (no *ring* 0) e executam as tarefas do usuário no nível de privilégio mais baixo (no *ring* 3). Dessa maneira, as tarefas do núcleo do sistema operacional conseguem acessar todos os recursos sem problemas e, as aplicações do usuário, por rodar em nível de privilégio mais baixo, podem acessar a maioria dos recursos de *hardware* somente por intermédio do sistema operacional, através das chamadas de sistema.

Quando a virtualização de um sistema operacional é feita, o hipervisor deve monitorar o código executado pelo sistema operacional convidado, prevenindo que execute código sensível e acesse diretamente os recursos de *hardware*.

Entretanto, quando o suporte *IVT* está presente e o sistema operacional convidado está executando em modo *non-root*, a responsabilidade de monitorar a execução de instruções sensíveis passa para o processador. Toda vez que o sistema operacional convidado executa código privilegiado, o processador automaticamente gera uma chamada ao hipervisor para que ele trate a instrução sensível.

Por não precisar mais monitorar dinamicamente o código executado pelo sistema con-

vidado graças a este suporte de virtualização, o desempenho dos hipervisores aumentou muito nos últimos anos[11].

Capítulo IV

Ferramentas de Virtualização

IV.1 *Xen*

O *Xen* nasceu como projeto de pesquisa na Universidade de *Cambridge*, sendo em seguida a base para a fundação da empresa *XenSource*. Em 2003, a primeira versão foi liberada experimentando rápida aceitação no mercado de virtualização e, como resultado de seu sucesso, em 2007 a *XenSource* foi comprada pela *Citrix Systems*, que no mesmo ano tornou pública a criação da *Xen AB, Xen Project Advisory Board*, composta por diversas grandes empresas de tecnologia, tais como *IBM, Intel, Hewlett-Packard, Novell, Red Hat, Sun Microsystems, Oracle* e a própria *Citrix*. Entre os objetivos desta organização, está o gerenciamento da tecnologia e a promoção de forma que esta atinja o maior número de plataformas de *hardware*. O código do *Xen*, sob a licença *GLP2, GNU Public License 2*, é aberto, o que promoveu o nascimento de uma comunidade engajada à sua volta. [12]

Xen é uma ferramenta de virtualização com hipervisor específico para a arquitetura x86[13]. A ferramenta de virtualização *Xen* difere em vários aspectos das demais ferramentas existentes. Em primeiro lugar, o *Xen* utiliza o conceitos de domínios, que são, basicamente, diferentes implementações de suas máquinas virtuais. O hipervisor do *Xen* é responsável por gerenciar os recursos compartilhados por estes domínios.

IV.1 *Xen*

Para o *Xen*, existem dois tipos de domínios que diferem, principalmente, no grau de privilégio que possuem:

- Domínio 0 (*Zero*): domínio único composto por um sistema operacional *Linux* modificado com o *Xen*, possui os *drivers* de dispositivos da máquina física e dois drivers especiais utilizados para atender às requisições de acesso à rede e ao disco efetuados pelas máquinas virtuais do domínio U. O domínio 0 é privilegiado e possui acesso total aos recursos de *hardware*. Além disso, o domínio 0 também é responsável por criar, inicializar e finalizar as máquinas virtuais dos domínios do tipo U;
- Domínio U(*User*): domínio não privilegiado que suporta um sistema operacional hóspede. O domínio do tipo U possui *drivers* virtuais que se comunicam com o domínio 0 para ter acesso ao *hardware* real. Existem dois tipos de domínio U, o tipo U-PV e o tipo U-HVM.

Os domínios do tipo *U-PV (User Para-Virtualized)* reconhecem sua natureza virtualizada, bem como a existência de outras máquinas virtuais, colaborando com o hipervisor para atingir melhor desempenho. Estes domínios têm plena consciência que não possuem acesso direto ao *hardware* real e possuem sistemas operacionais hóspedes modificados que efetuam chamadas ao hipervisor toda vez que uma instrução sensível seria executada. Esta técnica de virtualização, denominada para-virtualização, gera grandes ganhos de desempenho, uma vez que a sobrecarga do hipervisor monitorar a execução dos sistemas hóspedes em busca de instruções sensíveis é retirada.

A uma figura IV.1 mostra a arquitetura do *Xen*, enfatizando a relação entre o hipervisor, o domínio 0 e os domínios U.

Os domínios do tipo *U-HVM (User Hardware Virtual Machine)* não reconhecem sua natureza virtualizada e são implementados usando a técnica de virtualização total, podendo utilizar sistemas operacionais hóspedes e aplicações escritas para a plataforma x86 real. Para que se possa utilizar este tipo de domínio, é necessário o uso de processador com suporte às tecnologias de virtualização via hardware *IVT - Intel Virtualization Technology* ou *AMD-V - Advanced Micro Devices-Virtualization*.

