UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

**Jorge Augusto Guedes Viana**

# Estudo do ambiente Xen e suas Ferramentas

Niterói
2010

**JORGE AUGUSTO GUEDES VIANA**

# Estudo do ambiente Xen e suas Ferramentas

**Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação**

Orientador: Prof. Luiz Valter Brand Gomes

Niterói
2010

Jorge Augusto Guedes Viana

# Estudo do ambiente Xen e suas Ferramentas

**Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação**

**BANCA EXAMINADORA**

_____

Prof. MSc. LUIZ VALTER BRAND GOMES

Orientador

UFF

_____

Prof. Dr. ROSANGELA LOPES LIMA

UFF

_____

Prof. Dr. LEONARDO CRUZ DA COSTA

UFF

Niterói

2010

# RESUMO

Com a popularização das técnicas de virtualização surge o desafio de como manter e realizar as tarefas administrativas em um ambiente com dezenas de máquinas físicas e centenas de máquinas virtuais.

Este trabalho estuda a tecnologia de virtualização, e mais detalhadamente o funcionamento de um ambiente de virtual baseado em Xen. Como é inviável a construção de um ambiente virtual complexo sem a utilização de *softwares* complementares, também são estudadas as funcionalidades de algumas ferramentas utilizadas na construção e administração de um ambiente virtual, principalmente as que se propõem a transformar um conjunto de *hosts* de máquinas virtuais em nuvem.

**Palavras Chave:**
Virtualização, *hypervisor* Xen, Ferramentas de gerência, Nuvem.

# ABSTRACT

With the popularity of virtualization techniques comes the challenge of how to maintain and perform administrative tasks in an environment with tens of physical machines and hundreds of virtual machines.

This paper studies the virtualization technology, and more in detail the functioning of a virtual environment based on Xen. As it is impossible to build a complex virtual environment without use of additional software, are also studied features of some tools used to build and manage a virtual environment, especially those intended to transform a set of virtual machine hosts in a cloud.

**Keywords:**
Virtualization, hypervisor Xen, management tools, cloud computing.

**LISTA DE SIMBOLOS**

| | |
|---|---|
| AWS: | Amazon Web Service |
| DNS: | Domain Name Server |
| DRBD: | Distributed Replicated Block Device |
| GNU: | GNU is Not Unix |
| GPL: | Gereral Public Licence |
| HTTP: | HyperText Transfer Protocol |
| HVM: | Hardware Virtual Machine |
| IaaS: | Infrastructure as a Service |
| IRC: | Internet Relay Chat |
| NAS: | Network-Attached Storage |
| NFS: | Network File System |
| NTI: | Núcleo de Tecnologia e Comunicação |
| NWC-mode: | Non Work Conserving Mode |
| OCFS2: | Oracle Cluster File System 2 |
| OvS: | Open vSwitch |
| PaaS: | Platform as a Service |
| PCI: | Peripheral Component Interconnect |
| QoS: | Quality of Service |
| RAM: | Random Access Memory |
| RPC: | Remote Procedure Call |
| SaaS: | Software as a Service |
| SAN: | Storage Area Network |
| SEDF: | Simple Earliest Deadline First |
| SMP: | Symmetric Multi-Processing |
| SNMP: | Simple Network Management Protocol |
| vCPU: | Virtual Computer Processor Unit |
| VMM: | Virtual Machine Monitor |
| WC-mode: | Work Conserving Mode |
| XCP: | Xen Cloud Platform |
| XML: | Extensible Markup Langua |

# Sumário

**LISTA DE FIGURAS**

# Capítulo 1 - Introdução

Com o avanço da tecnologia de virtualização, nota-se o aumento de serviços sendo providos por máquinas virtuais. Para que o administrador de um ambiente virtual relativamente complexo consiga atender as suas demandas (i.e: administrar, monitorar, aprovisionar, balancear e prover recuperação de desastres), este necessita de ferramentas que possam ser integradas às especificidades do ambiente além de prover um certo grau de automatização de tarefas.

O Núcleo de Tecnologia da Informação e Comunicação (NTI) da Universidade Federal Fluminense (UFF) iniciou em 2009 um projeto de virtualização visando otimizar a utilização do espaço físico da sala de servidores, onde encontravam-se muitos computadores executando sistemas de pequeno porte. Estes computadores além de ocupar muito espaço exigiam uma infra-estrutura de energia e resfriamento cada vez maior, limitando diretamente a quantidade de serviços hospedados pelo NTI.

Durante a realização do projeto de virtualização, foi notada a falta de documentação sobre o assunto gerando atrasos desnecessários no andamento do projeto. Um dos maiores problemas encontrados durante o projeto foi a dificuldade na escolha da estrutura de suporte, como armazenamento, pois existe uma grande variedade de tecnologias e ferramentas disponíveis.

## 1.1 Objetivo

O objetivo deste trabalho é suprimir a deficiência de material encontrada durante a realização do projeto de virtualização do NTI da UFF.

Assim, este material disponibiliza informações suficientes para administradores que queiram desenvolver um projeto de virtualização baseado no *hypervisor* Xen utilizando *softwares* livres em sistemas operacionais GNU/Linux. Para isto será estudado em detalhes o *hypervisor* Xen, e algumas ferramentas que a sua utilização foi considerada no projeto de virtualização realizado pelo NTI da UFF.

## 1.2 Organização

Este trabalho está organizado em 5 capítulos: Introdução; Virtualização; Xen; Ferramentas e Tecnologias; Conclusão.

No capítulo 2 é introduzido o conceito de virtualização, tipos de virtualização, benefícios do uso desta técnica e quais tipos de sistemas são mais propícios para serem virtualizados.

O *hypervisor* Xen é visto detalhadamente no capítulo 3, assim como as peças que compõem o seu ambiente virtual, suas funções e funcionamento.

No capítulo 4 são estudados algumas ferramentas disponíveis para a construção de um ambiente complexo e flexível explicitando suas funcionalidades e objetivos.

Por fim, são mostrados no capítulo 5 conclusões e idéias para trabalhos futuros.

# Capítulo 2 - Virtualização

## 2.1 Introdução

Quando os primeiros sistemas operacionais surgiram, por volta dos anos 60, enquanto um *job* esperava pelo fim de uma operação de entrada e saída do computador IBM 7094, o processador permanecia ocioso, ficando freqüentemente de 80 a 90 por cento do tempo total esperando pelo fim de uma operação de entrada e saída. Para resolver isso, a memória do OS/360[1] foi dividida em diversos pedaços em que cada *job* dispõe de uma partição da memória, assim enquanto um *job* espera pelo final de uma operação de entrada e saída outro *job* poderia ser processado, mantendo assim o processador ocupado aproximadamente 100 por cento do tempo. Este conceito foi chamado de multiprogramação.[1][2]

Essa procura por conseguir respostas mais rápidas e maximizar a utilização do processador levou a criação de sistemas *time-sharing*, onde vários usuários podiam ter terminais interativos e o processador alocado em turnos para os *jobs* dos usuários, introduzindo o conceito de processo. Um processo está associado ao seu espaço de endereços de memória onde é possível ler e escrever. Para proteger periféricos de serem utilizados por um segundo processo sem que o primeiro tenha terminado, as instruções de entrada e saída devem ser feitas através do sistema operacional. Com isso, o sistema operacional agregou funções de gerenciar o processador, a memória e os periféricos para processos.[1]

Assim, um sistema operacional é uma camada de software localizada entre o hardware e os processos que executam tarefas para os usuários.

Em 1979 um grupo de pesquisadores da IBM[2] resolveram separar as funcionalidades de multiprogramação e a capacidade de prover uma interface mais conveniente para o usuário, criando o VM/370 [1]. A parte do sistema que prove a multiprogramação, conhecido como VMM (*virtual machine monitor*), disponibiliza várias máquinas virtuais que são cópias exatas de todas as características do *hardware* físico, possibilitando a execução de diferentes sistemas operacionais em cada máquina

[1] OS/360 é o sistema operacional da IBM.
[2] IBM – International Business Machines.

virtual. Quando um processo dentro de uma máquina virtual executa uma chamada de sistema, esta é capturada pelo sistema operacional dentro da máquina virtual que faz uma operação de entrada e saída. Por sua vez a operação de entrada e saída realizada pelo sistema operacional da máquina virtual é capturada pelo VM/370 que a realiza como parte da simulação da máquina real.

Assim, a técnica de virtualização consiste em inserir uma camada de *software* entre a máquina física e o sistema operacional, provendo ao sistema operacional os mesmos mecanismos de abstração do processador, memória e periféricos que o sistema operacional fornece aos processos. Possibilitando, assim como os processos, a execução de diversos sistemas operacionais simultaneamente no mesmo *hardware*.

Os sistemas modernos de virtualização possuem um componente, como o VMM no VM/370, que controla a interação entre as máquinas virtuais e o *hardware* físico. Este componente é chamado de *hypervisor*. Xen é um dos vários *hypervisors* disponíveis e será visto com mais detalhes no Capítulo 3.

## 2.2 Benefícios da Virtualização

Podemos citar como alguns dos benefícios da virtualização:

- Economia de energia
- Economia de espaço físico
- Economia de equipamento
- Fácil administração
- Alta disponibilidade e balanceamento de carga
- Automatização de tarefas

Com a diminuição dos preço dos computadores da plataforma x86, tem se visto desde meados de 1990, uma mudança na diretriz de *data centers*, onde agora investem em um paradigma descentralizado. Ao invés de investirem grandes quantias em *mainframes*, investem em servidores mais baratos da plataforma x86, adquirindo mais servidores conforme o crescimento da demanda e de serviços prestados, tendo assim um crescimento computacional horizontal.

