

**UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**SISTEMA DE CONTROLE PARA O
LANÇAMENTO DE SERVIDORES
VIRTUALIZADOS UTILIZANDO XEN**

TRABALHO DE GRADUAÇÃO

Guilherme Linck

Santa Maria, RS, Brasil

2008

SISTEMA DE CONTROLE PARA O LANÇAMENTO DE SERVIDORES VIRTUALIZADOS UTILIZANDO XEN

por

Guilherme Linck

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Benhur de Oliveira Stein

**Trabalho de Graduação N° 247
Santa Maria, RS, Brasil**

2008

**Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação**

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

**SISTEMA DE CONTROLE PARA O LANÇAMENTO DE
SERVIDORES VIRTUALIZADOS UTILIZANDO XEN**

elaborado por

Guilherme Linck

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Dr. Benhur de Oliveira Stein
(Presidente/Orientador)

Prof^a Dr^a Andrea Schwertner Charão (UFSM)

Prof. Antonio Marcos de Oliveira Candia (UFSM)

Santa Maria, 1 de fevereiro de 2008.

AGRADECIMENTOS

Em primeiro lugar eu quero agradecer a meus pais. O apoio, motivação e principalmente cobrança foram muito importantes durante a graduação. Quero agradecer ao meu orientador, professor Dr. Benhur Stein, por ter me escolhido como orientando para a realização deste trabalho de graduação, e cuja ajuda e conselhos foram decisivos para a conclusão deste trabalho. Quero também agradecer ao Diego Kreutz. Os trabalhos que realizamos juntos, seus conselhos e a dedicação para aquilo que faz são um exemplo para mim.

Finalizando, quero agradecer a todos os professores do Curso de Ciência da Computação pela dedicação que tiveram em suas aulas para repassar o seu conhecimento e também a todos que de alguma forma contribuíram para a realização deste trabalho.

“Primeiro eles te ignoram, depois riem de você, então finalmente resolvem te enfrentar e aí você vence” — MAHATMA GHANDI

“A sabedoria da vida não está em fazer aquilo que se gosta, mas gostar daquilo que se faz.” — LEONARDO DA VINCI

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

SISTEMA DE CONTROLE PARA O LANÇAMENTO DE SERVIDORES VIRTUALIZADOS UTILIZANDO XEN

Autor:

Guilherme Linck

Orientador: Prof. Dr. Benhur de Oliveira Stein

Local e data da defesa: Santa Maria, 1 de fevereiro de 2008.

Um grande número de servidores físicos a serem administrados faz com que o trabalho dos administradores seja dispendioso. Além disso o consumo de energia e área ocupada estão se tornando um problema dentro das empresas. Uma das soluções que vem ganhando grande aceitação na resolução deste problema envolve o uso de virtualização de serviços.

Neste contexto encontra-se o monitor de máquinas virtuais Xen, o qual permite que diversos sistemas operacionais executem de forma independente, compartilhando os recursos de *hardware* de um único servidor. Entretanto, durante o lançamento das máquinas virtuais Xen não faz uma avaliação de quais máquinas virtuais devem ser lançadas primeiro, ou seja, não há uma verificação das dependências entre elas. Isso acaba por gerar diversos problemas, pois comumente um servidor oferece serviços que são necessários para a correta inicialização de outros serviços executados em outras máquinas virtuais. O monitoramento dos serviços executados por estas máquinas virtuais também é importante, pois uma falha em algum dos serviços vai repercutir em problemas nos serviços que dele possam necessitar. O desligamento de máquinas virtuais em uma ordem segura garante que problemas sejam evitados caso uma das máquinas virtuais necessite de algum serviço já interrompido.

Este trabalho tem por objetivo solucionar estes problemas. As estratégias utilizadas na implementação da solução são descritas neste documento, juntamente com os resultados finais encontrados.

Palavras-chave: Virtualização, xen, máquina virtual, gerenciamento de redes.

LISTA DE FIGURAS

Figura 3.1 – Arquitetura Xen	21
Figura 4.1 – Arquitetura do <i>LauXen</i>	26
Figura 4.2 – Exemplo de Ciclo Encontrado em Arquivo de Configuração.	28
Figura 4.3 – Visão Geral dos Monitores em Funcionamento	29
Figura 4.4 – Grafo Correspondente aos Dados da Estrutura de Controle.....	29
Figura 4.5 – Visão Geral da Estrutura Interna de um Monitor	31
Figura 4.6 – Visão Geral do Sistema de Desligamento das MVs.	32
Figura 4.7 – Arquivo de Configuração Utilizado Para o Lançamento das MVs.....	34
Figura 4.8 – Arquivo de Configuração Utilizado Durante o Desligamento das MVs.	35
Figura 4.9 – Arquivo de Configuração do Cenário A	36
Figura 4.10 – Grafo Correspondente ao Cenário A.	36
Figura 4.11 – Monitores em Funcionamento no Cenário A.	38
Figura 4.12 – Monitor de Portas em Funcionamento no Cenário A.....	38
Figura 4.13 – MVs em Funcionamento no Cenário A.	39
Figura 4.14 – Grafo que Representa o Ciclo Gerado.....	39
Figura 4.15 – Alerta Gerado pela Rotina de Validação.....	39
Figura 4.16 – Arquivo de Configuração do Cenário B.....	40
Figura 4.17 – Grafo Correspondente ao Cenário B.	40
Figura 4.18 – Monitores em Funcionamento no Cenário B.	41
Figura 4.19 – Monitor de Portas em Funcionamento no Cenário B.	42
Figura 4.20 – MVs em Funcionamento no Cenário B.	42

LISTA DE TABELAS

Tabela 4.1 – Tabela com os tempos de inicialização/desligamento das MVs do Cenário A	37
Tabela 4.2 – Tabela de validação do monitoramento do Cenário A.....	37
Tabela 4.3 – Tabela com os tempos de inicialização/desligamento das MVs do Cenário B	41
Tabela 4.4 – Tabela de validação do monitoramento do Cenário B.....	41

LISTA DE ABREVIATURAS E SIGLAS

AMD	Advanced Micro Devices.
CPU	Central Processing Unit.
DHCP	Dynamic Host Configuration Protocol.
DDR	Double Data Rate.
DNS	Domain Name System.
LDAP	Lightweight Directory Access Protocol.
GB	Gigabyte.
GHz	Gigahertz.
GNU	GNU is Not Unix.
GPL	GNU General Public License.
HD	Hard Disk.
IBM	International Business Machines.
LauXen	Launcher Xen.
LGPL	GNU Lesser General Public License.
LMM	LauXen Monitor Manager.
MB	Megabyte.
MHz	Megahertz.
MIT	Massachusetts Institute of Technology.
MMV	Monitor de Máquina Virtual.
MV	Máquina Virtual.
NFS	Network File System.
PC	Personal Computer.
RAM	Random Access Memory.
RPM	Rotações Por Minuto.
SDRAM	Synchronous Dynamic Random Access Memory
SSH	Secure Shell.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

SUMÁRIO

1	INTRODUÇÃO	13
2	VIRTUALIZAÇÃO DE SERVIÇOS E RECURSOS COMPUTACIONAIS ..	16
2.1	Histórico	16
2.2	Benefícios	17
2.3	Monitores de Máquinas Virtuais	18
2.3.1	Virtualização Completa	18
2.3.2	Paravirtualização	18
3	ADMINISTRAÇÃO DE AMBIENTES VIRTUALIZADOS	20
3.1	O Monitor de Máquinas Virtuais Xen	20
3.1.1	Administrando o Ambiente	21
3.1.2	Arquivos de Configuração	21
3.2	Ferramentas para Administração de Ambientes Virtualizados em Xen ..	22
3.2.1	Enomalism	22
3.2.2	XenMan	23
3.2.3	Virt-Manager	23
3.2.4	AdXen	23
4	FERRAMENTA DESENVOLVIDA: LAUXEN	25
4.1	Arquitetura	26
4.2	Decisões de Implementação	27
4.3	Funcionamento	27
4.3.1	Análise do Arquivo de Configuração	27
4.3.2	Lançamento das Máquinas Virtuais	28
4.3.3	Monitoramento dos Serviços das Máquinas Virtuais	30
4.3.4	Desligamento das Máquinas Virtuais	31
4.3.5	Gerenciamento de Memória	32
4.3.6	<i>LauXen</i> : Outras Funcionalidades	33
4.4	Arquivos de Configuração	34
4.5	Avaliação	35
4.5.1	Metodologia	35
4.6	Cenário A	36
4.7	Cenário B	39
5	CONCLUSÃO E PRÓXIMAS ATIVIDADES	43
	REFERÊNCIAS	45

APÊNDICE A	INSTALAÇÃO E UTILIZAÇÃO DE <i>LAUXEN</i>	49
A.1	Pré-requisitos	49
A.2	Instalação e Configuração	49

1 INTRODUÇÃO

Devido à redução de custos de *hardware* e o surgimento de modernos sistemas operacionais multitarefa nas décadas de 80 e 90, *mainframes* começaram a ser substituídos por computadores menores e mais baratos (ROSENBLUM M.; GARFINKEL, 2005). Isso possibilitou que computadores e servidores se popularizassem entre as empresas, haja vista que somente universidades e centros de pesquisas podiam arcar com os custos de um *mainframe*.

Entretanto, para minimizar os problemas gerados pela falha de um servidor, adotou-se o modelo de um servidor por aplicação, aumentando os custos de *hardware* e gerenciamento (ROSENBLUM M.; GARFINKEL, 2005; XENSOURCE, 2005). Este modelo originou vários problemas, dentre eles um maior consumo de energia elétrica, falta de espaço físico para ampliação do número de servidores, baixa utilização da capacidade computacional e necessidade de aperfeiçoamentos no sistema de refrigeração (GOTH, 2007). Por outro lado, este modelo evitava que a paralisação de uma única máquina interrompesse vários serviços.