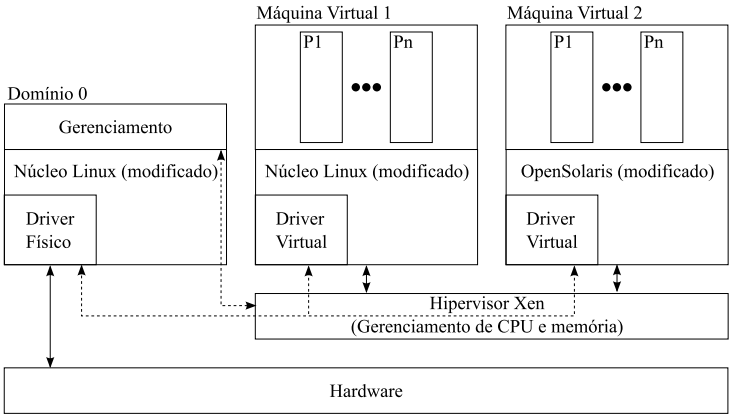

Figura IV.1: Arquitetura do Xen.

Os hóspedes que estão em domínios do tipo *U-PV* possuem *drivers* específicos de acesso à rede e ao disco que interagem com o domínio 0. No caso dos que estão em domínios do tipo *U-HVM*, as requisições de disco e rede são feitas através de um *daemon* do *QEMU* vinculado a cada instância de domínio do tipo *U-HMV*. O *hardware* disponível a cada máquina virtual de um domínio *U-HVM* é o oferecido pelo *QEMU*.

Atualmente, com a inserção de recursos de otimização de virtualização nos processadores, a diferença de desempenho entre a para-virtualização do *Xen* e a virtualização total diminuiu consideravelmente, chegando inclusive a ser mais rápida a virtualização total em alguns casos.

A versão mais nova do *Xen* é a 3.3 que foi lançada em agosto de 2008. Segundo o site oficial do *Xen* [13], o fato do hipervisor ter menos de 150.000 linhas de código e o núcleo do hipervisor ter menos de 50.000 linhas de código torna fácil a análise deste pela comunidade, tornando-o extremamente seguro e fazendo com que a sobrecarga causada pelo hipervisor seja minimizada.

IV.2 *VMWare*

Fundada em 1998, a *VMWare*, empresa que empresta seu nome aos seus produtos, desempenha papel relevante no atual mercado de virtualização. Seu principal diferencial

em relação aos seus concorrentes é o extenso conjunto de *hardwares* virtualizados que possui, adquiridos ao longo dos vários anos de existência, permitindo assim que as máquinas virtuais hóspedes tenham como referência praticamente os mesmos recursos de *hardware*. Na prática, isso facilita que estas máquinas virtuais sejam copiadas de um hospedeiro para outro de forma completamente transparente.

Os produtos da *VMware*, na maioria dos casos, usam a *CPU* para executar código diretamente. Entretanto, quando isso não é possível, nos casos de *real-mode code*, por exemplo, os produtos da *VMWare* reescrevem dinamicamente o código a ser executado – processo chamado de *Binary Translation*. O código reescrito é alocado então em memória, tipicamente no final do espaço de endereços, onde os mecanismos de segmentação os protegem. Este processo, segundo dados fornecidos pela própria *VMWare*, permite que seus produtos operem significativamente mais rápidos do que emuladores – até 80% de ganho de desempenho[1].

A *VMWare* possui uma completa gama de aplicações com o objetivo de explorar as mais diversas oportunidades que a virtualização oferece. Entre os produtos disponíveis e suas respectivas funções estão:

- ***VMware ESX Server 3*:** virtualiza os recursos de *hardware*, tais como, processador, memória, discos e rede, permitindo que um servidor físico seja particionado em várias máquinas virtuais e que cada uma seja vista como uma máquina física em uma infra-estrutura de rede convencional. É a solução de maior porte da *VMWare* e é muito utilizada comercial e academicamente;

- ***VMware ESX Server 3i*:** apesar de possuir as mesmas funcionalidades do *ESX Server*, difere deste em sua arquitetura interna e na forma como alguns procedimentos de gerenciamento são executados;

- ***VMware Virtual SMP*:** permite que uma única máquina virtual utilize mais de um processador físico simultaneamente. Na atual versão, uma máquina virtual pode utilizar simultaneamente até quatro processadores físicos;

- ***VMware VMFS*:** Sistema de arquivos desenvolvido de modo que diversas máquinas acessem concorrentemente o mesmo meio físico de armazenamento;

- ***VMware Server*:** versão gratuita dos produtos *ESX Server*, cujo objetivo é permitir o teste de uso por parte dos possíveis usuários. Porém, como ferramenta de garantia comercial, a *VMWare* disponibiliza apenas imagens pré-configuradas, chamadas *appliances*, com objetivos específicos. Assim, dispõem-se para teste dos usuários, máquinas virtuais para servidores *web, e-mail, DNS*, etc;

- ***VMware Workstation*:** ambiente no qual, efetivamente, se criam as máquinas virtuais. Isso significa que é possível carregar um sistema operacional qualquer nessa máquina virtual e a sua configuração é feita através de ferramenta específica que é parte integrante desse produto.