Alguns efeitos diretos dessa estratégia são os altos gastos em energia elétrica e a difícil manutenção. “Em um *data center* típico, a utilização média é de somente 20%-30%” [24], isso se dá pela utilização de técnicas de balanceamento e alta

disponibilidade que mantêm computadores esperando a falha do equipamento principal ou além do necessário para atender em altos picos de demanda. “Esse desperdício, somado à perdas de energia de infra-estrutura e resfriamento, aumentam as necessidades de energia de 50%-100%” [25]. Além de todas essas perdas, ainda há um grande desperdício em sua aquisição visto que mesmo tendo equipamentos com baixos índices de utilização, esta estratégia não utiliza computadores existentes para o processamento de novos serviços.

A técnica de virtualização permite a consolidação de serviços em menos recursos computacionais, possibilitando alocar diversos serviços em um único computador sem comprometer o isolamento. O *hypervisor* disponibiliza um *hardware* genérico uniforme para as máquinas virtuais. Ou seja, uma máquina virtual pode ser executada por um *hypervisor* em outro computador de arquitetura semelhante. Muitos *hypervisors* modernos permitem que as máquinas virtuais sejam migradas através da rede para outra máquina física sem interrupção. No Xen essa característica é chamada de *live migration*. Assim máquinas virtuais podem ser migradas através de um *cluster* de *hosts* para balanceamento de carga. Máquinas virtuais podem ser salvas em execução. Portanto, no caso de ocorrer uma falha de equipamento a máquina virtual pode ser iniciada do ponto salvo em outro *host*, provendo alta disponibilidade.

Um ambiente virtualizado provê algumas facilidades para a administração como: uma máquina virtual pode ser mantida como modelo para a criação de outras; o sistema pode ser configurado para iniciar e parar máquinas virtuais em uma data e hora especifica; ferramentas e *scripts* para resposta automática a acidentes; ferramentas *web* de administração de máquinas virtuais; ferramentas *cluster-aware* que tem ciência do *cluster* como um todo possibilitando a realização de balanceamento de carga e alta disponibilidade.

Uma prática comum é manter máquinas virtuais modelos com sistemas operacionais diferentes, assim quando uma máquina virtual é requisitada a criação da máquina se resume apenas na cópia do disco da máquina virtual modelo e a configuração de parâmetros como número de processadores virtuais e tamanho da memória. Esta prática é útil para manter homogeneidade nas configurações do sistemas operacional como regras de *firewall*, servidores de DNS, aplicativos básicos, usuários e etc.

Alguns sistemas tem uso restrito a certos períodos, o ambiente virtual pode facilmente ser configurado para iniciar e parar máquinas virtuais em horários ou dias específicos liberando recursos.

Ferramentas de monitoramento podem ser integradas a *scripts* com objetivo de prover respostas automáticas a acidentes ou eventos, tais como: iniciar uma máquina virtual caso a principal falhe; migrar caso o sistema operacional ou *hardware* do *hosts* apresente algum problema; aumentar a memória da máquina virtual em execução se necessário.

O administrador de um ambiente virtual necessita de ferramentas para gerenciar um *cluster* de máquinas virtuais. Hoje existem diversas ferramentas de gerenciamento que podem ser classificadas quanto a interface com o usuário. As interfaces mais básicas são acessadas através de um terminal por linha de comando. A maior parte de ferramentas de gerência são aplicativos clientes que se conectam a interfaces disponibilizadas pelos *hypervisores*. Hoje é notada a migração destas ferramentas para interfaces *web* dando a possibilidade de acesso remoto precisando somente de um navegador de internet com Java ativado, para acesso aos terminais das máquinas virtuais. Outras características importantes encontradas em algumas ferramentas de gerência *web* são: nível de acesso de usuários e sistema de cobrança, permitindo que *data centers* ofereçam serviços de hospedagem de máquinas disponibilizando uma interface de administração dos recursos contratados para usuário.

Ferramentas *cluster-aware* são usadas para automatizar o *cluster*, portanto, a ferramenta torna-se a própria administradora do *cluster* não necessitando de muitas intervenções. Ainda existem poucas ferramentas com esta característica, a maioria delas sendo soluções proprietárias fugindo do escopo deste trabalho.

## 2.3 Tipos de Virtualização

Destacam-se quatro arquiteturas de virtualização: emulação, virtualização completa, paravirtualização e virtualização em nível de sistema operacional. Cada uma provendo um nível de isolamento e performance.

Na emulação, a máquina virtual pode ter uma arquitetura diferente da arquitetura do *hardware* físico, o que não é possível em outros tipos de virtualização.

Uma instrução de uma máquina virtual emulada precisa ser traduzida para a arquitetura do *hardware* na qual está executando. O sistema operacional hospedado não precisa ser modificado. Assim, de dentro da máquina virtual não se consegue detectar que o sistema operacional está executando em um ambiente virtualizado. A figura 1 demonstra um ambiente utilizando emulação.[3][4]

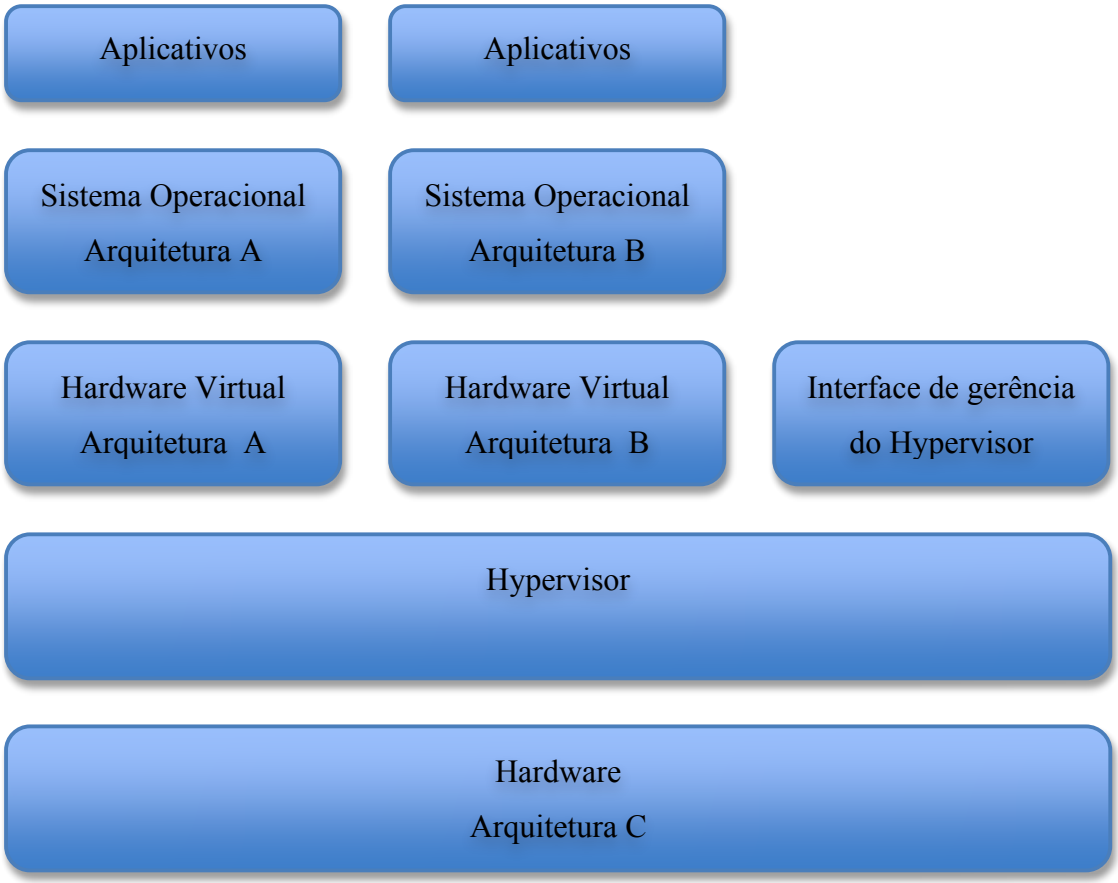


**Figura 1 - Relação *hardware* e ambiente virtual utilizando emulação.**

Um ambiente de virtualização completa e um ambiente de emulação são semelhantes em algumas características. Como na emulação, sistemas operacionais e aplicativos hóspedes de ambiente de virtualização completa são inalterados. A principal diferença entre as duas arquiteturas é que no ambiente de virtualização a arquitetura da máquina virtual é a mesma do *hardware* físico, como mostrado na figura 2, possibilitando que algumas instruções não precisem ser traduzidas pelo *hypervisor* sendo executadas diretamente no *hardware* físico subjacente. Muitos sistemas de virtualização completa requerem suporte de *hardware* para assisti-las.