Uma das soluções que contribuíram para minimizar todos estes problemas envolve o uso de virtualização. Sua primeira aparição data dos anos 60 (CHEN; NOBLE, 2001), onde o projeto M44/44X, realizado através de uma parceria entre a IBM e o MIT (VAUGHAN-NICHOLS, 2006), resultou em um sistema que possuía a capacidade de simular várias máquinas virtuais IBM 7044 (M44) rodando sobre o mesmo *hardware* (SINGH, 2004). O conhecimento obtido através deste projeto foi empregado no sistema operacional IBM VM/370 (VAUGHAN-NICHOLS, 2006) para melhorar a utilização da capacidade computacional dos *mainframes* IBM (GOLDBERG, 1974), oferecendo um ambiente exclusivo a cada usuário (GOLDBERG, 1973). Atualmente virtualização está se tornando uma tecnologia essencial em *datacenters*, pois dentre os benefícios encontram-se: isolamento

de falhas, isolamento da segurança e de ambiente (YOUNGGYUN et al., 2007). Além disso, com o uso de virtualização é possível executar várias máquinas virtuais de forma isolada em um único servidor físico. Cada máquina virtual pode executar seu próprio sistema operacional, cada uma executando diferentes aplicações e serviços de forma concorrente sobre o *hardware* adjacente. Isto é conhecido como consolidação de servidores (LAUREANO; MAZIERO; JAMHOUR, 2004). Esta técnica ameniza os problemas ocasionados pelo excesso de servidores, além de permitir um melhor aproveitamento dos recursos computacionais dos servidores, haja vista que o poder computacional presente nos computadores atuais não é completamente utilizado na maior parte do tempo (WANG et al., 2007). Existe uma camada de *software* posicionada acima do *hardware*, chamada Monitor de Máquina Virtual (MMV), cuja função é gerenciar as múltiplas máquinas virtuais em execução. Basicamente, esta camada de *software* realiza o compartilhamento, gerenciamento e alocação de recursos para cada máquina virtual, tornando-se o componente mais importante em sistemas que utilizam virtualização (ZHAO; BORDERS; PRAKASH;, 2004).

Neste contexto encontra-se Xen (BARHAM et al., 2003), um monitor de máquinas virtuais de código aberto lançado em 2003 (BARHAM et al., 2003), cujo suporte estende-se a diversas arquiteturas de processadores, dentre elas x86, x86_64, IA64 e Power 5 (XENSOURCE, 2005). Xen possui a capacidade de hospedar máquinas virtuais executando uma boa variedade de sistemas operacionais, tais como Windows XP, distribuições Linux, Solaris, FreeBSD e NetBSD. Uma funcionalidade importante oferecida por Xen é a *Live Migration*, tornando possível a relocação de máquinas virtuais entre diferentes servidores interconectados sem a necessidade de interromper os serviços que nelas executam.

Existem diversas ferramentas que auxiliam no gerenciamento de ambientes virtualizados, muitas delas com interfaces gráficas amigáveis. Contudo, ainda não existem ferramentas que auxiliem na ativação ordenada das máquinas virtuais. Uma ferramenta capaz de realizar esta tarefa é essencial em situações onde a correta inicialização dos serviços de uma máquina virtual está intimamente ligada a serviços de outras máquinas virtuais. Sendo assim, um dos objetivos do presente trabalho é apresentar uma ferramenta que resolva esta limitação. Outro objetivo é monitorar a hierarquia de dependências entre as máquinas virtuais, tomando as medidas necessárias para evitar problemas caso uma das

dependências apresente uma falha. O desligamento ordenado das máquinas virtuais tendo como critério as suas dependências também é realizado pela ferramenta desenvolvida. Por último, através desta ferramenta é permitido um gerenciamento flexível da memória destinada às máquinas virtuais.

O próximo capítulo aborda assuntos relacionados à virtualização de serviços. Em seguida, é apresentado o MMV Xen e algumas ferramentas que auxiliam na administração de arquiteturas virtualizadas. Na seqüência, é apresentada a ferramenta desenvolvida e suas funcionalidades, incluindo também uma série de testes para avaliação da ferramenta. As atividades que darão continuidade ao desenvolvimento da ferramenta são discutidas em seguida. Finalizando, é apresentada a conclusão e um apêndice com informações sobre a instalação, configuração e utilização da ferramenta apresentada.

2 VIRTUALIZAÇÃO DE SERVIÇOS E RECURSOS COMPUTACIONAIS

Este capítulo apresenta uma introdução ao conceito de virtualização de serviços e recursos computacionais. São abordados os benefícios de sua utilização e que justificam o porquê desta abordagem vir ganhando cada vez mais adeptos. Os tipos de virtualização utilizados em Monitores de Máquinas Virtuais são discutidos e concluem este capítulo.

2.1 Histórico

Conforme citado no capítulo anterior, o conceito de virtualização surgiu na década de 60 através de uma parceria entre a IBM e o *Massachusetts Institute of Technology* (MIT) (SINGH, 2004) como forma de compartilhar por tempo a utilização de *mainframes* (ROSE, 2004). Devido a isso, virtualização foi largamente empregada nos *mainframes* até a década 90, melhorando significativamente a utilização de seus recursos (GOLDBERG, 1974), quando passou a perder importância devido a uma série de motivos. Dentre eles podemos citar: redução do custo de *hardware*, aparecimento de sistemas operacionais multitarefa, substituição dos *mainframes* por minicomputadores e o surgimento dos primeiros PCs. O *hardware* destes computadores não permitia o uso de virtualização de forma eficiente, contribuindo ainda mais para o abandono desta técnica (ROSENBLUM M.; GARFINKEL, 2005).

Ironicamente, os motivos que levaram ao abandono do uso de plataformas virtualizadas no final dos anos 80 foram os principais responsáveis pelo seu retorno (ROSENBLUM M.; GARFINKEL, 2005). O aumento do número de servidores devido à redução dos custos de *hardware* permitiu que os administradores alocassem um servidor por aplicação, resolvendo o problema da paralização de diversos serviços devido a falha de um único servidor (ROSENBLUM M.; GARFINKEL, 2005; XENSOURCE, 2005). Esta

prática levou a adoção de um grande número de servidores, repercutindo em uma maior área física necessária para sua acomodação, sobrecarga do administrador devido ao gerenciamento dos múltiplos serviços localizados em diferentes máquinas, problemas de escalabilidade e também em aumentos de investimentos em fornecimento de energia e soluções de refrigeração, bem como a baixa utilização do poder computacional destes servidores (GOTH, 2007).

2.2 Benefícios

O uso de servidores hospedando sistemas operacionais virtualizados reduz significativamente a quantidade de servidores empregados (APPARAO; MAKINENI; NEWELL, 2006). Isto se deve ao fato de que a execução de várias máquinas virtuais sobre o mesmo *hardware* leva a uma melhor utilização do *hardware* disponível (YOUNGGYUN et al., 2007; SMITH; NAIR, 2005). Como consequência, a área ocupada pelos servidores e os gastos com energia, infraestrutura e refrigeração diminuem.

Com o uso de virtualização o gerenciamento se torna mais fácil. Características como a *Live Migration* permitem que os administradores migrem servidores virtuais para outros servidores físicos sem a necessidade de interrupção do serviço. Balanceamento de carga, escalabilidade e recuperação de desastres, quer seja por problemas de *hardware* ou comprometimento da integridade do sistema, fazem grande uso desta funcionalidade para aumentar a disponibilidade e desempenho dos serviços (QUYNH; TAKEFUJI, 2006; VAUGHAN-NICHOLS, 2006; ROSENBLUM M.; GARFINKEL, 2005).

A área de segurança também é beneficiada com a sua adoção. Como virtualização permite isolamento entre as máquinas virtuais hospedadas e o encapsulamento do seu estado atual (SMITH; NAIR, 2005), o comprometimento de uma das máquinas virtuais ou a realização de qualquer tipo de atividade maliciosa não prejudicará as outras máquinas virtuais ou o sistema por completo (YOUNGGYUN et al., 2007).

A configuração de novos servidores torna-se uma tarefa menos dispendiosa para os administradores. Ao invés de realizar a instalação completa de um novo servidor físico, basta utilizar a imagem de um *template* de servidor virtual previamente criado, contendo um sistema base e com configurações comuns a todos os outros servidores, instalar as aplicações que nele executarão, transferi-la via rede até a máquina destino ou simplesmente ativá-la no servidor em que esteja.

2.3 Monitores de Máquinas Virtuais

Basicamente, um Monitor de Máquina Virtual (MMV) nada mais é do que uma camada de *software* que executa sobre o *hardware* de um computador (ZHAO; BORDERS; PRAKASH;, 2004), sendo responsável por fazer a abstração dos recursos oferecidos por uma máquina física às máquinas virtuais em execução (ROSE, 2004), fazendo com que estas acreditem estar executando sobre um *hardware* real. Além disso, o MMV (também chamado de *hypervisor*) faz todo o processo de gerenciamento dos recursos computacionais presentes no servidor hospedeiro à disposição das máquinas virtuais (VAUGHAN-NICHOLS, 2006), e ainda o isolamento lógico entre elas, impedindo que uma acesse os recursos de outra (ZHAO; BORDERS; PRAKASH;, 2004).

MMVs podem utilizar duas abordagens de virtualização: virtualização completa e paravirtualização, cujas peculiaridades veremos a seguir.

2.3.1 Virtualização Completa

MMV que pertencem a esta categoria fornecem uma réplica virtual do *hardware* do sistema às máquinas virtuais. Deste modo, sistemas operacionais e aplicativos podem executar sem a necessidade de qualquer modificação, pois os mesmos acreditam estar executando sobre um *hardware* real.

MMVs pertencentes a esta categoria executam sobre um sistema operacional hospedeiro, possuindo um baixo nível de privilégios de execução. Como consequência há degradação de desempenho (ROSE, 2004; MENASCE, 2005), pois o acesso ao *hardware* tem um alto custo e envolve tradução binária em tempo de execução para ser realizado. Um dos mais conhecidos MMVs a usar esta metodologia é o VMWare (SUGERMAN; VENKITACHALAM; LIM, 2001).

2.3.2 Paravirtualização

Nesta abordagem há a necessidade de modificação do sistema operacional para que novas intruções de virtualização sejam utilizadas e o mesmo execute mais próximo do MMV (APPARAO; MAKINENI; NEWELL, 2006; MENASCE, 2005). Isto permite um nível maior de privilégios de execução a rotinas de acesso ao *hardware*. Nesta abordagem o sistema operacional está ciente de que executa em um ambiente virtualizado, permitindo que aprimorações sejam feitas, levando a um ganho de desempenho.

Paravirtualização, também conhecida como *Lightweight Virtualization* (VAUGHAN-NICHOLS, 2006), é 100% compatível com binários das MVs (ROSE, 2004), consome menos recursos computacionais e possui um desempenho superior à Virtualização Completa, pois não há consumo de CPU com a emulação completa de *hardware*. Xen (BARHAM et al., 2003) utiliza esta abordagem de virtualização em *hardware* que não oferece suporte à virtualização, do contrário a estratégia de virtualização completa pode ser utilizada.

3 ADMINISTRAÇÃO DE AMBIENTES VIRTUALIZADOS

A administração de ambientes virtualizados requer a execução de várias tarefas. Neste capítulo serão abordadas as tarefas básicas na administração de ambientes virtualizados utilizando o MMV Xen. Embora o emprego de virtualização ofereça vários benefícios (seção ??), ambientes onde existam um grande número de máquinas virtuais a serem administradas podem levar a uma sobrecarga dos profissionais responsáveis. Este foi o motivo pelo qual pesquisas foram feitas e resultaram na criação de ferramentas com interfaces amigáveis para auxiliar nesta tarefa. Algumas destas ferramentas são brevemente abordadas e concluem este capítulo.