- ***VMware Fusion*:** para o sistema operacional *MacOS X*, é o produto equivalente ao *VMware Workstation*;

- ***VMware Player*:** é a versão gratuita do produto *VMware Workstation*. Assim como ocorria na versão *Server*, o objetivo é permitir que usuários testem o uso da virtualização e não é possível definir (criar) o sistema hóspede na máquina virtual a partir do zero. Novamente, em seu *site*, a *VMware* distribui uma série de *appliances*, imagens de sistemas hóspedes, que contemplam diferentes distribuições de *Linux* e *Windows Server 2003*.

Os produtos da *VMWare* utilizam a técnica de virtualização total e, com o advento do suporte à virtualização por parte das *CPUs* modernas, a eficiência destes produtos aumentou, obtendo até desempenho próximo ao nativo dependendo da aplicação.

IV.3 *VirtualBox*

VirtualBox é uma família de produtos de virtualização para a plataforma x86 que possui produtos voltados desde o uso residencial até o uso comercial. A plataforma *VirtualBox*

é desenvolvida pela empresa *Sun Microsystems* que fornece a plataforma livremente sob a licença *GPL,GNU Pulic License*. *VirtualBox* possui diversas versões para rodar sobre os sistemas hospedeiros *Windows, Linux, Macintosh* e *OpenSolaris* fornecendo suporte para vários sistemas operacionais hóspedes como *Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows 3.x, Linux* (*kernel* versões 2.4 e 2.6), *Solaris, OpenSolaris* e *OpenBSD*.

Assim como o *Xen, VirtualBox* é desenvolvido em conjunto por uma empresa privada (no caso *Sun Microsystems*) e por uma comunidade o que faz com que seja uma ferramenta que se desenvolva com grande velocidade, graças ao esforço da comunidade, ao mesmo tempo em que é refinada para uso comercial, devido ao esforço da empresa privada.

O projeto do *VirtualBox* utiliza como base para alguns de seus componentes o projeto *QEMU*. Alguns dos dispositivos de *hardware virtual* do *QEMU* foram utilizados como ponto inicial de desenvolvimento para os do *VirtualBox*. Além disso, o mecanismo de tradução dinâmica do *QEMU* é utilizado como recurso de contingência do *VirtualBox* para os casos em o gerenciador de máquinas virtuais próprio não funciona corretamente, como por exemplo nas situações em que o programa hóspede executa em modo real.

Atualmente o *VirtualBox* suporta as tecnologias de virtualização da *Intel* (como a *Intel VT-x*) e da *AMD* (como a *AMD SVM*).

VirtualBox é um produto de virtualização total. Isso significa que ele permite que a instalação dos sistemas operacionais hóspedes e suas aplicações sejam feitas em uma máquina virtual sobre o sistema hospedeiro sem que haja a necessidade de modificar seus códigos-fonte. Por se tratarem de máquinas virtuais de plataformas x86 que executam sobre um sistema real x86, as máquinas virtuais do *VirtualBox* permitem que grande parte do código executado no sistema hóspede rode sobre o sistema hospedeiro diretamente, sem necessidade de tradução. Entretanto, as instruções que afetam outras máquinas virtuais ou o sistema hospedeiro são monitoradas e, quando ocorrem, são tratadas pelo hipervisor do *VirtualBox* de maneira que o sistema hóspede acredite que sua requisição foi executada exatamente como o previsto pelos seus desenvolvedores. Para exemplificar a

situação descrita anteriormente, pode-se pensar no acesso ao disco por parte da máquina virtual. Enquanto o sistema operacional hóspede faz um acesso ao disco virtual pensando que está acessando o *hardware* verdadeiro, o hipervisor do *VirtualBox* intercepta esse acesso e efetua a operação desejada no disco virtual sobre o arquivo utilizado por ele para armazenar a máquina virtual. Dessa forma, o sistema operacional hóspede tem sua requisição atendida e não há conflitos no uso do *hardware*.

Segundo o site oficial do *VirtualBox* [11], existem diversos cenários que tornam a virtualização interessante sendo os principais:

1. fornecer suporte de sistemas operacionais. Utilizar uma ferramenta de virtualização como *VirtualBox* possibilita rodar uma aplicação desenvolvida para outro sistema operacional (aplicação para *Macintosh* sobre *Windows* ou aplicação para *Solaris* sobre *Linux*, por exemplo) sem a necessidade de reiniciar o computador. Esta utilização é extremamente prática para desenvolvedores, pois frequentemente, ao testar um sistema, são necessários elementos que rodam em plataformas diferentes da utilizada pela aplicação em desenvolvimento e o uso de máquinas virtuais para estabelecer o ambiente completo onde a aplicação funciona traz grande economia de tempo e recursos de *hardware*.