Estas duas arquiteturas de virtualização são muito usadas para o desenvolvimento de sistemas operacionais.[3][4]

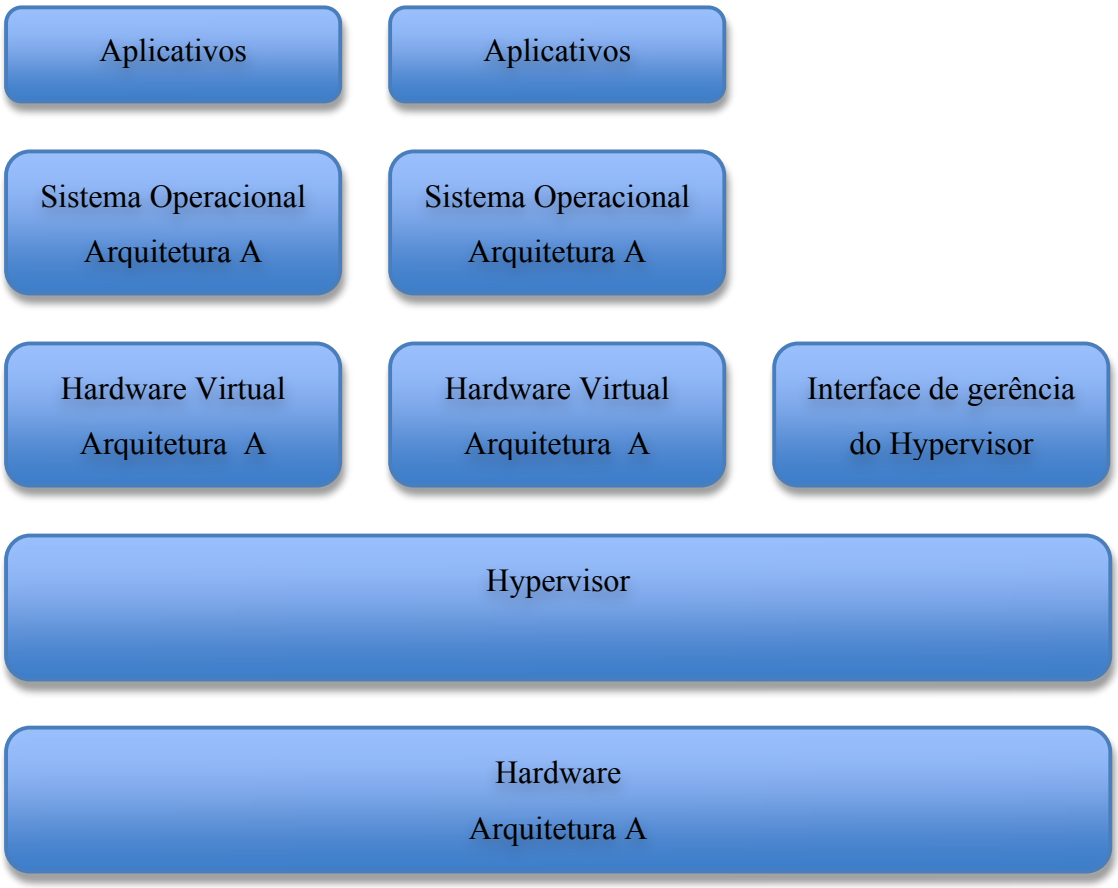


**Figura 2 - Relação *hardware* e ambiente virtual utilizando virtualização completa.**

Na técnica de paravirtualização, o *kernel* do sistema operacional hospedado na máquina virtual tem ciência de que está sendo executado em um ambiente virtualizado e assim utiliza a estrutura disponibilizada pelo sistema de virtualização evitando que instruções sejam traduzidas ou modificadas para serem adequadas. Para que isto seja possível o *kernel* do sistema operacional na máquina virtual tem que ser modificado para suportar especificamente o *hypervisor* desejado, como ilustrado na figura 3. Essa maior integração entre o *hypervisor* e o sistema operacional hospedado fornece um maior controle por parte do *hypervisor* da máquina virtual. Entretanto, o isolamento de uma máquina virtual utilizando a técnica de paravirtualização não é tão forte quanto o de uma máquina virtual utilizando técnicas de emulação ou virtualização completa, e em alguns casos pode encontrar-se efeitos colaterais da

paravirtualização. Esta técnica é utilizada pelo *hypervisor* Xen para disponibilizar um ambiente com o baixo *overhead* e altamente escalável. O *hypervisor* Xen permite a criação de ambientes mistos onde se encontram máquinas virtuais utilizando tanto técnicas de paravirtualização como de virtualização completa, esta através do QEMU[3].[3][4]

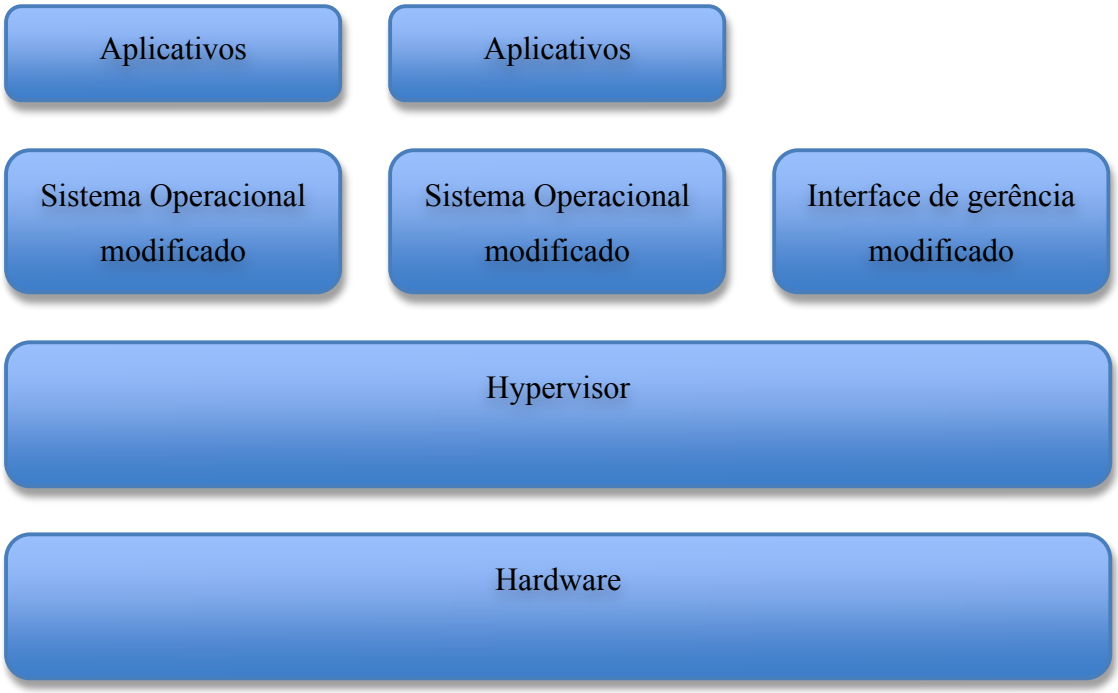


**Figura 3 - Relação *hardware* e ambiente virtual utilizando paravirtualização.**

A virtualização em nível de sistema operacional, ou virtualização por *container*, habilita a execução de múltiplos ambientes de sistemas operacionais em cima de um único *kernel*, assim os sistemas hóspedes tem uma forte dependência com o sistema hospedeiro. Este tipo de virtualização não permite a execução de outros *kernels* de outros sistemas operacionais, como ilustrado na figura 4.[4][6]

[3] QEMU é um emulador genérico de código livre com a capacidade de emular processadores e dispositivos.

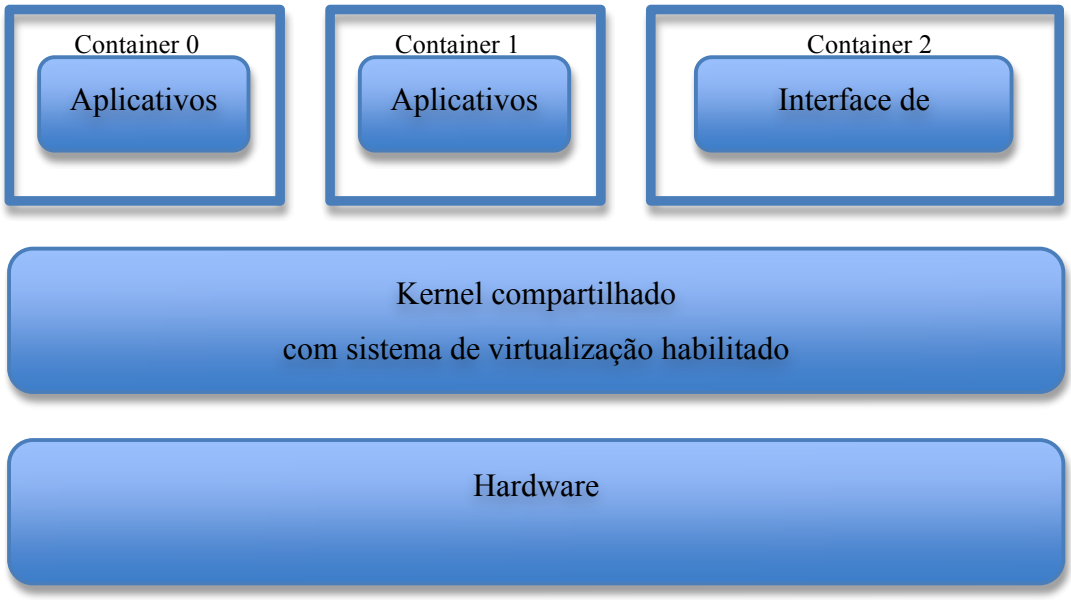


**Figura 4 - Relação hardware e ambiente virtual a nível de sistema operacional.**

## 2.4 Sistemas propícios a serem virtualizados

Não são todos os sistemas que podem ser virtualizados. Como visto anteriormente, muitos serviços não utilizam todo os recursos computacionais do computador em que estão alocados. Antes de virtualizar um serviço já existente devem ser levantadas informações de utilização de recursos como processador, memória, rede e disco.

Um serviço que realiza muitas operações de entrada e saída, especialmente em disco (i.e: banco de dados), não é aconselhável que seja virtualizado por conta do *overhead* da virtualização neste tipo de operações, podendo ter redução de performance.

A memória do *host* limita diretamente a quantidade de máquinas virtuais em execução. Por exemplo, um *host* com 10 Gigabytes de memória RAM, se for

reservado 1 Gigabyte de memória para o sistema operacional da máquina virtual que servirá de interface de gerência do *hypervisor*, esse host poderá por exemplo executar 9 máquinas virtuais de 1 Gigabyte de RAM. Então, é importante que seja monitorada a quantidade de memória disponível para criação de novas máquinas virtuais.

Uma regra geral para ajudar na escolha de quais sistemas serão virtualizados é evitar os sistemas que utilizam muitos recursos. Ferramentas como Cacti [4] e MRTG [5] monitoram o uso de recursos de servidores através do protocolo SNMP[6].