3.1 O Monitor de Máquinas Virtuais Xen

Xen (BARHAM et al., 2003) é um Monitor de Máquina Virtual que pode operar nas duas estratégias (APPARAO; MAKINENI; NEWELL, 2006), virtualização completa e paravirtualização (seções 2.3.1 e 2.3.2 respectivamente). Desenvolvido pela Universidade de Cambridge e com sua primeira versão publicada no ano de 2003 sob licença LGPL (*GNU Lesser General Public License*), hoje oferece suporte a processadores com arquiteturas x86, x86_64, IA64 e Power 5 (XENSOURCE, 2005). Tratando-se de sistemas operacionais, Xen possui a capacidade de hospedar Windows XP, distribuições Linux, Solaris, FreeBSD e NetBSD nas máquinas virtuais em execução. A figura 3.1 mostra o posicionamento de Xen sobre o *hardware*.

Acima do *hardware* encontra-se Xen (*hypervisor*), responsável por fazer a abstração do *hardware* para as máquinas virtuais. Contudo, o *hypervisor* não executa rotinas de acesso ao *hardware*, simplesmente oferece operações básicas de controle, que são expostas através de uma interface acessível aos domínios hospedados (DomUs).

Um domínio especial (Dom0) é criado durante a inicialização do sistema e tem acesso

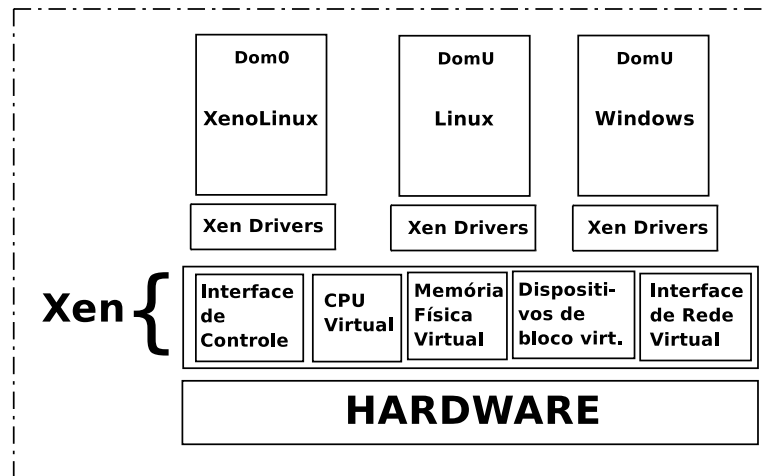


Figura 3.1: Arquitetura Xen

à interface de controle. Esta interface oferece funções que fazem todo o gerenciamento dos recursos de *hardware* e controle dos domínios hospedados, livrando o MMV desta função, simplificando-o e com isso aumentando o desempenho, pois todas as tarefas administrativas são executadas por este domínio.

3.1.1 Administrando o Ambiente

A administração das máquinas virtuais consiste basicamente na execução de diversos comandos no servidor hospedeiro. Para isso, Xen contém uma lista de rotinas que fornecem os meios necessários para a administração das máquinas virtuais hospedadas. Algumas das mais importantes são:

- *xm create* [Nome da MV]: Lança a máquina virtual;
- *xm shutdown* [Nome da MV]: Desliga a máquina virtual;
- *xm destroy* [Nome da MV]: Força a destruição da máquina virtual em caso do comando acima não surtir efeito;
- *xm migrate* [Nome da MV] [Nome do Servidor Destino]: Faz a migração em tempo real da máquina virtual para o servidor passado, sem paralização do serviço;
- *xm console* [Nome da MV]: Dá acesso ao console da máquina virtual;

3.1.2 Arquivos de Configuração

Todos os arquivos de configuração utilizados por Xen encontram-se no diretório `"/etc/xen/"`. Cada máquina virtual possui seu próprio arquivo de configuração, contendo, entre outros

parâmetros, as seguintes linhas:

- *kernel*: Localização do *kernel* a ser carregado;
- *memory*: Quantidade de memória reservada para esta MV;
- *disk*: Localização da imagem de disco contendo a MV;
- *vif*: Parâmetros de configuração da interface de rede;

Maiores informações sobre os arquivos de configuração e rotinas administrativas podem ser obtidas no manual do Xen (CAMBRIDGE COMPUTER LABORATORY, 2006).

A tarefa de administração torna-se dispendiosa em grandes ambientes, levando à sobrecarga dos administradores. Baseando-se nisso, pesquisas foram realizadas e resultaram em ferramentas que facilitam o trabalho dos administradores, algumas delas serão vistas a seguir.

3.2 Ferramentas para Administração de Ambientes Virtualizados em Xen

Esta seção contém uma breve descrição de algumas ferramentas que auxiliam na administração de ambientes virtualizados.

3.2.1 Enomalism

Enomalism (ENOMALY, 2005) oferece uma interface Web para o gerenciamento de ambientes virtualizados. Entre as funcionalidades existentes destacam-se: suporte a Linux e Windows, gerenciamento de recursos em tempo real, criação, ativação, monitoração, backup, destruição e migração de máquinas virtuais a partir de qualquer computador com acesso à Internet ou à rede do servidor onde executa.

Por oferecer a capacidade de administração remota destes ambientes, *Enomalism* possui um mecanismo de autenticação segura baseado em LDAP. Além disso, todo o funcionamento da interface está fortemente a ele ligado. Há uma versão livre distribuída sob a licença LGPL (*GNU Lesser General Public License*), bem como uma versão comercial que oferece mais recursos.

3.2.2 XenMan

XenMan(HAZARD; JEDI, 2006), renomeado como ConVirt há pouco tempo, oferece um painel onde é possível gerenciar e visualizar dados de todos os servidores virtuais baseados em Xen conhecidos, locais ou não. Dentre estes dados encontram-se: nome da máquina virtual, histórico de atividades, utilização de CPU e memória. Todos estes dados são coletados através de um canal de comunicação seguro utilizando SSH (*Secure Shell*), utilizado inclusive na realização das atividades de gerenciamento dos servidores remotos.

XenMan permite a realização de uma série de tarefas administrativas, como: fácil criação de uma nova MV, alterar arquivos de configuração e salvar o estado de uma MV existente, iniciar, desligar e destruir todas as máquinas virtuais de um determinado servidor. Servidores remotos podem ser adicionados ao painel, centralizando a administração de diversas máquinas virtuais localizadas em diferentes servidores em um único local. Sua distribuição é livre e sob a licença GPL (*GNU General Public License*).

3.2.3 Virt-Manager

Virt-Manager (*Virtual Machine Manager*) (VIRT-MANAGER: VIRTUAL MACHINE MANAGER, 2006) é uma interface gráfica que foi criada pela empresa desenvolvedora da distribuição Linux Redhat. Através desta interface é possível ver um sumário das máquinas virtuais ativas, contendo estatísticas da utilização de recursos, além de gráficos de desempenho em função do tempo. Outras funções oferecidas por esta ferramenta são: criação, configuração e ajustes na alocação de recursos para as MVs. Ao contrário de outras ferramentas, Virt-Manager utiliza uma biblioteca de virtualização chamada LibVirt (LibVirt - VIRTUALIZATION API, 2006) para abstrair o MMV.

Inicialmente esta ferramenta oferecia suporte apenas ao MMV Xen. Contudo, novas versões da LibVirt levaram a aperfeiçoamentos em Virt-Manager, que passou a oferecer suporte a MMVs independentemente de sua tecnologia.

3.2.4 AdXen

Adxen(KOSLOVSKI; BOUFLEUR; CHARÃO, 2007) é uma ferramenta gráfica para administração de ambientes virtualizados baseados em Xen. Ela oferece suporte às ações básicas para administração destes ambientes, como: criação, destruição, pausa, retomada e migração de máquinas virtuais.

Uma característica interessante desta ferramenta é a capacidade de dividir diversas máquinas virtuais em grupos. Isto permite que uma determinada ação seja aplicada a todos os membros de um determinado grupo, agilizando atividades que antes eram feitas individualmente. Ao contrário de XenMan (seção 3.2.2) onde cada máquina virtual deve ser manualmente cadastrada para que se possa centralizar a administração, Adxen possui uma funcionalidade responsável por fazer a descoberta automática de servidores virtualizados, o que é indiscutivelmente útil em grandes ambientes.

4 FERRAMENTA DESENVOLVIDA: *LAUXEN*

Dentre as ferramentas de gerenciamento de máquinas virtuais Xen existentes, não é de nosso conhecimento a existência de alguma que realize o ordenamento do lançamento de máquinas virtuais. Esta funcionalidade é particularmente útil em grandes ambientes virtualizados, pois não são raras as situações onde a interação entre diversos serviços localizados em diferentes máquinas virtuais aconteça. Considerando ainda que Xen suporta até 100 máquinas virtuais em execução simultânea (BARHAM et al., 2003), a necessidade de uma ferramenta que faça o lançamento ordenado com base nas dependências existentes entre as diversas máquinas virtuais se faz necessária. Esta situação pode ser ilustrada com o seguinte exemplo: 2 máquinas virtuais hospedadas no mesmo servidor, a primeira executando um servidor WEB, cuja interface de rede obtém a configuração através de um servidor DHCP que está hospedado na segunda máquina virtual. O servidor DHCP deve ser lançado primeiro e enquanto este serviço não estiver disponível, o servidor WEB não pode ser lançado, do contrário sua interface de rede não será configurada.

Além disso, o desligamento em uma ordem segura previne eventuais problemas causados por quebras de dependências ocorridas nesta etapa durante o próximo lançamento. Uma situação que expõe isso é a existência de um servidor que exporta um diretório para outras máquinas virtuais. Se estas máquinas necessitarem de acesso ao diretório importado para que seus serviços possam ser corretamente desligados, caso a máquina virtual que os exporta já tenha sido desligada poderá haver perdas de dados necessários para uma futura inicialização. A ferramenta desenvolvida realiza este processo com segurança, seguindo a hierarquia de dependências entre as máquinas virtuais em execução.

No entanto, estas dependências não são somente importantes durante o lançamento e desligamento das máquinas virtuais, mas também durante a execução dos seus serviços. Voltando ao exemplo exposto no primeiro parágrafo deste capítulo, onde o servidor

DHCP deve ser inicializado antes do servidor WEB, caso o primeiro pare de funcionar em um determinado momento, o servidor WEB pode ficar sem rede antes que o administrador possa restabelecer este serviço. Para evitar problemas como este, a ferramenta desenvolvida monitora o funcionamento dos serviços das máquinas virtuais e age de forma pró-ativa na manutenção destas dependências, prevenindo falhas nos serviços que possam ser afetados por problemas em suas dependências.