2. Consolidação de Infra-estrutura. Como hoje é comum a subutilização da capacidade de processamento dos computadores, torna-se interessante unir diversos computadores lógicos em poucos computadores físicos, obtendo assim grande economia de dinheiro e energia.

3. Teste de aplicações instáveis e recuperação em caso de desastres. Com o uso do recurso de *snapshots*,*backup* do estado de uma máquina virtual em um determinado momento, é possível restabelecer rapidamente uma máquina virtual para um estado prévio caso algo de errado aconteça, tornando seguro o teste de aplicações instáveis.

O *VirtualBox* provê alguns recursos interessantes como [14]:

- Controlador *USB* virtual: um *driver* virtual *USB* que permite que dispositivos *USB* sejam utilizados no sistema convidado sem que haja a necessidade de instalar o *driver* específico para o dispositivo dentro do sistema hospedeiro;
- *Remote Desktop Protocol (RDP)*: o *VirtualBox* implementa este protocolo padrão de maneira que quando a máquina virtual for configurada para rodar um servidor *RDP*, esta pode ser acessada remotamente usando qualquer *thin client* que seja compatível com este protocolo.
- *USB over RDP*: este recurso permite que dispositivos *USB* conectados no cliente remoto possam ser utilizados e acessados pela máquina virtual.

O *VirtualBox* é uma ferramenta bastante madura de virtualização que está em constante desenvolvimento e que a cada dia acumula diversos novos recursos.

IV.4 *User-Mode Linux*

User-Mode Linux é uma ferramenta de virtualização *open-source* baseada na adaptação do *kernel* do *Linux*. Desenvolvida e mantida principalmente por Jeff Dike, esta ferramenta permite instanciar diversas versões do *Linux* sobre ela, fornecendo uma máquina virtual cujo disco virtual é um único arquivo dentro da máquina física.

Assim como no *VMWare* e *Xen*, no *User-Mode Linux* é possível configurar o acesso ao *hardware* desejado que a máquina virtual possua, dessa forma, configurando-se a máquina virtual com os limites de acesso ao *hardware* apropriados, é possível proteger a máquina real e o *software* nela presente.

O *User-Mode Linux* é bastante utilizado pela comunidade *Linux*, fazendo parte da árvore de desenvolvimento do *kernel*.

Um dos seus usos é o de testar novas versões do *kernel* ou distribuições *Linux*. Dessa forma, ao sair uma nova versão do *kernel*, pode-se executá-la dentro de uma máquina

virtual do *User-Mode Linux* durante algum tempo antes de atualizar o *kernel*, certificando-se que a nova versão não tenha sofrido mudanças que gerem instabilidade em elementos frequentemente utilizados no sistema corrente.

Um outro uso desta ferramenta de virtualização é o desenvolvimento ou modificação do *kernel*. O uso de uma máquina virtual do *User-Mode Linux* constitui um excelente laboratório para testar um *kernel* recompilado pois, em caso de instabilidade, basta fechar a máquina virtual e voltar ao desenvolvimento, o que economizada grande quantidade de tempo quando comparado ao processo tradicional de testar o *kernel* recompilado sobre a própria máquina física.

IV.5 *QEMU*

QEMU é uma ferramenta genérica de virtualização e de emulação desenvolvida por Fabrice Bellard e com código *open-source*.

Quando o *QEMU* é utilizado com a opção de emulação ativada, é possível executar sistemas operacionais e programas sobre ele que foram desenvolvidos para outras plataformas, como por exemplo programas para plataforma *ARM* em uma plataforma *PC*. A emulação de plataformas computacionais distintas é obtida através de tradução dinâmica e chega a atingir níveis de desempenho muito bons [15].

Quando utilizado como ferramenta de virtualização, o *QEMU* atinge desempenho próximo ao nativo por conseguir executar código (instruções não-sensíveis) diretamente no processador. Esse modo de operação requer que tanto o sistema hospedeiro quanto o sistema hóspede usem um processador compatível com a plataforma x86.

O *QEMU* foi utilizado como ponto inicial para o desenvolvimento do *VirtualBox*, que, até hoje, utiliza alguns de seus mecanismos em condições excepcionais.

IV.6 *Microsoft VirtualPc*

O *VirtualPC* é a solução de Virtualização da *Microsoft*. A versão atual do *VirtualPC* é o *VirtualPC2007*, que é distribuído gratuitamente pela *Microsoft*.

Segundo a companhia [16], a distribuição gratuita da ferramenta se justifica pela visão da empresa de que o valor da virtualização não está nas ferramentas que possibilitam a virtualização, mas sim nas ferramentas de gestão dos ambientes virtuais e nos sistemas operacionais convidados.

A estratégia de virtualização utilizada por esta ferramenta é a virtualização total. A ferramenta provê ambientes virtuais completos para suportar sistemas operacionais convidados.

Capítulo V

Comparação de Desempenho das Ferramentas de Virtualização

V.1 Testes de Desempenho de Ferramentas de Virtualização

Nessa sessão serão apresentadas algumas comparações de desempenho entre as principais ferramentas de virtualização. Essas comparações irão analisar o desempenho de algumas dessas ferramentas entre si e em relação ao desempenho nativo.