## 2.5 Computação nas Nuvens e Virtualização

Computação nas nuvens, ou *cloud computing*, é o acesso a computadores e serviços através da internet ou rede local abstraindo os detalhes de sua implementação dos usuários[4]. Quando um usuário requer recursos da "nuvem", uma fração dos recursos disponíveis é alocada ao usuário até que esse libere. Assim recursos são retirados da "nuvem" quando utilizados, e retornados quando ociosos. Uma "nuvem" pode ser classificada quanto ao tipo de serviço oferecido ou tipo de acesso. Quando classificado quanto ao tipo de serviço, as categorias são divididas em relação a porção da pilha de *software* disponibilizada como serviço, as principais são: *Infrastructure as a Service, Platform as a Service, Software as a Service*. Quando classificado pelo tipo de acesso, as categorias são dividias em relação ao acesso do usuário a aprovisionamento de recursos virtuais e físicos, e são divididos em: pública, privada e híbrida.

Nuvens são freqüentemente construídas acima de uma camada de virtualização. A abstração dos recursos de diversos *hosts* em uma única coleção de recursos é realizada por ferramentas, algumas delas são apresentadas no Capítulo 4.

[4] Cacti: Ferramenta de monitoração de recursos, usada para gerar gráficos.
[5] MRTG: Ferramenta usada para gerar gráfico de utilização de recursos de alguns equipamentos.
[6] SNMP: Protocolo utilizado para adquirir informações de servidores e serviços.

- Classificação por tipo de Serviço

*Infrastructure as a Service*

Nuvens do tipo *Infrastructure as a Service* (IaaS) provem acesso a coleções de recursos virtualizados, incluindo máquinas, redes, e *storage*. Com nuvens IaaS usuários podem desenvolver seu próprio *cluster* virtual na qual são responsáveis por instalar, manter e executar sua própria pilha de *software*.

*Platform as a Service*

Nuvens do tipo *Platform as a Service* (PaaS) provem acesso a ambientes de programação e rotinas escaláveis com estruturas de dados embutidas para que o usuário possa desenvolver e executar seus aplicativos dentro do ambiente oferecido pelo provedor de serviço.

*Software as a Service*

Nuvens do tipo *Software as a Service* (SaaS) permitem acesso a coleções de aplicativos controlados e executados no provedor, também referenciado como *software* por demanda.

- Classificação por tipo de Acesso

Nuvem Pública

Nuvens públicas provem acesso a recursos computacionais através da internet. Provedores de nuvem pública permitem que clientes aluguem recursos conforme a sua necessidade. Nuvens públicas oferecem acesso a recursos temporários aos usuários sem a necessidade em investimento em infra-estrutura de *data center*.

Nuvem Privada

Nuvens privadas permitem acesso a recursos computacionais hospedados dentro da própria infra-estrutura organizacional. Usuários provisionam e escalam coleções de recursos da nuvem privada.

Nuvem Híbrida

Nuvens híbridas combinam recursos de uma ou mais nuvens públicas e uma ou mais nuvens privadas para seus usuários.

Nuvens de diferentes tipos de serviços podem ser empilhadas. Como mostra a figura 5, serviços SaaS podem ser hospedados em uma nuvem PaaS, ou podem localizar-se diretamente acima de uma nuvem IaaS. No capítulo 4 são apresentadas algumas ferramentas que auxiliam o administrador a desenvolver, manter e controlar nuvens do tipo IaaS.

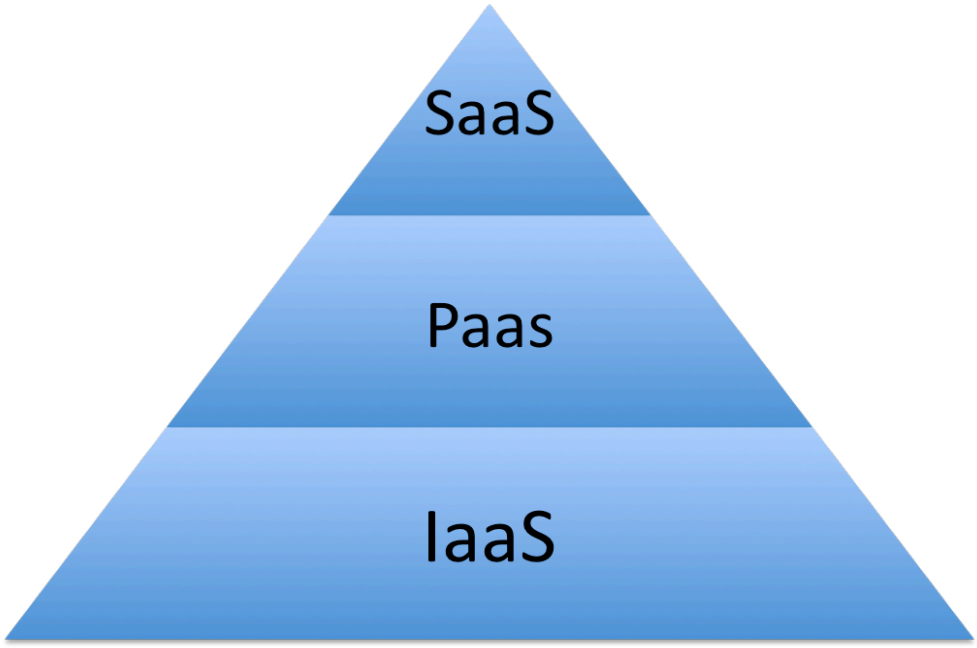


**Figura 5 - Exemplo de empilhamento de diversos tipos de nuvens.**

# Capítulo 3 - Xen

## 3.1 Introdução

O *hypervisor* Xen é desenvolvido colaborativamente pela comunidade Xen.org e engenheiros das principais empresas de soluções de *data centers* como AMD[7], Citrix[8], Dell, Fujitso, HP[9], IBM, Intel[10], NEC, Novell, Red Hat, Samsung, Silicon Graphics, Sun[11], Unisys [12]entre outras. O *hypervisor* Xen é uma solução livre licenciada através da GNU *General Public Licence* (GPL2).

O Xen é um *hypervisor* que permite criar máquinas virtuais paravirtualizadas e de virtualização completa. O *hypervisor* está localizado entre o sistema operacional e o *hardware* físico. Em um ambiente Xen as máquinas virtuais são chamadas de *domains*. Como o sistema de gerenciamento também é considerado uma máquina virtual, máquina virtual privilegiada, é chamado de *Domain* 0 abreviado para *dom0*. Todas as outras máquinas virtuais são chamadas de *DomainU* abreviado para *domU*.

Para máquinas paravirtualizadas não privilegiadas o Xen não exporta um *hardware*, ao menos que seja explicitamente configurado com *pci-passthrough*. Portanto, para o sistema da máquina virtual poder comunicar-se com o *hardware* deve ser através de módulos programados para usar o *hypervisor*, como o *xen_mem*, *xenblk* e *xenfb*. As máquinas virtuais configuradas com *pci-passthrough* recebem acesso físico direto e exclusivo ao dispositivo PCI[13] definido.[3]

Um sistema operacional executando em uma máquina virtual de virtualização completa espera encontrar um ambiente semelhante ao que seria de um computador normal. Para criar esse ambiente de *hardware* emulado o Xen utiliza o QEMU. O termo utilizado no ambiente Xen para definir uma máquina virtual de virtualização completa é *Hardware Virtual Machine* (HVM).

---

[7] AMD – Advanced Micro Devices.
[8] Citrix – Citrix Systems.
[9] HP – Hewlett-Packard.
[10] Intel – Integrated Electronics Corporation.
[11] Sun – Sun Microsystems.
[12] Unisys – Unisys Corporation.
[13] PCI – Peripheral Component Interconnect – É um barramento para conectar dispositivos físicos, como placas de rede, vídeo, som e etc.

## 3.2 Componentes

Um ambiente virtual Xen consiste de diversos componentes, os principais são:

- *Hypervisor* Xen
- *Domain* 0 (*dom0*)
    - *Xend*
    - *Xenstored*
    - *Qemu-dm*
- *DomainU* (*domU*)
    - *Xen Virtual Firmware*

### 3.2.1 Hypervisor

O *hypervisor* Xen é a camada básica de abstração localizada entre o *hardware* e o sistema operacional. É responsável por checar as tabelas de páginas, alocar recursos para novos domínios, e escalonar domínios. O *hypervisor* não apenas abstrai o *hardware* para as máquinas virtuais como também controla a execução das máquinas virtuais pois compartilham o mesmo ambiente de execução. Não contêm nenhum conhecimento de rede, dispositivos de armazenamento externo, vídeo, ou qualquer função de entrada e saída. É responsável também por iniciar a máquina suficiente para o sistema do *dom0*.

Assim como um aplicativo interage com o sistema operacional via chamadas de sistema (*syscalls*), domínios interagem como o *hypervisor* por *hypecalls*. O *hypervisor* responde enviando ao domínio um evento, que exerce a mesma função de uma interrupção de um *hardware* real.[3][4]

### 3.2.2 DomainU

*DomainU's* não tem acesso direto ao *hardware* como o *dom0*, referenciados como domínios não privilegiados. A máquinas virtuais paravirtualizadas executando

no *hypervisor* Xen são referenciadas como hóspedes *DomainU PV* e são sistemas operacionais modificados Linux, Solaris, FreeBSD, ou outro sistema operacional UNIX.

Um máquina virtual *DomainU PV* está ciente que não tem acesso ao *hardware* diretamente. Enquanto uma máquina virtual do tipo de virtualização completa, referenciada como *DomainU* HVM, não está ciente que está em um ambiente compartilhado.

Um hóspede *DomainU* PV contem dois *drivers*, um para acesso a rede e outro para disco, *PV Network Driver* e *PV Block Driver* respectivamente.

Um *DomainU* HVM não contem os *drivers* PV localizados dentro da máquina virtual; ao invés um *daemon*, Qemu-dm é iniciado no *Domain0* para cada hóspede HVM.