Durante a execução de uma máquina virtual, situações onde a sua quantidade de memória inicialmente concedida ser insuficiente podem acontecer. A ferramenta desenvolvida permite ao administrador alterar esta quantidade de memória em tempo de execução, auxiliando na prevenção de problemas por falta de memória.

As próximas seções descrevem a arquitetura da ferramenta *LauXen*¹, seu funcionamento e uma avaliação que comprova o funcionamento da ferramenta.

4.1 Arquitetura

A figura 4.1 exibe a estrutura básica do *LauXen*. A arquitetura é formada basicamente por 4 componentes: *Arquivo de Configuração*, *Lançador*, *Monitor de Portas* e *Monitores*. Os números de 1 a 4 servem para indicar o funcionamento da arquitetura. Inicialmente, o *Lançador* acessa (1) o *Arquivo de Configuração* e realiza uma série de análises para garantir o correto funcionamento de *LauXen*. Concluída esta etapa, o *Lançador* lança (2) tantos *Monitores* quantas forem as máquinas virtuais a serem lançadas, declaradas no *Arquivo de Configuração*, lança (3) o *Monitor de Portas* e encerrando o seu papel dentro do sistema.

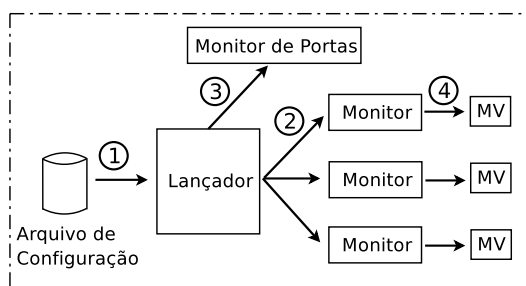


Figura 4.1: Arquitetura do *LauXen*.

Ao ser disparado pelo *Lançador*, um *Monitor* é responsável por decidir o momento em que o lançamento da máquina virtual que monitora é permitido, realizando-o. Após

¹Lau do inglês *Launcher* (*lançador*) e Xen como nome do MMV escolhido.

disparar (4) o lançamento de sua máquina virtual, o *Monitor* inicia o processo de monitoramento dos serviços que esta máquina virtual executa, além dos serviços que esta necessita e que estão em execução em outras máquinas virtuais.

4.2 Decisões de Implementação

A arquitetura proposta busca tornar seu uso o mais simples possível. O arquivo de configuração pode ser facilmente preenchido pelo administrador da rede. Além disso, *LauXen* é inicializado por um *script init*² durante o boot do sistema.

LauXen foi implementado de forma a executar sobre Dom0. O motivo pelo qual *LauXen* executa em Dom0 tem como principal argumento a centralização do cadastro das máquinas virtuais a serem lançadas. Outra forma de realizar o controle de lançamento dos servidores virtualizados seria inserir meios de controle nas próprias máquinas virtuais.

Entretanto, esta abordagem envolveria uma maior participação do administrador, pois o mesmo deve copiar estes mecanismos para cada máquina virtual e configurar o sistema para executá-los. Pior do que isso, estes mecanismos devem ser executados imediatamente após a configuração da rede da máquina virtual, pois sem conexão de rede Nmap não consegue realizar o seu trabalho. Isto acaba por limitar a configuração da rede de forma estática em cada máquina virtual, sem utilizar ferramentas que automatizam isso, como *dhcpcd*³.

4.3 Funcionamento

Esta seção tem por objetivo detalhar o funcionamento das funcionalidades oferecidas por *LauXen*.

4.3.1 Análise do Arquivo de Configuração

Durante a análise do *Arquivo de Configuração* o *Lançador* busca identificar a existência de ciclos de dependência entre duas ou mais máquinas virtuais. A existência de um ciclo significa que uma máquina virtual necessita de um ou mais serviços que executam em outra máquina virtual, e esta também depende de algum serviço que é oferecido pela primeira. Conseqüentemente, ambas não serão lançadas. Outra análise executada pelo *Lançador* é identificar máquinas virtuais que possuem como dependência uma máquina

²Programa que controla a inicialização de aplicativos durante o *boot* de sistemas operacionais Linux.

³Cliente de servidores DHCP.

virtual que não é lançada localmente, a menos que esta informação esteja explicitada em sua declaração. Isto tem por objetivo garantir que um nome errado impossibilite o lançamento da máquina virtual dependente, executado por seu *Monitor*. Ao ser encontrado qualquer um dos problemas mencionados, o *Lançador* interrompe a execução de *LauXen*, gerando mensagens de erro que auxiliam o administrador a resolvê-los. Um exemplo de mensagem de erro pode ser visto na figura 4.2, avisando o administrador sobre a existência de um ciclo entre as máquinas virtuais.

```

root@xen:~/implementacao
xen implementacao # lauxen launch
servidor1
servidor3
servidor2
servidor1
Ciclo Encontrado (Ler de baixo para cima)
xen implementacao #

```

Figura 4.2: Exemplo de Ciclo Encontrado em Arquivo de Configuração.

Caso erros não sejam identificados, o lançamento e o monitoramento das máquinas virtuais têm início, os quais serão detalhados nas próximas seções.

4.3.2 Lançamento das Máquinas Virtuais

Esta seção detalha o processo de lançamento das máquinas virtuais. Basicamente, este processo busca identificar as portas dos serviços das máquinas virtuais de que outra máquina virtual depende. Para isso, é utilizado um *scanner* de portas, e quando todas estiverem abertas o *Monitor* aciona o lançamento da máquina virtual que controla. Para auxiliar na descrição do funcionamento desta tarefa, considere a existência de um *Arquivo de Configuração* com três máquinas virtuais, as quais oferecem o serviço correspondente ao nome que lhe foi dado, sendo WWW, NFS-SERVER e DNS. Considere ainda que as duas primeiras possuam como dependência a máquina virtual DNS, e a primeira também depende da segunda. O *Lançador* lança ((2) da figura 4.1) em paralelo todos os *Monitores* das máquinas virtuais declaradas neste *Arquivo de Configuração*. Esta abordagem simplifica o lançamento das máquinas virtuais, pois cada uma é controlada por um *Monitor* independente. Além disso, isto facilita o lançamento de várias máquinas virtuais em

paralelo, aproveitando os benefícios de processadores com vários núcleos de execução. A figura 4.3 detalha o funcionamento dos *Monitores*.

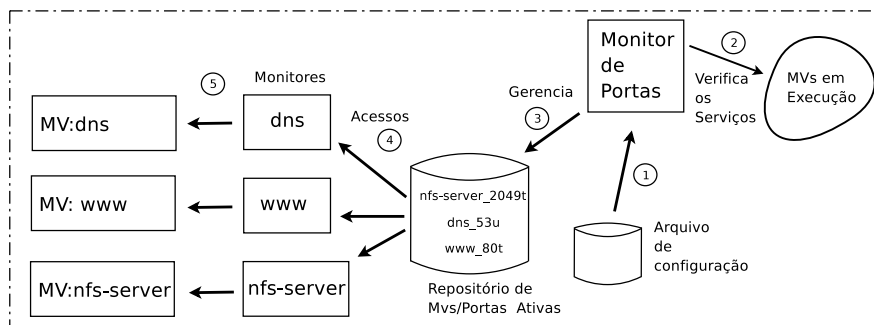


Figura 4.3: Visão Geral dos Monitores em Funcionamento

O *Monitor de Portas* lê (1) o *Arquivo de Configuração* criado pelo administrador recuperando as máquinas virtuais que devem ser lançadas, juntamente com seus serviços. Com estas informações, é criada uma estrutura interna de controle, que contém as máquinas virtuais e suas dependências. Estas informações são utilizadas na construção de um grafo, representado pela figura 4.4 e que também serviu de base para execução das rotinas de verificação descritas na seção 4.3.1. Este monitor verifica (2) os serviços que estão ativos nas máquinas virtuais em execução. Ao certificar que a porta de um serviço está aberta, o *Monitor de Portas* cria um arquivo dentro do *Repositório de MVs/Portas Ativas*, cujo nome obedece ao formato *NomeVM_PortaLetraProtocolo*, como mostrado no interior da figura que o representa. Quando este monitor localiza uma porta fechada o arquivo correspondente é apagado. As tarefas de criar e apagar estes arquivos são representadas por (3).

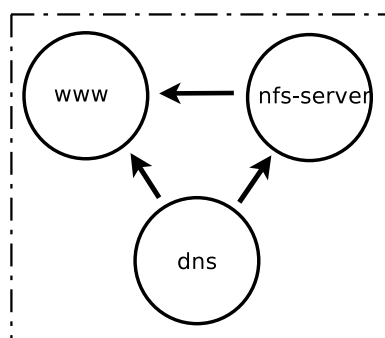


Figura 4.4: Grafo Correspondente aos Dados da Estrutura de Controle.

Os *Monitores* lançam (5) a máquina virtual que controlam com base no conteúdo do *Repositório de MVs/Portas Ativas*. Este processo é feito através de acessos (4) a este diretório, onde a existência de um arquivo cujo nome contém o nome da máquina virtual,

porta do serviço e seu protocolo significa que o serviço está ativo. A não existência deste arquivo com estas informações significa que o serviço está inativo. A razão para a existência deste repositório é centralizar a execução de varreduras de portas, diminuindo o tráfego na rede e uso da CPU. Além disso, é normal que várias máquinas virtuais dependam de uma máquina virtual em comum, esta abordagem faz com que várias varreduras nas portas desta MV partindo de diferentes *Monitores* seja evitada. No exemplo tomado, a máquina virtual DNS não possui nenhuma dependência, sendo imediatamente lançada. A máquina NFS-SERVER é lançada assim que seu *Monitor* comprovar a existência do arquivo relacionado com sua dependência (dns_53u neste caso), a mesma regra se aplica para a máquina WWW (arquivos dns_53u e nfs-server_2049t).

Conforme citado na seção 4, quando um *Monitor* efetua o lançamento de uma máquina virtual, o mesmo entra em um estágio de monitoramento, o qual será detalhado a seguir.

4.3.3 Monitoramento dos Serviços das Máquinas Virtuais

O monitoramento dos serviços das máquinas virtuais tem a função de identificar problemas nos serviços que estas executam. Se problemas forem identificados pelo *Monitor* de uma máquina virtual, uma ação reparadora é iniciada. Ao identificar um problema nos serviços da máquina virtual lançada, o *Monitor* aguarda um determinado tempo e volta a testar a porta do serviço com problemas. Caso esta estiver aberta, o processo de monitoramento continua normalmente, do contrário esta máquina virtual é reiniciada e tem seu monitoramento retomado ao término da reinicialização. Durante a reinicialização, os demais *Monitores* responsáveis por monitorar as máquinas virtuais que possuam como dependência o serviço da máquina virtual sendo reiniciada, pausam a execução da sua máquina virtual e aguardam o restabelecimento deste serviço, quando retomam a sua execução. Uma máquina virtual é pausada sempre que uma de suas dependências apresenta falhas e somente tem sua execução retomada quando o problema for corrigido.