No início de 2007, em uma nota divulgada pela *VMWare* [1], é realizado um estudo de desempenho comparando o produto *ESX* da *VMWare* com o *XEN* e com os mesmos processos rodando nativamente no servidor. Os testes, mostrados na figura V.1, foram realizados utilizando ferramentas usadas no dia-a-dia, tais como o compilador de *C/C++ gcc*, e os programas de compressão de dados *gzip* e *bzip2*.

Nesta figura, pode-se notar que a perda de desempenho introduzida pelo *VMWare* é de, em média, 3%, enquanto que o Xen introduz 6% de perda. No entanto, esses testes usam métodos tradicionais de medição de tempo, que podem dar resultados imprecisos quanto medidos dentro da máquina virtual. Além disso, o *Xen* foi executado com sua con-

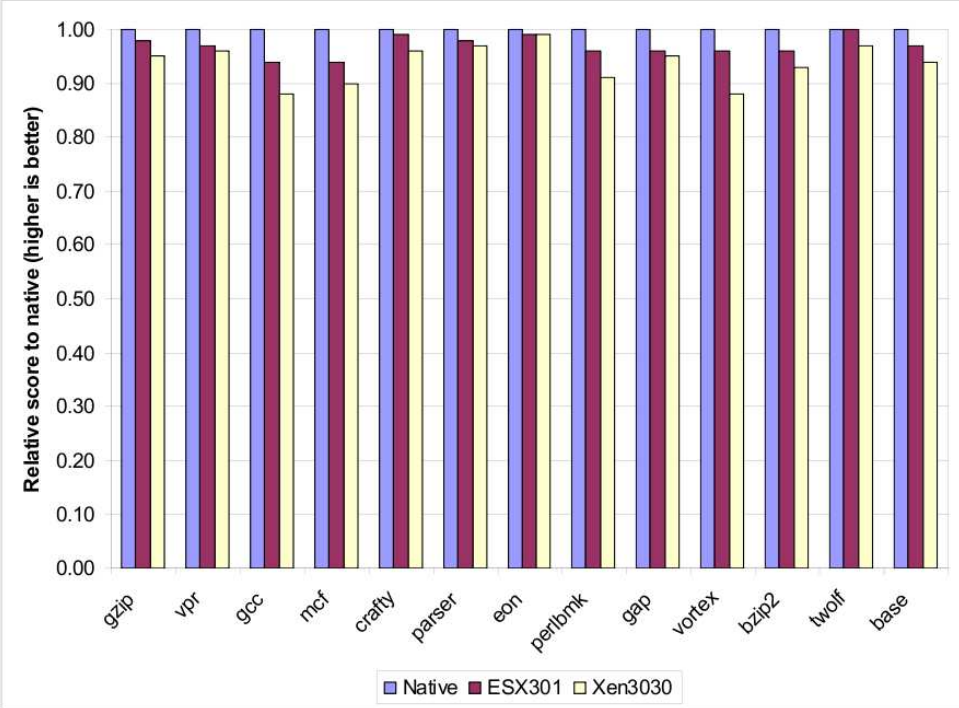

Figura V.1: Comparação de desempenho entre *VMWare, Xen* e execução nativa [1].

figuração padrão, diferentemente de como acontece em servidores reais, onde o hipervisor é configurado de acordo com as características da máquina hospedeira.

Em outro estudo, Camargos et al. [2] comparam a desempenho entre diferentes métodos de virtualização, incluindo sistemas completamente virtualizados, para-virtualizados, e virtualização a nível de sistema operacional. Os testes realizados exploram os diferentes aspectos de cada método de virtualização. A figura V.2 mostra o desempenho em duas tarefas: na esquerda a compilação do núcleo do *Linux* e na direita a tarefa de compressão da imagem de um CD. Em ambos os gráficos, 1.0 é definido como o desempenho da tarefa sendo executada no *Linux* hospedeiro.

A tarefa de compilação, mostrada na parte da esquerda da figura V.2, demanda muita CPU e muito acesso a disco. Nessa figura fica claro que as soluções de virtualização total sofrem uma considerável perda de desempenho. O *Linux-VServer* e o *OpenVZ* são soluções de virtualização a nível de sistema operacional e ficam respectivamente em primeiro e terceiro lugar. Nesse teste, *Xen* é uma máquina para-virtualizada, e fica