O hóspede de um *DomainU* HVM deve inicializar assim como uma máquina física faria, então um *software* é adicionado ao *DomainU* HVM, *Xen Virtual Firmware*, para simular a BIOS por qual um sistema operacional espera ser iniciado.

### 3.2.3 Xen Virtual Firmware

Xen Virtual Firmware é uma BIOS virtual inserida dentro de cada domínio HVM, assegurando que o sistema operacional receba todas as instruções padrões iniciais esperadas durante uma inicialização normal, provendo um ambiente PC-compatível. [3]

### 3.2.4 Domain 0

*Domain* 0 é a primeira máquina virtual a ser iniciada no sistema e tem como função gerenciar os dispositivos do *hardware* e outras máquinas virtuais. É a única máquina virtual executando no *hypervisor* Xen que tem permissões especiais, sendo referenciada como domínio privilegiado (*privileged domain*).

Dois *drivers* são incluídos no *Domain* 0 para suportar requisições de rede e disco de hóspedes o *Network Backend Driver* e o *Block Backend Driver*. O *Network Backend Driver* comunica-se com o *driver* do dispositivo de rede para processar as

requisições dos hóspedes. O *Block Backend Driver* comunica-se com *driver* de disco para ler e escrever dados das requisições do *driver* dos *domU's*. O *dom0* pode conceder acesso direto e exclusivo de um dispositivo a uma outra máquina virtual, esta máquina virtual é descrita como *domain driver* e é considerada como integrante do sistema básico de virtualização.

Por ser a única máquina virtual que pode submeter operações diretamente para o *hypervisor*, o *dom0* é responsável por componentes essenciais para o ambiente virtual:

- Xend
- XenStore
- Qemu-dm

### 3.2.4.1 Xend

O *daemon* Xend é um aplicativo Python que é considerado o gerenciador do *hypervisor* Xen. Todas as requisições feitas ao Xend devem ser feitas através de XML RPC. A ferramenta *xm* é uma interface que traduz comandos do administrador em requisições XML RPC feitas ao Xend.[3][4]

Apesar de todas as operações de gerência de máquinas virtuais passarem pelo Xend, o Xend pode ser parado ou reiniciado sem afetar o funcionamento dos *domains* em execução.

### 3.2.4.3 XenStore

O XenStore é uma base de dados de configurações mantido no *dom0* pelo *daemon* Xenstored. É usado geralmente para controlar dispositivos em hóspedes, *event channels* e áreas de memória compartilhadas. O XenStore é na verdade um arquivo no *dom0*, geralmente localizado em /var/lib/xenstored/tdb. A sua estrutura é hierárquica, e lembra um sistema de arquivos.

O XenStore é dividido em 3 diretórios na sua raiz: /vm , /local/domain e /tool. Os *domains* possuem dois identificadores, uuid e domId. O uuid é um identificador único e é mantido quando o domínio é migrado entre *hosts*. O domId é um identificador

local do *domain* e pode ser modificado quando migrado entre *hosts*. Cada *domain* possui um diretório em /vm contendo informações quanto ao domínio, esse diretório é indexado por uuid. O caminho /local/domain contêm informações quanto a execução do domínio e é indexado por *domId's*. O diretório /tools armazena informações sobre ferramentas.

#### 3.2.4.4 Qemu-dm

Toda máquina virtual *DomainU* HVM requer que tenha o seu próprio *daemon* Qemu-dm. Este *daemon* processa todas as requisições de rede e disco do *Domain* U HVM de uma máquina virtual emulada no ambiente Xen. Qemu-dm fica localizado no *Domain* 0 por precisar de acesso a rede e a dispositivos de entrada e saída.

Uma nova ferramenta, Stub-dm, foi desenvolvida para remover a necessidade de ter um *daemon* Qemu-dm para cada *Domain U* HVM e ao invés irá prover um conjunto de serviços disponíveis para cada *Domain U* HVM. Com essa ferramenta são criados domínios paravirtualizados, referenciados como *Stub Domains,* que auxiliam o *dom0* a executar componentes do Xen para HVM. Este domínio fica responsável por *drivers frontend* de dispositivos de bloco, *frame buffer* e gerenciamento de memória. Este novo modelo retira a responsabilidade do dom0 de processar requisições e melhora a performance dos domínios HVM.[3]

## 3.3 Funcionamento

A função do *hypervisor* é alocar e gerenciar recursos e implementar mecanismos de segurança, provendo um ambiente seguro e integro para os hóspedes. A seguir é mostrado como o *hypervisor* Xen realiza a virtualização dos recursos físicos para as máquinas virtuais.

### 3.3.1 Processador

A arquitetura x86 apresenta diversos desafios para a virtualização de processador, especialmente para hospedes de virtualização completa onde é assumido

que o sistema operacional executa diretamente sobre o *hardware*. A arquitetura x86 possui quatro níveis de privilégio, chamados de anéis de proteção ou *rings*. O anel de proteção mais privilegiado é o 0*,* e o menos privilegiado é o 3. Em um sistema não virtualizado, o *kernel* do sistema operacional executa no anel 0 e os aplicativos de usuários geralmente executam no anel *3*, os anéis 1 e 2 geralmente não são utilizados.[3][4]

Para garantir mecanismos de proteção e isolamento eficientes o *hypervisor* deve executar em um nível de proteção mais privilegiado do que os sistemas hóspedes. Assim o *hypervisor* Xen foi desenvolvido para ser executado no anel 0, e como nenhum sistema operacional x86 utilizou os anéis 1 e 2 desde o OS/2 da IBM, os sistemas operacionais hóspedes puderam ser modificados para executarem no anel 1, mantendo os aplicativos de usuários no anel 3. Executando um sistema hóspede no anel 1 provê isolamento e proteção de outros hóspedes, e assegura que nenhum sistema hóspede executará operações privilegiadas do anel 0. Esta técnica é aplicada em hóspedes paravirtualizados.[3][4]

Na virtualização completa, o sistema operacional hóspede espera ser executado no anel 0, logo técnica acima no pode ser aplicada. A virtualização completa no ambiente Xen só é possível com suporte de *hardware* à virtualização, como as extensões VT-x da Intel e AMD-V da AMD. Essas extensões provêem dois modos de execução: *root* e *non-root.* Ambos modos tem 4 anéis de proteção, dessa forma o *hypervisor* Xen executa no modo *root* acessando o *hardware* físico, enquanto o sistema hóspede da virtualização completa fica no modo *non-root* acessando o *hardware* disponibilizado pelo sistema de emulação de drivers do QEMU.

Uma máquina virtual pode ter até trinta e dois processadores virtuais (vCPU), sendo que os vCPU's são do mesmo modelo do processador real. O número de processadores virtuais não está ligado ao número de processadores físicos sendo definido na criação da máquina virtual. O Xen permite que o número de vCPU's de uma máquina virtual seja alterado em execução. [13]

### 3.3.2 Escalonador

Escalonamento de processador virtual é uma das tarefas criticas do *hypervisor*. Cada processador físico tem uma fila de vCPU's gerenciada por um algoritmo de escalonamento.

Atualmente é possível escolher um entre dois algoritmos de escalonamento no Xen, o SEDF (*Simple Earliest Deadline First*) e o *Credit Scheduler*. O *Credit Scheduler* é o algoritmo padrão na versão atual do Xen (desde a versão 3.0). Os algoritmos de escalonamento do Xen podem ser classificados de acordo com duas categorias:

- *Work-conserving* (*WC-mode*) ou *Non Work-conserving* (*NWC-mode*): No *WC-mode* o processador fica ocioso somente se não existir mais nenhum processo para ser executado. Isto significa que enquanto um processo estiver bloqueado outro processo pode utilizar o processador. No *NWC-mode* cada processo utilizará somente a porção de tempo destinado a ele, e nada mais, assim o processador pode ficar ocioso mesmo tendo processos a serem executados.[12]
- Preemptivo ou Não-preemptivo: Um algoritmo de escalonamento preemptivo verifica a prioridade toda vez que um processo entra no estado de pronto, caso a prioridade do processo em execução for menor do que a do novo processo este será substituído imediatamente pelo novo processo com maior prioridade. Algoritmos não-preemptivos permitem que um processo termine o tempo de execução concedido para depois verificar as prioridades para ser escolhido qual processo vai utilizar o processador.[12]

Os dois algoritmos disponíveis no Xen trabalham em *WC-mode* e *NWC-mode*, mas somente o SEDF é preemptivo, apresentando assim melhor desempenho em operações de entrada e saída do que o *Credit Scheduler*.

No algoritmo *Credit Scheduler*, cada domínio tem dois parâmetros peso (*weight*) e *cap*. O *cap* define a porcentagem máxima de tempo que um domínio pode consumir do processamento, se o valor do *cap* for zero o domínio não tem tempo limite de execução. O valor do *cap* é expressado em relação a um processador físico, sendo 100 equivalente a um processador, 200 equivalente a dois processadores. E o peso define a quantidade de vezes que um domínio utilizará o processador em relação aos outros domínios, assim um domínio com o dobro de peso em relação a outro poderá utilizar o processador o dobro de vezes. O algoritmo *Credit Scheduler* é otimizado para sistemas SMP.[3][11]

O algoritmo SEDF utiliza uma tupla (*s*, *p*, *x*) para cada domínio, assim um domínio tem no mínimo s unidades de tempo em um período *p*, e *x* define se este domínio pode ou não exceder o tempo concedido (*WC-mode* ou *NWC-mode*).[3][11]

### 3.3.3 Memória

O *hypervisor* Xen é responsável por alocar e gerenciar a memória física do *host*, garantindo que nenhum domínio acesse uma área de memória alocada a outro domínio. Cada atualização de página ou tabela de diretórios deve ser validada pelo *hypervisor* garantindo que cada domínio manipule somente as próprias tabelas. A segmentação também deve ser validada pelo *hypervisor*, verificando se segmentos de domínios não ocupam o mesmo espaço de endereçamento ou sejam inválidos.