A figura 4.5 mostra a estrutura interna de um *monitor*. Após o lançamento da máquina virtual controlada, o *Monitor* aguarda o término da inicialização dos seus serviços. Em seguida, o *Monitor* de uma MV monitora os serviços que essa máquina oferece e os serviços das quais ela depende. Caso detecte que algum dos serviços das quais ela depende não esteja ativo, o *Monitor* pausa a execução de sua máquina virtual e sua execução somente é retomada após o restabelecimento do serviço inativo. O *Monitor* da máquina virtual que

máquina virtual WWW é a primeira a ser desligada, NFS-SERVER a segunda e só então DNS é desligada. Mais uma vez pensou-se nos benefícios de processadores com vários núcleos de execução, pois o desligamento de cada máquina virtual provavelmente será executado por um dos núcleos destes processadores, agilizando o processo. Considerando a presença de uma quarta máquina virtual neste exemplo e que a mesma não ofereça serviços para as demais, esta seria desligada simultaneamente com a WWW. O desligamento das máquinas virtuais difere do lançamento em um ponto. Ao invés de verificar as portas dos serviços, o comando *ping* é utilizado. Quando a resposta do *ping* não for recebida, assume-se que a interface de rede da máquina virtual em processo de desligamento foi desligada e as máquinas virtuais das quais ela dependia podem ter seu desligamento iniciado. A figura 4.6 exibe os processos executados durante o desligamento.

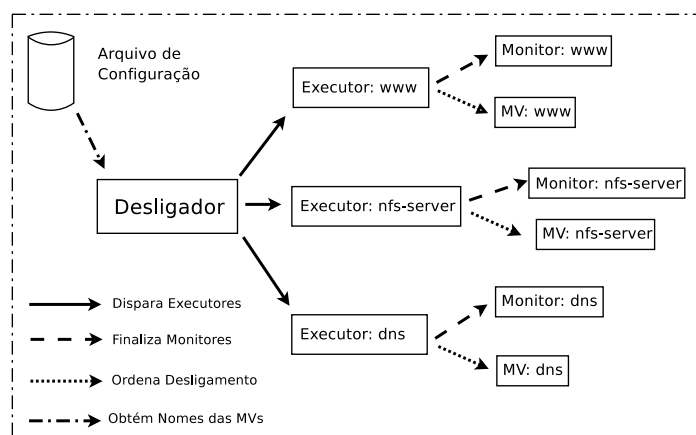


Figura 4.6: Visão Geral do Sistema de Desligamento das MVs.

O *Desligador* obtém os nomes das máquinas virtuais a serem desligadas acessando o arquivo de configuração. De posse destes nomes, o *Desligador* dispara *Executores* para cada nome recuperado. Estes executores são responsáveis por duas tarefas: Desativar os *Monitores* e desligar as máquinas virtuais. Estas atividades são executadas a medida que uma máquina virtual puder ser desligada, de acordo com o processo descrito acima. É necessário desativar os *Monitores* pois do contrário este perceberá que a máquina virtual que monitora parou de funcionar e imediatamente tentará relança-la.

4.3.5 Gerenciamento de Memória

Esta funcionalidade é importante porque situações onde uma máquina virtual venha a necessitar de mais memória do que lhe foi concedida inicialmente podem acontecer. Cada máquina virtual em Xen possui seu próprio arquivo de configuração, onde deve

ser setado um valor para o parâmetro *memory*, que representa a quantidade de memória a ser reservada para esta máquina virtual. Sendo assim, este problema poderia ser resolvido lançando as máquinas virtuais com bastante memória concedida neste parâmetro de configuração, contudo, esta mesma quantidade é descontada do montante disponível, resolvendo o problema de poucas MVs.

A solução adotada é lançar as máquinas com uma boa quantidade de memória setada neste parâmetro, e logo em seguida reduzi-la para uma quantidade que é suficiente na maioria dos casos. Esta quantidade é informada no *Arquivo de Configuração* utilizado durante o lançamento e passada como argumento para a função *xen mem-set*, que altera a quantidade de memória destinada a cada máquina virtual que já foi lançada. Caso ajustes sejam necessários na quantidade de memória destinada a uma máquina virtual em um determinado momento, o administrador pode utilizar esta função em tempo de execução para aumentá-la ou diminuí-la. É interessante ressaltar que o parâmetro *memory* define o máximo de memória que uma MV poderá alocar durante a sua criação, mas se durante a execução de uma MV o mesmo for reduzido, a diferença entre a quantidade de memória anterior e a setada é devolvida para o sistema, podendo ser alocada a outras máquinas virtuais.

4.3.6 *LauXen*: Outras Funcionalidades

LauXen é iniciado durante a inicialização do sistema através de um *script init*⁴. Como os *Monitores* buscam constantemente identificar serviços inativos, a manutenção de serviços das máquinas virtuais pode levar a situações em que a máquina virtual em manutenção seja reiniciada indevidamente pelo *Monitor* que a monitora. Para evitar estes transtornos, existem duas rotinas administrativas, chamadas *lmm*⁵ *pause* e *lmm* *unpause*, que permitem pausar e retomar a execução de um determinado *Monitor*, passando nome da MV monitorada como argumento.

O lançamento de novas máquinas virtuais também é facilitado. Para isso, basta acrescentar a máquina virtual a ser lançada no *Arquivo de Configuração* e seguir dois procedimentos básicos. *LauXen* oferece outras duas rotinas administrativas, *lauxen rebuild* e *lauxen launch*. A primeira realiza todos os testes (citados na seção 4.3.1) necessários para garantir que a máquina virtual acrescentada não gere problemas na configuração ante-

⁴Programa que controla a inicialização de aplicativos durante o *boot* de sistemas operacionais Linux.

⁵*LauXen* Monitor Manager.

rior, como a criação de um ciclo entre as máquinas virtuais, prejudicando um lançamento futuro. A segunda realiza o lançamento das máquinas virtuais presentes no *Arquivo de Configuração* que não estejam em execução, ou, caso o nome de uma MV seja passada como argumento, somente esta será lançada.

Outra tarefa normalmente executada é o desligamento de uma máquina virtual. Para isso, existe outra rotina administrativa oferecida por *LauXen*, chamada *lauxen shutdown*. A execução deste rotina passando como parâmetro o nome de uma máquina virtual faz com que somente esta seja desligada. Entretanto, caso nenhum parâmetro seja passado, o desligamento ordenado de todas as máquinas virtuais tem início.

4.4 Arquivos de Configuração

O *Arquivo de Configuração* descreve as dependências que uma máquina possui. Cada linha deste arquivo contém a descrição dos serviços que a máquina dependente oferece, além dos serviços dos quais depende localizados em outras máquinas virtuais. Um exemplo de *Arquivo de Configuração* pode ser observado na figura 4.7.

```
www(22T, 80T, 443T){300}: dns(53U) nfs-server(2049T)
dns(22T, 53U){50}: nada
nfs-server(2049T){1024}: dns(53U)
```

Figura 4.7: Arquivo de Configuração Utilizado Para o Lançamento das MVs.

A primeira parte de cada linha delimitada pelo “:” contém o nome da máquina a ser lançada e as portas de comunicação utilizadas pelos serviços que serão executados. Já as letras “T” e “U” que sucedem os números das portas representam o protocolo de comunicação utilizado, sendo “T” para TCP e “U” para UDP. A segunda parte de cada linha, posterior ao “:”, contém as dependências da máquina a ser lançada. Ali devem constar o nome da máquina virtual, porta e protocolo dos serviços que a máquina a ser lançada depende para que tenha o seu lançamento permitido. As letras “T” e “U” têm o mesmo significado da primeira parte. O número entre {} representa a quantidade de memória que será alocada durante o lançamento das máquinas virtuais.

Conforme citado na seção 4.3.4, um novo *Arquivo de Configuração* é criado para a realização do desligamento ordenado das máquinas virtuais em execução. No caso do *Arquivo de Configuração* da figura 4.7, o seguinte arquivo seria gerado:

A diferença entre ambos é que ao invés de listar depois do “:” as máquinas virtuais

```
www:  
dns: www nfs-server  
nfs-server: www
```

Figura 4.8: Arquivo de Configuração Utilizado Durante o Desligamento das MVs.

que uma máquina virtual depende, são listadas as máquinas virtuais que dela depende. A ausência da especificação dos serviços oferecidos se deve ao fato de que não são feitas varreduras de portas durante o desligamento.

4.5 Avaliação

A avaliação realizada teve como objetivos validar o correto funcionamento da ferramenta desenvolvida, identificar seu impacto no desempenho do servidor e os tempos de resposta dos *Monitores*.

4.5.1 Metodologia

O teste foi realizado em 2 cenários hipotéticos, cada um representando uma hierarquia distinta de dependências entre as máquinas virtuais. Os *Arquivos de Configuração* utilizados possuem as informações necessárias para o lançamento de 9 máquinas virtuais que executam servidores SSH (*Secure Shell*). Para fins de teste, considerou-se uma situação hipotética onde existem dependências entre estes servidores. Optou-se por este cenário devido à facilidade de criá-lo, e também devido ao fato de que o SSH é um serviço latente do sistema, pois só vai consumir processamento quando houver uma conexão para ele. Deste modo, é possível avaliar a utilização da CPU em cada uma das máquinas virtuais de forma mais precisa.

A validação do lançamento ordenado deu-se através da medição do tempo decorrido desde o comando para lançar todas as MVs até o lançamento de uma dada MV. A ordem crescente destes tempos representa ordem lógica do lançamento das máquinas virtuais. O mesmo se aplica para a validação do desligamento ordenado das máquinas virtuais.

A validação dos *Monitores* deu-se através de pausas e despausas de uma MV. A medição do tempo decorrido para que um determinado *Monitor* pause sua máquina virtual devido a problemas em suas dependência é utilizada, sendo que a ordem crescente destes tempos reflete na ordem lógica em que as MVs foram pausadas ou despausadas, comprovando o funcionamento dos *Monitores*.

O impacto no desempenho do servidor é exibido através de imagens que mostram a utilização de CPU e memória dos componentes da arquitetura desenvolvida. Os cenários a seguir foram utilizados na validação da ferramenta.

4.6 Cenário A

A figura 4.9 exibe o arquivo de configuração utilizado durante o teste neste cenário.

```

servidor1(22t): nada
servidor2(22t){40}: servidor1(22t) servidor6(22t)
servidor3(22t){40}: servidor1(22t)
servidor4(22t){40}: servidor1(22t)
servidor5(22t){40}: servidor1(22t)
servidor6(22t){40}: servidor1(22t)
servidor7(22t){40}: servidor1(22t)
servidor8(22t){40}: servidor1(22t) servidor3(22t)
servidor9(22t){40}: servidor1(22t) servidor5(22t)

```

Figura 4.9: Arquivo de Configuração do Cenário A

O grafo da figura 4.10 tem por objetivo facilitar a visualização desta hierarquia de dependências. Para uma melhor visualização, somente o dígito identificador da MV é exibido.