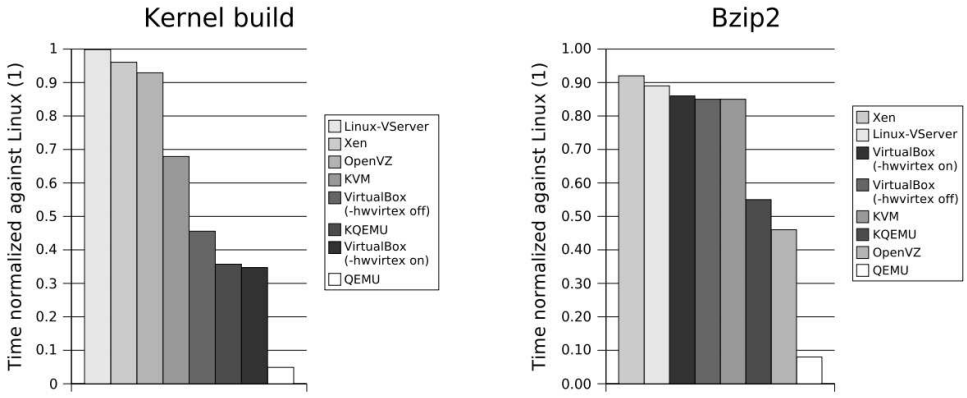

Figura V.2: Comparação entre técnicas de virtualização [2]. Na esquerda, o desempenho relacionado à compilação do núcleo do *Linux*. Na direita, a compressão de aproximadamente 700MB de dados com o *bzip2*.

em segundo lugar. A tarefa de compressão com o *bzip2*, mostrada na parte da direita da mesma figura, demanda muita memória e processamento, mas pouco acesso a disco. Nesse caso, a diferença entre o *VirtualBox*, que faz virtualização completa, e as técnicas de para-virtualização e virtualização a nível de sistema operacional é bem menor. Isso se deve ao fato de que o hipervisor, que roda como um processo do sistema operacional hospedeiro, não necessita fazer muitas interceptações de I/O, e o código convidado roda nativamente na CPU real.

V.2 Problemas de Testes de Desempenho para Ferramentas de Virtualização e Soluções

Efetuar testes de desempenho em ambientes virtualizados não é uma tarefa tão simples. A acurácia dos métodos tradicionais de medição de desempenho, baseados nos números de ciclos de *clock* necessários para efetuar uma tarefa, pode não ser tão alta quando esses métodos são aplicados a ambientes virtualizados. A distorção nos resultados ocorre pelo fato das ferramentas tradicionais não avaliarem a sobrecarga imposta pela presença da camada de virtualização.

V.2 Problemas de Testes de Desempenho para Ferramentas de Virtualização e Soluções

Não existe, no entanto, uma ferramenta de *benchmark* imparcial para avaliar o desempenho de hipervisores. Isso levou a *Intel* a definir a metodologia *vConsolidate* [17] para avaliação de desempenho de sistemas virtualizados. Essa metodologia consiste em uma bateria de testes efetuados com simuladores de servidores comuns de empresas como servidor de *web* ou *e-mail*, todos eles operando em alta carga. A pontuação de desempenho é medida tanto com servidores nativos quanto com servidores virtualizados. Como o uso tradicional de ciclos de *clock* para medir desempenho, utilizada nos *benchmarks* tradicionais, não é válido para o caso de aplicações virtualizadas, a metodologia *vConsolidate* mede o desempenho pelo fluxo de saída de dados de várias máquinas executando os testes concorrentemente, ao invés de medir o tempo que cada uma leva em cada tarefa.

Nos próximos anos é bastante provável que um *benchmark* imparcial para virtualização seja implementado utilizando a metodologia do *vConsolidate*. Por enquanto, as ferramentas de *benchmark* para virtualização são consideradas tendenciosas, explorando pontos específicos das arquiteturas dos sistemas que desejam favorecer, como por exemplo a ferramenta de *benchmark* de virtualização da *VMWare*, o *VMmark* [18].

Capítulo VI

Algumas Aplicações para as Técnicas de Virtualização

VI.1 Novas Possibilidades Através da Virtualização

A tecnologia de virtualização, ao quebrar o antigo paradigma de um sistema operacional por unidade de *hardware*, tem permitido o desenvolvimento de soluções e ferramentas que prometem expandir os horizontes da computação de grande escala. Potencialmente, grandes mercados, tais como o financeiro, de varejo e de prestação de serviços, podem sofrer drásticas mudanças, cujo objetivo, em última análise é o do aumento da eficiência. Nas seções a seguir algumas das soluções propiciadas pela tecnologia de virtualização são evidenciadas.

VI.2 Consolidação de Servidores

A virtualização permite o melhor aproveitamento dos recursos de *hardware* ao permitir que diversas máquinas virtuais operem em uma mesma máquina real, prática usualmente chamada de consolidação de servidores. Atualmente, tal prática é altamente difundida,

sobretudo para aplicações leves que não requeiram grande desempenho do *hardware*. Esta prática garante grande economia de energia e de recursos de hardware quando há capacidade ociosa nos servidores reais originais.