O Xen reserva uma parte da memória física para o seu próprio uso, o tamanho desse espaço de memória depende da arquitetura do *hardware* subjacente. O restante da memória é partilhado entre o dom0 e os domU's.

Durante a criação, um domínio é alocado a uma parte da memória separada logicamente dos outros domínios. Cada domínio tem uma quantidade atual e máxima de memória alocada. Esses parâmetros podem ser alterados depois da criação do domínio.

O Xen tem um *driver*, chamado de *ballon driver*, que permite que o sistema operacional hóspede possa ajustar a alocação de memória até a quantidade máxima configurada para este. O *ballon driver*, permite que o espaço de memória alocada a um hóspede mas não utilizada possa ser consumido por outros domínios. Com esta dinâmica na alocação de memória, combinada com a criação e o término de máquinas virtuais, a memória é alocada e disponibilizada no nível de paginação. Por causa dessa granularidade o *hypervisor* não pode garantir que um domínio receberá um espaço de memória contiguo.[3][11]

Como os sistemas operacionais da arquitetura x86 esperam um espaço de memória contiguo. Para garantir compatibilidade, o Xen abstrai a paginação física em uma memória pseudo-física baseada por domínio. Assim, mesmo que as páginas físicas de um domínio não sejam contiguas, é apresentado ao sistema operacional hóspede um espaço de memória pseudo-física onde a memória é representada como um espaço continuo de paginação. Para a realizar essa abstração, o *hypervisor*

necessita de duas tabelas. Uma tabela mapeia as páginas físicas em pseudo-físicas, acessada pelo *hypervisor* e todos os domínios. E outra tabela que é fornecida para cada domínio independentemente mapeia páginas pseudo-físicas em páginas físicas. [3]

### 3.3.4 Entrada e Saída

Como o *hypervisor* Xen não foi desenvolvido para dar suporte diretamente a requisições de entrada e saída, todas as requisições devem ser processadas pelo domínio responsável pelo dispositivo, geralmente o *dom0*. Existem duas técnicas de virtualização de operações de entrada e saída.

A primeira consiste na emulação de dispositivos, esta técnica é geralmente empregada em máquinas virtuais HVM e implica em uma grande perda de performance. Nessa técnica é exportado um dispositivo virtual semelhante ao dispositivo físico, e as requisições de entrada e saída devem ser traduzidas para o dispositivo físico. A emulação dos dispositivos é realizada através do QEMU.

A segunda técnica consiste em instalar *drivers* no sistema operacional hóspede que foram projetados para o ambiente virtual, esses *drivers* aproveitam-se de estruturas oferecidas pelo *hypervisor,* essa técnica é referenciada como paravirtualização de entrada e saída. Apesar de geralmente ser aplicada em máquinas virtuais paravirtualizadas, também pode ser utilizada em máquinas virtuais HVM já que não requer na alteração do sistema operacional hóspede, os *drivers* para máquinas virtuais HVM são chamados de GPLPV *drivers*. Para completar a operação, esse modelo requer dois *drivers*, um localizado na maquina virtual hóspede, chamado de *frontend driver,* e outro *driver* localizado no domínio responsável pelo dispositivo referenciado como *backend driver.*

O exemplo apresentado a seguir descreve a requisição de escrita de um dado em disco de um *domU*.

O *driver* de bloco paravirtualizado do *domU* (*frontend driver*) recebe a requisição para escrever no disco local do sistema operacional hóspede. Através do *hypervisor* Xen, escreve o dado no local apropriado da memória que é compartilhado com o *dom0*. Um canal de eventos (*event channel*) existe entre o *dom0* e o *domU* permitindo que comuniquem-se através de interrupções assíncronas. O *dom0* recebe

um interrupção do *hypervisor* Xen fazendo que o *driver* de bloco paravirtualizado localizado no *dom0* (*backend driver*) acesse a memória, lendo os dados apropriados da memória compartilhada com o *domU*. O dado da memória compartilhada é então escrito no disco local.[3][11]

Uma visão simplificada de um *event channel* é um link direto entre o *Domain0* e um *DomainU*. De fato, um *event channel* executa através do *hypervisor* Xen com interrupções especificas registradas no XenStore, permitindo que o dom0 e um domU compartilhem informações rapidamente através da memória local.

Utilizando-se de *event channel's* o *hypervisor* Xen possibilita que dispositivos físicos sejam compartilhados entre domínios paravirtualizados sem a necessidade de um suporte especifico do *hardware*.

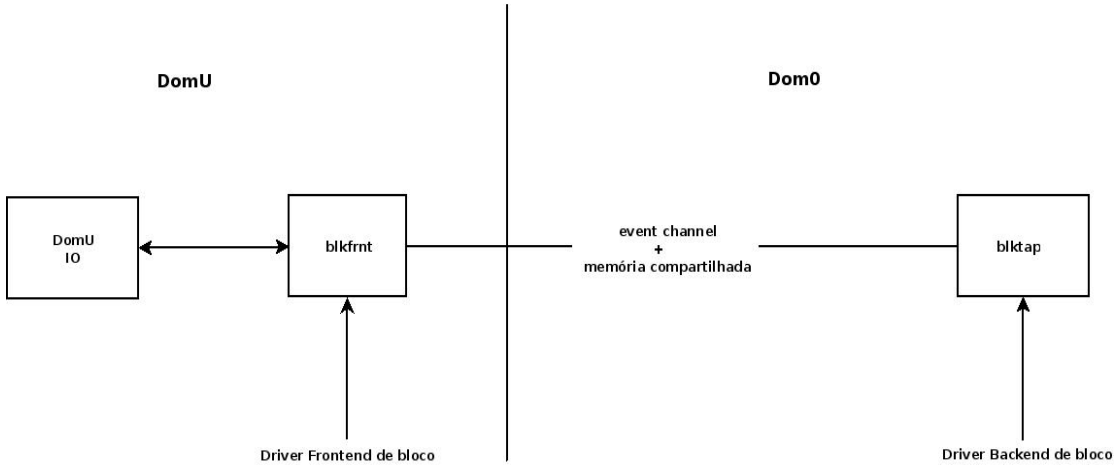


**Figura 6 - Interação entre os componentes responsáveis por realizar uma operação de escrita em disco.**

A figura 6 mostra a interação entre os diversos componentes responsáveis por realizar uma operação de escrita em disco. Na figura o *backend driver* utilizado é o blktap, uma das opções de *drivers* disponíveis para Xen que permite armazenamento de imagens de máquinas virtuais em arquivo.[11]

## 3.4 Onde encontrar ajuda

A comunidade Xen.org possui diversos documentos, disponíveis em seu site http://www.xen.org, sobre administração e configuração do *hypervisor* Xen. A

comunidade Xen.org também possui listas de e-mail (http://lists.xensource.com) de usuário e desenvolvedores do Xen onde podem ser tiradas dúvidas e reportados erros.

A comunidade Xen-br.org é uma comunidade de usuários brasileiros de Xen, seu site wiki.xen-br.org contem informações sobre configuração, administração e ferramentas. As comunidades Xen.org e Xen-br.org possuem canais no servidor IRC irc.freenode.net, ##xen e ##xen-br respectivamente.

Outra fonte de documentação é disponibilizada pelo comando xm, com os parâmetros *create --help_config* exibe todas as variáveis suportadas para configuração de arquivos de criação de máquinas virtuais.

# Capítulo 4 – Tecnologias e Ferramentas

## 4.1 Introdução

Para alcançar todo o potencial de um *cluster* de ambiente virtual é necessário utilizar diversas tecnologias e ferramentas. A escolha de quais tecnologias e ferramentas serão utilizadas depende dos objetivos do projeto em questão. Nesse capítulo são descritas as tecnologias e ferramentas que foram consideradas como opções a serem utilizadas no curso do projeto de virtualização realizado no NTI da UFF. Essas ferramentas e tecnologias podem ser classificadas em três tipos:

- Armazenamento
- Rede
- Gerência

## 4.2 Armazenamento

Ambientes virtuais complexos necessitam de ferramentas para armazenar e disponibilizar as imagens das máquinas virtuais entre os vários *hosts* do *cluster*. Algumas operações de máquinas virtuais, como *Live Migration*, requerem que as imagens dos discos das máquinas virtuais estejam disponíveis entre diversos *hosts*.

### 4.2.1 Network File System

O Network File System (NFS) é um sistema de arquivos compartilhado, onde um computador compartilha um diretório ou uma árvore de diretórios pela rede para outros clientes, essa estrutura é conhecida como *Network-Attached Storage* (NAS). Sua finalidade é disponibilizar um acesso transparente aos clientes ao diretórios compartilhados, como se os dados estivessem armazenados localmente. Sua instalação e configuração é simples e rápida, porém o seu *overhead* é muito alto

tornando não aconselhável em ambientes de grande porte ou que se requeira performance.

É compatível com diversos sistemas como Linux, Microsoft Windows, OSX, e outros, sendo de fácil instalação e configuração. Necessita que aplicativos de suporte sejam instalados tanto no servidor quanto nos clientes.

O NFS possui mecanismo de *lock* de arquivos, garantindo que um arquivo seja usado somente por um dos clientes por vez.

### 4.2.2 Oracle Cluster File System 2

O Oracle Cluster File System (OCFS2) é um sistema de arquivos de *cluster* de disco compartilhado para Linux capaz de prover tanto alta performance quanto alta disponibilidade. É utilizado em ambientes onde um dispositivo de bloco é compartilhado com um grupo de computadores, esse tipo de arquitetura é conhecido como *Storage Area Network* (SAN). Não necessita que aplicativos sejam instalados no equipamento local do disco, podendo assim ser utilizado com *storage apliances*[14]. Como provê um sistema de arquivos de semântica local, é compatível com quase todos aplicativos.

O sistema de arquivos é atualmente usado na solução de virtualização Oracle VM no domínio de gerência, para armazenar imagens de máquinas virtuais e sistema de arquivos possam ser compartilhados.