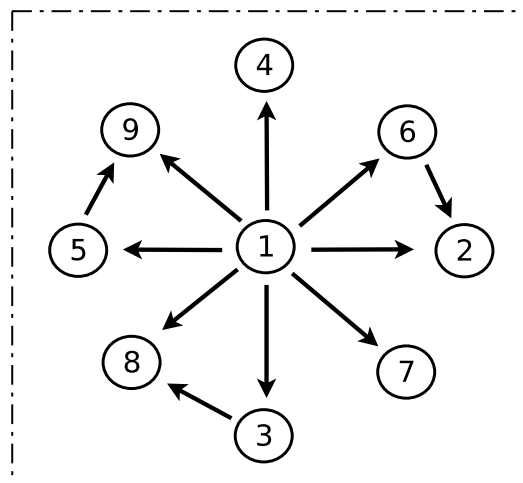


Figura 4.10: Grafo Correspondente ao Cenário A.

O lançamento e o desligamento ordenado das máquinas virtuais foi realizado com sucesso. A tabela 4.1 exibe os resultados. A primeira linha desta tabela representa o dígito identificador da máquina virtual. A segunda linha representa o tempo decorrido após o

comando para lançar ordenadamente todas as máquinas virtuais ser executado para que a MV seja lançada. A terceira linha exibe o tempo decorrido após a execução do comando para desligar todas as máquinas virtuais para que a MV da coluna comece a ser desligada. O maior tempo de cada linha indica o tempo total decorrido desde o início da ação da primeira coluna até que todas as MVs façam a ação pretendida.

Ação	1	2	3	4	5	6	7	8	9
Lançamento	0s	75s	22s	29s	34s	31s	24s	68s	79s
Desligamento	44s	1s	20s	3s	23s	24s	1s	0s	1s

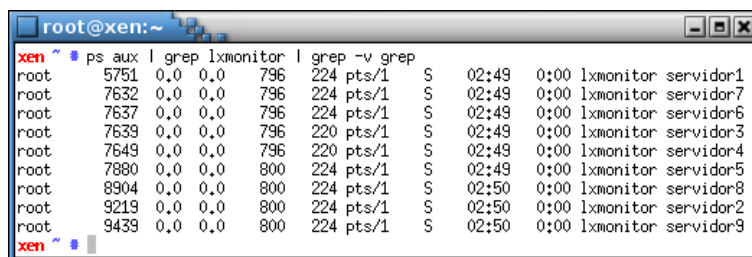
Tabela 4.1: Tabela com os tempos de inicialização/desligamento das MVs do Cenário A

Para verificar o funcionamento dos *Monitores*, a máquina virtual servidor1, ((1) na figura 4.10), foi pausada e despausada por 3 vezes. A tabela 4.2 exibe os resultados. Os tempos exibidos nas linhas 2,4 e 6 são os tempos decorridos para que uma MV seja pausada pelo seu *Monitor*, após a MV servidor1 ser pausada. As linhas 3, 5 e 7 representam o tempo decorrido desde a retomada da execução da MV servidor1 até a retomada da execução da MV da respectiva coluna ordenada por seu *Monitor*. Os tempos para que todas as MVs sejam pausadas é relativamente curto pois todas dependem diretamente da MV servidor1, enquanto as MVs servidor2, servidor8, servidor9 necessitam dos serviços das MVs servidor6, servidor3 e servidor5, respectivamente, para terem a sua execução retomada, justificando a ocorrência de tempos maiores para retomada na última linha da tabela. A última coluna indica o tempo total decorrido desde o início da ação da primeira coluna até que todos os *Monitores* reajam a esta mudança.

Ação	1	2	3	4	5	6	7	8	9
Pausa 1	0s	3s	4s	4s	4s	4s	4s	3s	4s
Despausa 1	0s	6s	5s	5s	5s	5s	5s	5s	6s
Pausa 2	0s	4s	4s	4s	4s	4s	3s	4s	4s
Despausa 2	0s	5s	3s	4s	3s	3s	4s	4s	6s
Pausa 3	0s	4s	4s	4s	4s	4s	4s	4s	4s
Despausa 3	0s	4s	3s	3s	3s	3s	3s	4s	4s

Tabela 4.2: Tabela de validação do monitoramento do Cenário A

A figura 4.11 exibe os *Monitores* em funcionamento. É interessante ressaltar o baixo consumo de memória dos *Monitores*, ficando em torno de 224KB cada. Os *Monitores* executaram por 25 minutos, e como pode ser observado, não há um consumo significativo de CPU.



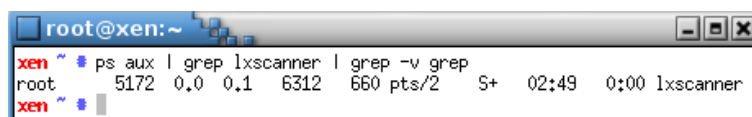
```

root@xen:~
xen ~ # ps aux | grep lxmonitor | grep -v grep
root 5751 0,0 0,0 796 224 pts/1 S 02:49 0:00 lxmonitor servidor1
root 7632 0,0 0,0 796 224 pts/1 S 02:49 0:00 lxmonitor servidor7
root 7637 0,0 0,0 796 224 pts/1 S 02:49 0:00 lxmonitor servidor6
root 7639 0,0 0,0 796 220 pts/1 S 02:49 0:00 lxmonitor servidor3
root 7649 0,0 0,0 796 220 pts/1 S 02:49 0:00 lxmonitor servidor4
root 7880 0,0 0,0 800 224 pts/1 S 02:49 0:00 lxmonitor servidor5
root 8904 0,0 0,0 800 224 pts/1 S 02:50 0:00 lxmonitor servidor8
root 9219 0,0 0,0 800 224 pts/1 S 02:50 0:00 lxmonitor servidor2
root 9439 0,0 0,0 800 224 pts/1 S 02:50 0:00 lxmonitor servidor9
xen ~ #

```

Figura 4.11: Monitores em Funcionamento no Cenário A.

A figura 4.12 mostra o *Monitor de Portas* em funcionamento. Há um consumo maior de memória por parte do *Monitor de Portas*, cerca de 660KB neste caso, 0,1% do total disponível. Não há um consumo significativo de CPU por este componente durante os 25 minutos em que executou.



```

root@xen:~
xen ~ # ps aux | grep lxscanner | grep -v grep
root 5172 0,0 0,1 6312 660 pts/2 S+ 02:49 0:00 lxscanner
xen ~ #

```

Figura 4.12: Monitor de Portas em Funcionamento no Cenário A.

A figura 4.13 mostra todas as MVs em funcionamento. Nas MVs servidor1 a servidor9 o consumo de CPU se deve à varredura de portas pelo *Monitor de Portas*, pois não há componentes do sistema desenvolvido nelas executando. A coluna CPU(sec) mostra a utilização do processador em segundos. Neste caso, a maioria das máquinas virtuais ocuparam o processador por 22 segundos durante os 1500 segundos (25 minutos), que foi a duração do teste. Retirando 7 segundos gastos durante o *boot* de cada MV, resulta em 15 segundos, o que corresponde a 0,01% de utilização da CPU no período considerado. Em relação a Dom-0, no momento do início do teste ele havia consumido 20 segundos de tempo de CPU, sendo que ao término da inicialização de todas as MVs esse valor aumentou para 49 segundos. Como pode ser observado, no momento da captura desta imagem este valor foi de 67 segundos, transcorrendo então 18 segundos desde o fim da inicialização de todas as MVs até o final da realização do teste, o que corresponde a 0,012% de utilização da CPU por parte de todos os *Monitores* em execução e o *Monitor e Portas*. Estes dados mostram um baixo impacto causado pela ferramenta desenvolvida na máquina em que está sendo executada.

Uma modificação no *Arquivo de Configuração* originou um ciclo entre todas as MVs deste cenário, exceto a MV servidor1, como pode ser observado no grafo da figura 4.14 que o representa.

NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPU	NETS	NETTX(k)	NETRX(k)	VBDs	VBD_00	VBD_RD	VBD_WR	SSID
Domain-0	-----r	67	2,8	573440	56,6	no limit	n/a	1	4	3657	3230	0	0	0	0	0
servidor1	--b---	22	2,3	81920	8,1	81920	8,1	1	1	283	429	1	0	542	4858	0
servidor2	--b---	22	2,2	40960	4,0	40960	4,0	1	1	292	400	1	0	551	4759	0
servidor3	--b---	22	2,3	40960	4,0	40960	4,0	1	1	281	411	1	0	548	4813	0
servidor4	--b---	22	2,3	40960	4,0	40960	4,0	1	1	293	422	1	0	547	4817	0
servidor5	--b---	22	2,3	40960	4,0	40960	4,0	1	1	291	422	1	0	550	4819	0
servidor6	--b---	22	2,3	40960	4,0	40960	4,0	1	1	286	418	1	0	551	4816	0
servidor7	--b---	23	2,3	40960	4,0	40960	4,0	1	1	276	414	1	0	2370	4906	0
servidor8	--b---	22	2,3	40960	4,0	40960	4,0	1	1	282	396	1	0	2367	4816	0
servidor9	--b---	22	2,2	40960	4,0	40960	4,0	1	1	282	391	1	0	2374	4830	0

Figura 4.13: MVs em Funcionamento no Cenário A.

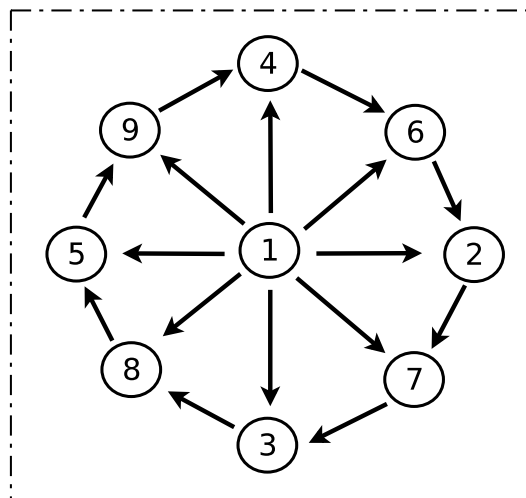


Figura 4.14: Grafo que Representa o Ciclo Gerado.