VI.3 Serviços de Alta Disponibilidade

Utilizando técnicas de virtualização, é possível desenvolver modelos específicos de atendimento a serviços de alta demanda, explorando à escalabilidade dos ambientes virtuais. Na prática, se um *data center* que disponibiliza através da internet um serviço, passar a sofrer um súbito aumento no número de requisições, pode-se rapidamente aumentar a capacidade de resposta deste serviço. Para isso, bastaria replicar a máquina virtual onde tal aplicação estivesse instalada em outros servidores, migrá-la para servidores ociosos e/ou de maior capacidade ou até mesmo para outros *data centers*, equilibrando a carga.

A virtualização também é ideal para possibilitar rápida resposta a incidentes em serviços de alta disponibilidade. Hipervisores possuem ferramentas de monitoramento das máquinas virtuais. Utilizando essas ferramentas, uma vez detectado qualquer problema que comprometa a disponibilidade de um serviço em uma máquina virtual, o hipervisor, corretamente configurado, pode optar pela remoção desta máquina virtual e a sua rápida substituição por uma nova.

VI.4 Reprodução de Ambientes Multi-Plataforma e Portabilidade dos Ambientes de Desenvolvimento

Outro uso para a virtualização é a criação de ambientes muito mais consistentes para o desenvolvimento e teste de novos aplicativos para os mais diversos sistemas operacionais. Tais ambientes, embarcados em máquinas virtuais, poderiam ser distribuídos para os desenvolvedores e estes poderiam levá-los consigo para os mais diversos lugares, desde a sala ao lado até a sua casa, através da prática de *home office*. Adicionalmente, empre-

sas interessadas em criar provas de conceitos para seus produtos de *software*, poderiam desenvolver máquinas virtuais de modo que toda a experiência do usuário pudesse ser facilmente reproduzida.

Usuários interessados no uso de múltiplos sistemas operacionais podem facilmente fazê-lo através do uso de máquinas virtuais. O uso de ferramentas como *VirtualBox* ou *Wine* permitem o uso contemporâneo de aplicações de plataformas distintas, agilizando a execução de tarefas e aumentando a comodidade do usuário.

VI.5 Melhoria dos Procedimentos de Segurança da Informação

O uso de virtualização possibilita o aumento de segurança da informação. Uma das possíveis medidas é a implementação de *Honeypots* virtuais, ou seja, máquinas virtuais propositalmente expostas na rede com vulnerabilidades intencionais para atrair terceiros mal-intencionados. Com o uso de *Honeypots*, as empresas podem iludir o intruso com a falsa sensação de sucesso da invasão, preservando os recursos realmente vitais. O uso de honeypots virtuais também é extremamente interessante para estudar as técnicas de invasão de *hackers* e auxiliar no desenvolvimento de proteções para os ataques identificados.

VI.6 Navegação na Internet Menos Exposta a Danos

O acesso à Internet em seu uso rotineiro, muitas vezes não demanda persistência de dados localmente, podendo ser feito através de uma máquina virtual descartada após sua utilização. Em contrapartida, operações críticas, tais como transações comerciais, poderiam lançar mão de máquinas virtuais com recursos de segurança muito mais avançados e utilizados somente em tais ocasiões. Dessa forma, o usuário poderia proteger seu computador real de *softwares* maliciosos que porventura estejam presentes em algum *site* visitado, pois estes iriam causar danos à máquina virtual, mantendo a máquina real

intacta.

VI.7 Facilitar a Gerência de Laboratórios de Informática para Ensino

Instituições de ensino que dispõem de laboratórios de informática podem se utilizar de máquinas virtuais específicas para cada uma das disciplinas. Dessa maneira, apenas os aplicativos específicos estariam disponíveis para o aluno. Outro aspecto deste uso de virtualização é que o gerenciamento do ambiente necessário para aprendizado de uma disciplina torna-se muito mais simples, bastando gerenciar apenas uma máquina virtual deste ambiente e replicá-la em todos os computadores do laboratório.

Capítulo VII

Conclusões

Para finalizar este estudo, é importante ressaltar alguns pontos expostos ao longo deste trabalho.

Conforme apresentado no capítulo de técnicas de virtualização, cada técnica possui suas características que a tornam mais ou menos eficiente para uma determinada aplicação.

Para o uso em serviços de alta disponibilidade, máquinas virtuais de aplicação não são uma opção muito apropriada, visto que as camadas de *software* subjacentes introduzem vulnerabilidades e competem com os recursos disponíveis de maneira muito mais ativa do que em outras técnicas de virtualização. Para este tipo de aplicação, as máquinas virtuais de sistema com um hipervisor subjacente bem configurado para o gerenciamento de recursos seria uma opção muito mais eficiente. Além de garantir a alocação necessária de recursos, essa técnica permite o melhor desempenho do serviço, pois a fina camada de virtualização introduz menor sobrecarga de processamento.

Para o caso em que o requisito mais importante é a portabilidade, as máquinas virtuais de aplicação são uma opção apropriada. Prova disso é o crescente uso de linguagens de programação que utilizam máquinas virtuais deste tipo como *Java* ou *C#*.