Em caso de um dos computadores onde o sistema de arquivos está montado perder conexão com os outros computadores do *cluster,* um *daemon* executando localmente, que atua como mecanismo de proteção, reinicia a máquina para evitar o corrompimento do sistema de arquivos. Assim, em um cenário onde todos os hosts estão ligados entre si e ao *storage* por um *switch,* se este ficar inoperante ou sem conectividade todos os *hosts* serão reiniciados. Portanto, um sistema de replicação é vital em um ambiente de produção.

[14] *Storage apliences* – Equipamento para ser utilizado exclusivamente como *storage*.

### 4.2.3 Distributed Replicated Block Device

O Distributed Replicated Block Device (DRBD) é um módulo para o *kernel* do Linux que disponibiliza replicação de dispositivos de bloco através da rede. Portanto, o DRBD não trabalha a nível de sistema de arquivos e sim a nível de dispositivo de bloco.

Cada dispositivo envolvido exerce a função ou de nó primário ou nó secundário. Toda escrita é realizada no nó primário, que depois propaga os dados para os nós restantes. As leituras são sempre realizadas localmente. Em caso do nó primário falhar, um nó secundário receberá a função de primário, e as transferências partirão desse nó.

## 4.3 Rede

O crescimento da complexidade de *clusters* virtuais e a dinamicidade imposta por este tipo de ambiente, requerem que a configuração da estrutura de rede acompanhe as migrações de configurações realizadas na camada da virtualização. Ferramentas como o Open vSwitch e o Cisco Nexus 1000V tem como objetivo endereçar este tipo de problema como também permitir monitoramento de tráfego entre máquinas virtuais no mesmo *host*, incluindo assim a transferência de dados entre máquinas virtuais como parte integrante da infra-estrutura da rede. O Cisco Nexus 1000V não será abordado por não ser *software* livre.

### 4.3.1 Open vSwitch

Open vSwitch (OvS) é um *software switch* distribuído desenvolvido para ser utilizado em ambientes virtuais, compatível com a maioria dos ambientes virtuais baseados em Linux incluindo Xen, XenServer, KVM e QEMU. Está licenciado sob os termos da Apache License versão 2.0, está disponível em http://openvswitch.org.

Residindo no domínio de gerência, o Open vSwitch permite que uma rede seja criada dentro do *host* conectando as máquinas virtuais a rede física. Com atributos encontrados somente em *switches* físicos *high-end,* o OvS permite que configurações

de rede como isolamento, QoS e segurança possam adaptar-se as mudanças impostas pelo ambiente virtual.

Com o avanço da virtualização é comum encontrar quarenta ou mais máquinas virtuais por *host*, nessa escala, administradores precisam de ferramentas para visualizar e administrar as redes virtuais. Com uma posição privilegiada, o OvS pode associar diretamente pacotes de rede com máquinas virtuais e suas configurações. Para manter a configuração de vários *switches* prática, suporta políticas de configuração centralizada, permitindo que administradores gerenciem vários *switches* em *hosts* separados como fosse um *switch* virtual distribuído.

## 4.4 Gerência

As ferramentas de gerência podem ser interfaces web onde permitem o administrador visualizar e realizar algumas operações no *cluster* a ferramentas que gerenciam e tomam decisões sem necessitar a intervenção do administrador. Ferramentas para *clusters* pequenos tendem a requerer mais a intervenção do administrador e serem mais fáceis de serem configuradas, enquanto as destinadas a grandes *clusters* tendem a ser mais autônomas e necessitarem de longos estudos quanto a sua configuração, instalação e utilização. Alguns *softwares*, geralmente *cluster-aware*, disponibilizam demonstrações para administradores ter uma experiência de uso.

### 4.4.1 ConVirt 2.0

O ConVirt 2.0 provê uma interface web de gerência para ambientes de virtualização de *software* livre, suporta Xen e KVM. Permite que o administrador realize operações em máquinas virtuais como iniciar, parar, visualizar console, criar e alterar configurações além de fornecer gráficos de utilização de processador e memória dos *hosts*. Necessita que sejam alterados algumas configurações do *daemon* Xend e instalado *softwares* para permitir que o console seja visualizado.

Nos testes realizados essa ferramenta apresentou instabilidade quando o número de máquinas virtuais ultrapassou cinqüenta.

### 4.4.2 Ganeti

O Ganeti é uma ferramenta de administração de *clusters* desenvolvida pelo Google Inc. e colaboradores sob a licença GNU General Public Licence versão 2. Após instalado em todos os *hosts* todas as tarefas administrativas relativas a máquinas virtuais devem ser realizadas através dele.

Foi desenvolvido para facilitar a gerência de ambientes grandes e prover meios de recuperação rápida de falhas de *hardware*. Permite a administração do *cluster* através de comandos no *host* eleito como *master*, *master node* na nomenclatura do Ganeti, caso este falhe um dos candidatos a *master node* pode assumir seu papel, não tendo um ponto único de falha.

Atualmente está na versão 2.2, conta com suporte aos *hypervisores* Xen e KVM. Suporta armazenamento das máquinas virtuais em arquivo, disco plano, LVM e DRBD, fornecendo automatização de operações como criação, redimensionamento, importação e exportação de discos. Como não suporta estruturas SAN e NAS, muitas operações, como *Live Migration*, só estão disponíveis para máquinas virtuais utilizando modo de armazenamento DRBD. O suporte a *storage* compartilhado está previsto para ser incluído na versão 2.3.

O Ganeti possui *hooks* onde o administrador pode incluir *scripts* que adéqüe a ferramenta as necessidades do ambiente. O administrador também pode criar seus próprios *scripts* para a automatização da instalação do sistema operacional, é fornecido no site do projeto o *script* debootstrap que instala o sistema operacional Debian GNU/Linux no disco da máquina virtual.

O Ganeti possui um conjunto de ferramentas elaboradas para grandes *clusters* chamada de htools. Essas ferramentas permitem: balancear o *cluster* de modo que os recursos sejam utilizados uniformemente; estimar a capacidade de alocação de máquinas virtuais do *cluster*; decidir em qual *host* máquinas virtuais espelhadas deverão ser iniciadas.

### 4.4.3 OpenNebula

O OpenNebula é um projeto de *software* livre desenvolvido com o objetivo de permitir que seja integrado em *clusters* utilizando qualquer tipo infra-estrutura. Suporta atualmente as ferramentas de virtualização Xen, KVM e VMware.

Criado com o objetivo de ser um ponto único de gerência de nuvens do tipo IaaS, permite assim que o *cluster* seja gerenciado através de linha de comando em um *host frontend*, não necessita que seja instalado *softwares* nos *hosts* de máquinas virtuais. Suporta a criação de nuvens públicas, privadas e hibridas podendo ser integrado a recursos disponíveis através do Amazon EC2.[17]

Sua estrutura é modular para permitir a integração com qualquer outra ferramenta ou serviço de virtualização. Está disponível sob a licença Apache versão 2.0.

### 4.4.4 OpenEucalyptus

O OpenEucalyptus implementa nuvens do tipo IaaS privadas, públicas e híbridas. A arquitetura é altamente modular com componentes internos Web *services*, o tornando os fácil de serem expandidos ou trocados. OpenEucalyptus implementa as especificações Amazon EC2, S3 e EBS e é compatível com a API Amazon Web Service (AWS). O objetivo primário é prover um *software* livre que suporte diversas interfaces de nuvens IaaS.

OpenEucalyptus é direcionado para sistemas Linux que utilizem o *hypervisor* Xen.

### 4.4.5 openQRM

O openQRM é uma ferramenta *software* livre de gerência de *data center*. Sua arquitetura é expansível e possui módulos especializados em automatização, monitoramento, virtualização, alta disponibilidade e computação em nuvem. O openQRM é um ponto único de gerência para a infra-estrutura do *data center* e prove uma API bem definida que pode ser integrada com outras ferramentas e *plugins*

adicionais. Provendo um ambiente altamente flexível escalável, openQRM permite integrar diferentes estruturas e tecnologias.

Suporta as principais tecnologias de virtualização como: Xen, KVM, VMware e Linux-VServer. Tarefas como migrar um servidor físico para virtual, virtual para físico, entre diferentes tecnologias de virtualização ou configurar uma máquina para ser monitorada pela ferramenta Nagios podem ser realizadas rapidamente pelo administrador através de *scripts*.

O sistema de alta disponibilidade permite que máquinas de diferentes tecnologias sejam acionadas em caso de falhas. Por exemplo, caso uma máquina física falhe, o openQRM pode iniciar uma máquina virtual para responder pelo serviço.

Atualmente está na versão 4.1, e é disponibilizado imagem para servidores das distribuições Linux Debian, Ubuntu, CentOS e openSuse.

### 4.4.6 Puppet

O Puppet é uma linguagem declarativa para expressar configuração de sistemas. Desenvolvida para expressar os relacionamentos entre servidores, serviços e objetos primários que compõe o serviço. Ao invés de lidar com os detalhes de como alcançar certa configuração ou prover um serviço, usuários de Puppet podem simplesmente expressar a configuração desejada utilizando abstrações que o Puppet é responsável por alcançar a configuração desejada ou prover ao usuário informação suficiente para corrigir os problemas encontrados. Um dos principais objetivos do projeto é permitir que usuário abstraiam os detalhes da configuração e extraiam os relacionamentos das entidades envolvidas.

O Puppet pode ser integrado a outras ferramentas como o openQRM.

## 4.5 Coleções de Ferramentas

Nesse tópico é descrito as funcionalidades de algumas coleções de ferramentas que provem nuvens do tipo IaaS.

### 4.5.1 Xen Cloud Platform

O Xen Cloud Platform (XCP) tem como objetivo prover uma combinação de atributos. O projeto combina a flexibilidade de um ambiente virtualizado Xen com tecnologias de *storage*, segurança e rede. O XCP não utiliza o *daemon* xend para comunicar-se com o *hypervisor* Xen, ao invés utiliza o daemon xapid, todas as requisições feita ao xapid devem ser realizadas através de uma API chamada XAPI. O XCP não suporta a API do xend, assim para que uma ferramenta seja compatível com o XCP deve suportar XAPI.