Com a alteração do *Arquivo de Configuração* é necessário executar a rotina de validação da nova configuração. A figura 4.15 exibe esta ação e o respectivo ciclo foi encontrado e informado na tela do servidor.

```

root@xen:~
xen ~ # lauxen rebuild
servidor2
servidor7
servidor3
servidor8
servidor5
servidor9
servidor4
servidor6
servidor2
Ciclo Encontrado (Ler de baixo para cima)
xen ~ #
  
```

Figura 4.15: Alerta Gerado pela Rotina de Validação.

4.7 Cenário B

A figura 4.16 exibe o arquivo de configuração utilizado durante o teste neste cenário. O grafo da figura 4.17 tem por objetivo facilitar a visualização desta hierarquia de

```

servidor1(22t){40}: nada
servidor2(22t){40}: servidor1(22t)
servidor3(22t){40}: servidor1(22t)
servidor4(22t){40}: servidor2(22t)
servidor5(22t){40}: servidor2(22t)
servidor6(22t){40}: servidor3(22t)
servidor7(22t){40}: servidor3(22t)
servidor8(22t){40}: servidor5(22t) servidor6(22t)
servidor9(22t){40}: servidor4(22t) servidor7(22t) servidor8(22t)

```

Figura 4.16: Arquivo de Configuração do Cenário B

dependências. Novamente, para uma melhor visualização somente o dígito identificador da MV é exibido.

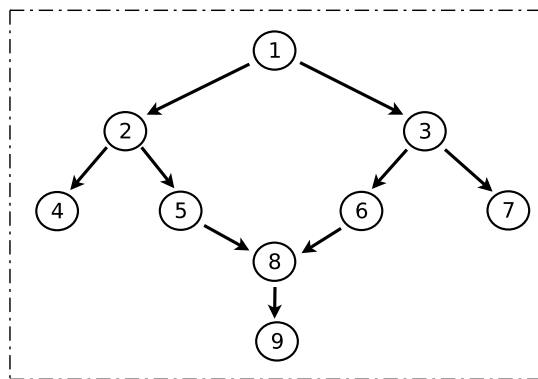


Figura 4.17: Grafo Correspondente ao Cenário B.

O lançamento e o desligamento ordenado das máquinas virtuais foi realizado com sucesso. A tabela 4.3 exibe os resultados e o seu significado é semelhante ao da tabela 4.1. Como pode ser notado pela tabela, os servidores servidor2 e servidor3 são iniciados quase que simultaneamente, devido a posição que ocupam na hierarquia de dependências. O mesmo se aplica aos servidores4 a servidor7, sendo que havia 2 MVs sendo lançadas simultaneamente e por isso a demora para elas terem sua inicialização liberada. O servidor8 tem sua inicialização um pouco retardada devido à carga em que o computador se encontrava, pois neste caso havia 4 MVs sendo inicializadas ao mesmo tempo. O curto intervalo de tempo entre o início do lançamento do servidor8 e do servidor9 é devido a pouca carga no servidor, pois somente uma MV estava sendo inicializada. O maior tempo de cada linha indica o tempo total decorrido desde o início da ação da primeira coluna até que todas as MVs façam a ação pretendida. Estes são alguns detalhes que mereciam destaque nesta tabela.

Ação	1	2	3	4	5	6	7	8	9
Lançamento	0s	20s	21s	50s	47s	48s	50s	87s	105s
Desligamento	45s	34s	32s	7s	9s	9s	8s	5s	0s

Tabela 4.3: Tabela com os tempos de inicialização/desligamento das MVs do Cenário B

Para verificar o funcionamento dos *Monitores*, a máquina virtual servidor1, ((1) na figura 4.17), foi pausada e despausada por 3 vezes. A tabela 4.4 exibe os resultados e a explicação para os valores encontrados é semelhante a da tabela 4.2. As MVs servidor1 e servidor2 são pausadas e despausadas simultaneamente pois ambas dependem diretamente da MV servidor1. As MVs servidor4 e servidor5 dependem diretamente da MV servidor2 e também são pausadas ou despausadas ao mesmo tempo. O mesmo se aplica às MVs servidor6 e servidor7, pois ambas dependem da MV servidor3. A última coluna indica o tempo total decorrido desde o início da ação da primeira coluna até que todos os *Monitores* reajam a esta mudança.

Ação	1	2	3	4	5	6	7	8	9
Pausa 1	0s	4s	4s	10s	10s	7s	7s	12s	13s
Despausa 1	0s	2s	3s	12s	12s	12s	13s	17s	20s
Pausa 2	0s	5s	5s	15s	15s	15s	15s	20s	22s
Despausa 2	0s	3s	3s	9s	8s	4s	3s	10s	12s
Pausa 3	0s	3s	3s	9s	9s	6s	6s	11s	12s
Despausa 3	0s	1s	2s	8s	8s	2s	3s	9s	12s

Tabela 4.4: Tabela de validação do monitoramento do Cenário B

A figura 4.18 exibe os *Monitores* em funcionamento. É interessante ressaltar o baixo consumo de memória dos *Monitores*, ficando também em torno de 224KB e novamente não há um consumo significativo de CPU durante os 25 minutos do teste.

```

root@xen:~
xen ~ # ps aux | grep lxmonitor | grep -v grep
root    5633  0.0  0.0   800   224 pts/0    S   03:20   0:00 lxmonitor servidor1
root    6893  0.0  0.0   796   224 pts/0    S   03:20   0:00 lxmonitor servidor2
root    7079  0.0  0.0   800   220 pts/0    S   03:20   0:00 lxmonitor servidor3
root    8759  0.0  0.0   800   224 pts/0    S   03:21   0:00 lxmonitor servidor6
root    8792  0.0  0.0   800   224 pts/0    S   03:21   0:00 lxmonitor servidor5
root    8793  0.0  0.0   800   224 pts/0    S   03:21   0:00 lxmonitor servidor4
root    8989  0.0  0.0   796   224 pts/0    S   03:21   0:00 lxmonitor servidor7
root    9718  0.0  0.0   800   224 pts/0    S   03:21   0:00 lxmonitor servidor8
root   10125  0.0  0.0   796   224 pts/0    S   03:22   0:00 lxmonitor servidor9
xen ~ #

```

Figura 4.18: Monitores em Funcionamento no Cenário B.

A figura 4.19 mostra o *Monitor de Portas* em funcionamento. Há um consumo maior de memória por parte do *Monitor de Portas*, cerca de 664KB neste caso, 0,1% do total

disponível. Também não há um consumo significativo de CPU durante os 25 minutos do teste por parte deste componente.



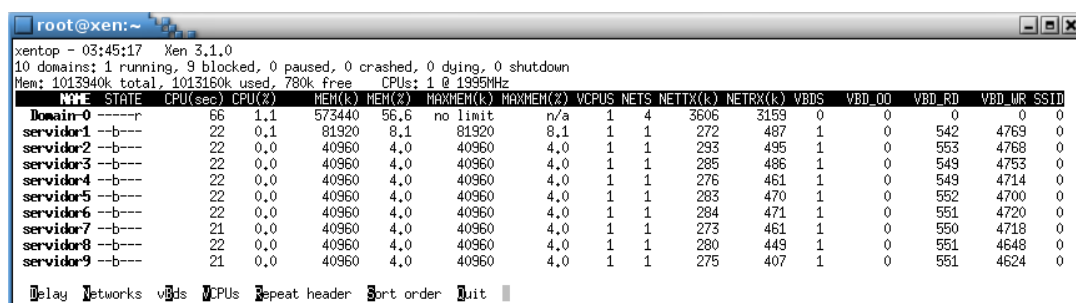
```

root@xen:~
xen ~ # ps aux | grep lxscanner | grep -v grep
root   5122  0,0  0,1  6312  664 pts/1  S+   03:20   0:00 lxscanner
xen ~ #

```

Figura 4.19: Monitor de Portas em Funcionamento no Cenário B.

A figura 4.20 mostra todas as MVs em funcionamento. A coluna CPU(sec) mostra a utilização do processador em segundos. Neste caso, a maioria das máquinas virtuais ocuparam o processador por 22 segundos durante os 1500 segundos (25 minutos), que foi a duração do teste. Retirando 7 segundos gastos durante o *boot* de cada MV, resulta em 15 segundos, o que corresponde a 0,01% de utilização da CPU no período considerado. Em relação a Dom-0, no momento do início do teste ele havia consumido 20 segundos de tempo de CPU, sendo que ao término da inicialização de todas as MVs esse valor aumentou para 50 segundos. Como pode ser observado, no momento da captura desta imagem este valor foi de 66 segundos, transcorrendo então 16 segundos desde o fim da inicialização de todas as MVs até o final da realização do teste, o que corresponde a 0,01% de utilização da CPU por parte de todos os *Monitores* em execução e o *Monitor e Portas*. Os dados resultantes do teste neste cenário são semelhantes aos dados obtidos no primeiro cenário e mostram uma baixa utilização de recursos computacionais da máquina em que a ferramenta está sendo executada.



```

root@xen:~
xentop - 03:45:17 Xen 3.1.0
10 domains: 1 running, 9 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 1013940k total, 1013160k used, 780k free CPUs: 1 @ 1995MHz

```

NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	NETS	NETTX(k)	NETRX(k)	VBDS	VBD_00	VBD_RD	VBD_WR	SSID
Domain-0	-----r	66	1,1	573440	56,6	no limit	n/a	1	4	3606	3159	0	0	0	0	0
servidor1	--b---	22	0,1	81320	8,1	81320	8,1	1	1	272	487	1	0	542	4769	0
servidor2	--b---	22	0,0	40960	4,0	40960	4,0	1	1	293	495	1	0	553	4768	0
servidor3	--b---	22	0,0	40960	4,0	40960	4,0	1	1	285	486	1	0	549	4753	0
servidor4	--b---	22	0,0	40960	4,0	40960	4,0	1	1	276	461	1	0	549	4714	0
servidor5	--b---	22	0,0	40960	4,0	40960	4,0	1	1	283	470	1	0	552	4700	0
servidor6	--b---	22	0,0	40960	4,0	40960	4,0	1	1	284	471	1	0	551	4720	0
servidor7	--b---	21	0,0	40960	4,0	40960	4,0	1	1	273	461	1	0	550	4718	0
servidor8	--b---	22	0,0	40960	4,0	40960	4,0	1	1	280	449	1	0	551	4648	0
servidor9	--b---	21	0,0	40960	4,0	40960	4,0	1	1	275	407	1	0	551	4624	0

Figura 4.20: MVs em Funcionamento no Cenário B.

O computador utilizado na realização deste teste possui o processador AMD Turion 64 2GHz (*Single Core*), 1024MB de memória RAM DDR SDRAM 333 MHz e HD de 120GB com 4200RPM.