Conforme apresentado no capítulo de desempenho, quanto maior o nível de abstração

fornecido pela virtualização, pior é o desempenho das aplicações. Ferramentas de virtualização com níveis de abstração mais baixo como *Xen* e *LinuxV-Server* apresentam desempenho superior a ferramentas de nível de abstração mais alto como *VirtualBox*, vide a figura V.2.

Outro fator de suma importância para o desempenho é se a plataforma hóspede é do mesmo tipo da plataforma hospedeira, possibilitando a execução de instruções não-sensíveis diretamente sobre o *hardware* real. Há grande perda de desempenho com a necessidade do uso de tradução dinâmica como ocorre com a emulação de *hardware* do *QEMU*.

Conforme pode ser concluído após a leitura dos capítulos que precedem este, a virtualização é uma área de conhecimento muito extensa e que fornece inúmeras possibilidades. Alta disponibilidade de serviços, rápida escalabilidade, utilização mais eficiente dos ativos de *hardware* e fácil criação de ambientes controlados para execução de testes são apenas algumas das possibilidades exploradas hoje das técnicas de virtualização.

Com o desenvolvimento conjunto das ferramentas de virtualização e das plataformas de *hardware*, as barreiras de desempenho impostas hoje à virtualização podem ser vencidas, tornando realidade muitas outras possíveis aplicações desta área do conhecimento. É bem provável que no futuro todos os dispositivos eletrônicos possuam diversos perfis para serem ativados nos momentos apropriados. Dessa forma, seria possível, por exemplo, possuir um perfil específico para trabalho no telefone celular que, dentre outras coisas, poderia filtrar as ligações que não fossem de trabalho ou urgentes, disponibilizar somente os dados de trabalho e as aplicações utilizadas ao trabalhar.

Outra possibilidade que se pode vislumbrar é a existência de um computador pessoal virtual onipresente. Cada pessoa poderia possuir um computador virtual que migrasse para o dispositivo que estivesse em uso. Dessa maneira, aonde quer que uma pessoa estivesse, ela sempre teria seus dados e aplicações disponíveis para utilizar.

A virtualização pode ter grandes impactos positivos para o meio-ambiente. Uma vez que a capacidade ociosa dos sistemas computacionais pode ser drasticamente reduzida

com o uso da virtualização, grandes quantidades de energia e outros recursos cuja geração oneram o meio-ambiente podem ser economizados.

Existe um mundo de possibilidades com o uso da virtualização e aquelas que apresentarem mais vantagens competitivas e comercias deverão ser as que irão se concretizar.

Referências Bibliográficas

[1] "A performance comparison of hypervisors". http://www.vmware.com/pdf/, 2007. (Acesso em Janeiro 2009).

[2] F. L. Camargos, B. des Ligneris e G. Girard, "Virtualization of linux servers - a comparative study", *Proceedings of the Linux Symposium*, vol. 1, 2008.

[3] M. A. P. Laureano e C. A. Maziero, *Virtualização: Conceitos e Aplicações em Segurança*. Gramado, RS: Sociedade Brasileira de Computação, Setembro 2008.

[4] A. Carissimi, *Virtualização: da teoria a soluções*. Rio de Janeiro, RJ: Sociedade Brasileira de Computação, Maio 2008.

[5] A. Silberchatz e P. Galvin. Rio de Janeiro: Campus, 2001.

[6] G. J. Popek e R. P. Goldberg, *Formal requirements for virtualizable third generation architectures*, vol. 17. Julho 1974.

[7] "An introduction to virtualization". http://www.kernelthread.com/publications/, 2004. (Acesso em Janeiro 2009).

[8] M. Rosenblum, *The reincarnation of virtual machines*. ACM Press, 2004.

[9] S. Nanda e T. Chiueh, *A survey on virtualization technologies*. University of New York, 2005.

[10] "Site oficial do wine". http://www.winehq.org/, 2009. (Acesso em Janeiro 2009).

[11] "Site oficial do virtualbox". http://www.virtualbox.org/, 2008. (Acesso em Dezembro 2008).

[12] "Site oficial da citrix". http://www.citrix.com/, 2009. (Acesso em Janeiro 2009).

[13] "Site oficial do xen". http://www.xen.org/, 2009. (Acesso em Janeiro 2009).

[14] *Manual de Usuário do VirtualBox*. No. 2.1.2, Sun Microsystems, 2009. (Acesso em Janeiro 2009).

[15] "Site oficial do qemu". http://bellard.org/qemu, 2009. (Acesso em Janeiro 2009).

[16] "Site oficial do virtualpc". http://www.microsoft.com/windows/products/winfamily/virtualpc/d
2009. (Acesso em Janeiro 2009).

[17] J. P. Casazza, M. Greenfield e K. Shi, "Redefining server performance characterization for virtualization benchmarking", *Intel Technology Journal*, vol. 10, no. 3, no. 3, 2006.

[18] "Site oficial do vmware vmmark". http://www.vmware.com/products/vmmark/, 2009. (Acesso em Janeiro 2009).