A versão 0.5 do Xen Cloud Platform contem as seguintes tecnologias e funcionalidades:

- Xen 3.4.2
- Kernel Linux 2.6.27
- Open vSwitch
- Drivers paravirtualizados assinados para Windows
- Modelos de hóspedes, incluindo as últimas versões das distribuições Ubuntu e Debian
- Alerta e monitoramento de performance
- Suporte a recuperação a desastre

### 4.5.2 OpenStack

O OpenStack é uma plataforma para computação em nuvem. O objetivo do projeto é oferecer a comunidade um *software* livre que seja capaz de concorrer com as plataformas proprietárias. O projeto inclui códigos doados pelo RackSpace e está previsto incorporar tecnologias do projeto Nebula Cloud Platform desenvolvido pela NASA. O projeto tem ganhado apoio de diversas empresas, entre elas estão: Intel, Citrix, Riptano, Dell, Cloud.com, AMD, Puppet Labs e Scalr.

OpenStack conta com o suporte aos *hypervisores* Xen, KVM e QEMU. O projeto é distribuído através da licença Apache 2.0.

### 4.5.4 Nimbus

Nimbus é um conjunto de ferramentas *software* livre que juntas provem uma solução de computação em nuvem do tipo IaaS. O objetivo do projeto é endereçar as necessidades de infra-estrutura de projetos científicos. Nimbus permite que clientes requisitem recursos emprestados para alocar máquinas virtuais e configure o ambiente de modo que represente as necessidades do usuário.

O Nimbus possui diversos componentes que podem ser selecionados de diversos modos, os componentes são:

- Workspace Service - Gerenciador de ambiente de máquinas virtuais que pode ser controlado remotamente por diferentes protocolos baseados em HTTP.
- *Frontend* WSRF – Implementação do protocolo WSRF utilizados por Workspace services anteriores e a ferramenta Cloud client.
- Metadata Server – Serviço que responde a requisições HTTP informações especificas de cada máquina virtual utilizando os mesmos caminhos que o EC2 metadata server.
- Cloud client – O seu objetivo é permitir que usuários possam iniciar instancias de máquinas virtuais rapidamente.
- Reference Client - Expõe todas as funções do *frontend* WSRF por linha de comando. É geralmente utilizado por scripts para realizar tarefas especificas.
- Workspace Pilot – Programa que submete tarefas para serem executadas nos *hosts* de máquinas virtuais. Funções de segurança foram implementadas para assegurar que o *host* voltará a hospedar máquinas virtuais quando programado.
- Resource Management API – Módulo que escalona recursos entre usuários.
- Workspace-Control – *Software* instalado em cada host de máquinas virtuais utilizado para realizar operações básicas de máquinas virtuais e disponibilizar informações para contextualização.
- Context Broker – Serviço que disponibiliza aos clientes coordenar grandes clusters virtuais automaticamente. Utilizado para iniciar um conjunto de máquinas virtuais conectadas, também prove a função de personalizar as máquinas virtuais através do script chamado de Context Agent.

- Context Agent – Um agente leve instalado em cada máquina virtual, recolhe informações relevantes ao cluster do Context Broker e realiza mudanças necessárias para adaptar a instancia ao ambiente.
- Nimbus Web – Interface web para o Nimbus com funções administrativas.

# Capítulo 5 – Conclusão

Com a popularização das técnicas de virtualização, a utilização de ferramentas complexas e flexíveis se faz necessário. Surgiu uma categoria de ferramentas criadas com o intuito de disponibilizar um ambiente integrado e autônomo.

Neste trabalho foi estudado o funcionamento do *hypervisor* Xen assim como as funcionalidades de ferramentas que o suportam, principalmente as ferramentas que se propõe a transformar um conjunto de *hosts* de máquinas virtuais em uma nuvem do tipo IaaS. Essas ferramentas são de fundamental importância na virtualização, pois permitem que um ambiente complexo com um grande número de máquinas virtuais seja mantido por um número reduzido de administradores, além de prover sistemas de balanceamento de carga, alta disponibilidade e recuperação de desastres.

Apesar do aumento da complexidade das ferramentas, os *software* livres disponíveis ainda tem as suas funcionalidades pouco documentadas. A falta de padronização dos termos é outro problema que dificulta a compreensão das funcionalidades suportadas pela ferramenta. Infelizmente, com essas lacunas de documentação, a melhor maneira de saber se uma ferramenta adéqua-se a um cenário ainda se resume a uma série de testes.

Algumas ferramentas, como o Ganeti, são muito atreladas a infra-estrutura de armazenamento, dificultando ou impossibilitando o uso em outros cenários.

Por fim, apesar da notável evolução das pesquisas na tecnologia de virtualização, um administrador, com o objetivo de alcançar um ambiente escalável, tem que desprender um grande esforço na pilha de *software* acima do *hypervisor*.

## Trabalhos Futuros

Por ser uma tecnologia nova, existem muitos aspectos que ainda não foram devidamente explorados, alguns deles são:

- Ferramenta para realizar backup de discos de máquinas virtuais sem que haja necessidade de desligamento das mesmas.

- Ferramenta para aprovisionamento de memória sob demanda, permitindo que o administrador defina máximo, mínimo e a porcentagem de memória que deverá ser mantida livre na máquina virtual. Diminuindo a fragmentação da memória disponível entre domínios de memória de máquinas virtuais, consolidando o espaço disponível dentro de um *host* de máquinas virtuais.
- Documentar os problemas e enganos mais comuns na criação de um ambiente utilizando o *hypervisor* Xen.
- Estudar comparações de performance, estabilidade e isolamento entre diversas tecnologias de virtualização com o objetivo de elaborar indicativos de perfis no emprego de cada tecnologia.

# REFERENCIAS BIBLIOGRÁFICAS


[1] Tanenbaum, Andrew S.; WIIDHUL, Albert S., "Operating Systems – Design and Implementation" – 2ª edição, 1997.

[2] OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo, "Sistemas operacionais" – 2ª edição 2001.

[3] Williams, David E.; Garcia, Juan R., "Virtualization with Xen: Including XenEnterprise, XenServer, and XenExpress" – Elsevier, 2007

[4] Matthews, Jeanna N.; Dow, Eli M.; Deshane, Todd; Bongio, Jeremy, "Running Xen: A Hands-On Guide to the Art of Virtualization" – Prentice Hall, 2008

[5] Xen.org, http://www.xen.org/, Acessado em Agosto 2010.

[6] OpenVZ Intro | http://ftp.openvz.org/doc/openvz-intro.pdf , Acessado Agosto 2010.

[7] Diogo Ezídio da Silva; Gilberto Lima de Oliveira; Leandro de Sousa Rangel; Lucas Timm Florão, "Virtualização como alternativa para ambiente de servidores", http://msinhore.xen-br.org/files/TCC-Lucas_Timm_Florao.pdf , Acessado em Agosto 2010.

[8] Kundu, Sajib; Rangaswami, Raju; Dutta, Kaushik; Zhao, Ming, "Application Performance Modeling in a Virtualized Environment" – IEEE, 2009.

[9] Apparao, Padma; Makineni, Srihari; Newell, Don, "Characterization of network processing overheads in Xen", IEEE 2006.

[10] Diwaker Gupta, Ludmila Cherkasova, Rob Gardner, and Amin Vahdat "Enforcing Performance Isolation Across Virtual Machines in Xen" – Universidade da Califórnia, 2007.

[11] Wiki da Comunidade Xen.org | http://wiki.xensource.com , Acessado em Agosto 2010.

[12] Baruchi, Artur; Midorikawa, Edson T**,** "Influência do Algoritmo de Escalonamento Credit Scheduler no Desempenho de Rede", SBC, 2008.

[13] Xianghua Xu; Peipei Shan; Jian Wan; Yucheng Jiang, "Performance Evaluation of the CPU Scheduler in XEN", IEEE, 2008.

[14] Fraser, Keir; Hand, Steven; Neugebauer, Rolf; Pratt, Ian; Warfield, Andrew; Williamson, Mark, “Safe Hardware Access with the Xen Virtual Machine Monitor”, 2006.

[15] Pettit, Justin; Gross, Jesse; Pfaff, Ben; Casado, Martin; Crosby, Simon,“Virtual Switching in an Era of Advanced Edges”, Segundo Workshop on Data Center, 2010.

[16] Wang, Guohui; Andersen, David G.; Kaminsky, Michael; Kozuch, Eugene Ng, Michael; T. S.; Papagiannaki, Konstantina; Glick, Madeleine; Mummert, Lily; “Extending Networking into the Virtualization Layer”, HotNets-VIII, 2009.

[17] Documentação do OpenNebula | http://www.opennebula.org, Acessado em Agosto 2010.

[18] Documentação do Puppet | http://projects.puppetlabs.com/projects/puppet, Acessado em Setembro 2010.

[19] Documentação do Nimbus | http://www.nimbusproject.org/, Acessado em Outubro 2010.

[20] Documentação do OpenStack | http://openstack.org/, Acessado em Outubro 2010.

[21] Documentação do openQRM | http://www.openqrm.com/, Acessado em Outubro 2010.

[22] Documentação do OpenEucalyptus | http://open.eucalyptus.com/, Acessado em Outubro 2010.

[23] Documentação do Ganeti | http://docs.ganeti.org/ganeti/2.2/html/, Acessado em Outubro 2010.

[24] Meisner, David; T. Gold, Brian; F. Wenisch, Thomas; “PowerNap: Eliminating Server Idle Power”, 2009 ACM.

[25] Moore, Justin; Chase, Jeff; Ranganathanf, Parthasarathy; Sharma, Ratnesh; “Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers”, USENIX, 2005