5 CONCLUSÃO E PRÓXIMAS ATIVIDADES

O uso de virtualização tem se intensificado e conseqüentemente a complexidade do gerenciamento destes ambientes tende a aumentar. Este trabalho apresentou uma ferramenta que busca auxiliar no gerenciamento de ambientes virtualizados, sendo responsável por lançar e desativar as máquinas virtuais de forma ordenada, respeitando a hierarquia de dependências entre elas. A tarefa de monitorar os serviços também é importante, prevenindo problemas que possam acontecer devido a falhas em uma das máquinas virtuais. Os testes realizados mostraram que *LauXen* consome poucos recursos computacionais, o que garante um baixo impacto no desempenho do servidor.

A estrutura interna de *LauXen* permite a inserção de novas funcionalidades sem grandes problemas. Uma funcionalidade prevista para ser implementada é o mecanismo de migração de máquinas virtuais sem que a estrutura de controle dos *Monitores* seja comprometida. Além disso, no estágio em que a ferramenta se encontra, quando um *Monitor* detecta que um serviço da máquina virtual que monitora parou, ele simplesmente aguarda um determinado tempo e caso o serviço não estiver funcionando após este tempo, a decisão tomada é reiniciar a máquina virtual.

Contudo, no momento não há garantia de que isto resolva o problema do serviço que apresentou problemas. Sendo assim, novas estratégias para a suprir esta falta são necessárias. Para isso, pretende-se tentar reiniciar a máquina virtual e se isto não resolver o problema, a mesma será desativada e em seu lugar será lançada uma máquina virtual reserva, contendo um estado consistente da máquina virtual que apresentou defeito.

Uma terceira funcionalidade planejada é realizar a alocação dinâmica de memória para as máquinas virtuais. Atualmente o administrador é responsável por administrar a quantidade de memória destinada a cada máquina virtual, no entanto, pretende-se implementar um mecanismo que ao perceber que uma máquina virtual necessite de mais memória do

que lhe foi concedida inicialmente, faça a concessão de mais memória a esta máquina automaticamente.

REFERÊNCIAS

APPARAO, P.; MAKINENI, S.; NEWELL, D. Characterization of network processing overheads in Xen. **VTDC 2006. First International Workshop on**, [S.l.], v.17, n.17, p.2–2, Nov 2006. <http://ieeexplore.ieee.org/iel5/4299339/4299340/04299347.pdf?tp=&arnumber=4299347&isnumber=4299340>.

BARHAM, P.; DRAGOVIC, B.; FRASER, K.; HAND, S.; HARRIS, T.; HO, A.; NEUGEBAUER, R.; PRATT, I.; WARFIELD, A. Xen and the art of virtualization. In: **ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES (SOSP '03)**, 19., 2003, Bolton Landing, USA. **Proceedings...** ACM, 2003. p.164–177.

BARHAM, P. R.; DRAGOVIC, B.; FRASER, K. A.; HAND, S. M.; HARRIS, T. L.; HO, A. C.; KOTSOVINOS, E.; MADHAVAPEDDY, A. V.; NEUGEBAUER, R.; PRATT, I. A.; WARFIELD, A. K. **Xen 2002**. Cambridge, United Kingdom: University of Cambridge - Computer Laboratory, 2003. <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-553.pdf>. (553).

CAMBRIDGE COMPUTER LABORATORY, U. of. **Xen Users' Manual, V3.0**. <http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user/user.html>.

CHEN, P.; NOBLE, B. When virtual is better than real [operating system relocation to virtual machines]. In: **HOT TOPICS IN OPERATING SYSTEMS, 2001. PROCEEDINGS OF THE EIGHTH WORKSHOP ON**, 2001. **Anais...** IEEE, 2001. p.133–138. <http://ieeexplore.ieee.org/iel5/7758/21324/00990073.pdf?tp=&arnumber=990073&isnumber=21324>.

ENOMALY. **Enomalism - Virtualized Management Console**. <http://www.enomalism.com>.

GOLDBERG, R. Architecture of Virtual Machines. **AFIPS National Computer Conference**, New York, NY, USA, 1973.

GOLDBERG, R. Survey of virtual machine research. **IEEE Computer**, [S.l.], v.7, n.6, p.34–45, 1974.

GOTH, G. Virtualization: old technology offers huge new potential. **IEEE Distributed Systems Online**, [S.l.], v.8, n.2, 2007. http://dsonline.computer.org/portal/site/dsonline/menuitem.6dd2a408dbe4a94be487e0606bcd45f3/index.jsp?&pName=dso_level1_article&TheCat=1025&path=dsonline/2007/02&file=o2003.xml.

HAZARD, H.; JEDI, J. **XenMan - Xen management tool**. <http://sourceforge.net/projects/xenman/>.

KOSLOVSKI, G.; BOUFLEUR, M. P.; CHARÃO, A. S. Adxen: uma ferramenta para administração de arquiteturas virtualizadas distribuídas baseadas em xen. **WSCAD 2007**, [S.l.], 2007.

LAUREANO, M.; MAZIERO, C.; JAMHOUR, E. Intrusion Detection in Virtual Machine Environments. In: **EUROMICRO CONFERENCE, 2004. PROCEEDINGS. 30TH, 2004. Anais...** IEEE, 2004. p.520–525. <http://ieeexplore.ieee.org/iel5/9268/29441/01333416.pdf?tp=\arnumber=1333416&isnumber=29441>.

LibVirt - Virtualization API. <http://libvirt.org/index.html>.

MENASCE, D. Virtualization: concepts, applications and performance modeling. In: **CMG PROCEEDINGS, 2005. Anais...** [S.l.: s.n.], 2005. <http://cs.gmu.edu/~menasce/papers/menasce-cmg05-virtualization.pdf>.

QUYNH, N. A.; TAKEFUJI, Y. Secure Reincarnation of Compromised Servers Using Xen Based Time-Forking Virtual Machines. **ICNS '06. International conference**

on, [S.l.], p.90–90, 2006. <http://ieeexplore.ieee.org/iel5/4144774/4144775/04144882.pdf?tp=&arnumber=4144882&isnumber=4144775>.

ROSE, R. **Survey of system virtualization techniques**. <http://www.robertwrose.com/vita/rose-virtualization.pdf>.

ROSENBLUM M.; GARFINKEL, T. Virtual Machine Monitors: current technology and future trends. **IEEE Computer**, [S.l.], v.38, n.5, p.39–47, may 2005. <http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/2/30853/01430630.pdf&arnumber=1430630>.

SINGH, A. **An Introduction to Virtualization**. <http://www.kernelthread.com/publications/virtualization/>.

SMITH, J.; NAIR, R. The architecture of virtual machines. **IEEE Computer**, [S.l.], v.38, n.5, p.32–38, may 2005. <http://ieeexplore.ieee.org/iel5/2/30853/01430629.pdf?tp=&arnumber=1430629&isnumber=30853>.

SUGERMAN, J.; VENKITACHALAM, G.; LIM, B.-H. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In: **USENIX ANNUAL TECHNICAL CONFERENCE, 2001.**, 2001. **Proceedings...** Usenix Assoc., 2001. p.1–14.

VAUGHAN-NICHOLS, S. New Approach to Virtualization Is a Lightweight. **IEEE Computer**, [S.l.], v.39, n.11, p.12–14, nov 2006. <http://ieeexplore.ieee.org/iel5/2/4014747/04014757.pdf?tp=&arnumber=4014757&isnumber=4014747>.

VIRT-MANAGER: virtual machine manager. <http://virt-manager.et.redhat.com/>.

WANG, Z.; XIAOYUN, Z.; PADALA, P.; SINGHAL, S. Capacity and Performance Overhead in Dynamic Resource Allocation to Virtual Containers. **IM '07. 10th IFIP/IEEE International Symposium on**, [S.l.], p.149–158, may 2007. <http://ieeexplore.ieee.org/iel5/4258513/4258514/04258531.pdf?tp=&arnumber=4258531&isnumber=4258514>.

XENSOURCE. **Xen**: enterprise grade open source virtualization - a xensource white paper v06012006. [S.l.]: XenSource Inc, 2005. http://xen.org/files/xensource_wp2.pdf.

YOUNGGYUN, K.; KNAUERHASE, R.; BRETT, P.; BOWMAN, M.; WEN, Z.; PU, C. An Analysis of Performance Interference Effects in Virtual Environments. **ISPASS 2007. IEEE International Symposium on**, [S.l.], p.200–209, apr 2007. <http://ieeexplore.ieee.org/iel5/4211006/4211007/04211036.pdf?tp=&arnumber=4211036&isnumber=4211007>.

ZHAO, X.; BORDERS, K.; PRAKASH, A. Towards Protecting Sensitive Files in a Compromised System. In: **SECURITY IN STORAGE WORKSHOP, 2004. Anais...** IEEE, 2004. p.8. <http://ieeexplore.ieee.org/iel5/10842/34160/01628479.pdf?tp=&arnumber=1628479&isnumber=34160>.

APÊNDICE A INSTALAÇÃO E UTILIZAÇÃO DE *LAUXEN*

O objetivo deste apêndice é guiar o utilizador nas tarefas de instalação e utilização de *LauXen*. As atividades de instalação do Xen e criação de máquinas virtuais não fazem parte do escopo deste trabalho.

A.1 Pré-requisitos

- *Scanner* de portas Nmap
- MMV Xen instalado
- Compilador GCC

A.2 Instalação e Configuração

Inicialmente, *LauXen* possui todos os seus arquivos localizados dentro de um diretório compactado chamado *lauxen.tar.gz*¹. Através de um *script* de instalação, este diretório é descompactado e os arquivos contendo o código-fonte são compilados. A compilação dá origem há 3 arquivos executáveis, *scanner*, *monitor* e *lauxen*. Todos são copiados para o diretório “*/usr/sbin*”.

Terminada esta etapa, é necessário editar o *Arquivo de Configuração lauxen.conf*, respeitando o formato detalhado na seção 4.4. Este arquivo está dentro do diretório “*/etc/lauxen*”, criado durante a instalação. Com a configuração realizada e assumindo que Xen esteja executando, o comando “*lauxen launch*” realiza o lançamento ordenado das máquinas virtuais.

Para tornar a inicialização de *LauXen* automática, é necessário que o *script lauxend* seja executado durante o *boot* do sistema. No caso da distribuição Gentoo Linux, basta

¹Arquivo disponível em <http://www.inf.ufsm.br/~linck/tcc/lauxen.tar.gz>

copiar este arquivo para o diretório “*/etc/init.d/*” e executar o comando “*rc-update add lauxend default*”. Este procedimento pode variar de uma distribuição Linux para outra. O motivo pelo qual isto não foi automatizado se deve à variedade de distribuições Linux, onde não há uma padronização sobre como fazer um aplicativo ser iniciado durante o *boot* do sistema. Desta forma, o leitor deste trabalho saberá como agir em uma distribuição que não realize este procedimento desta forma. Além disso, *lauxend* está no formato utilizado pelo Gentoo Linux, em caso de uma distribuição diferente é necessário reformatar este arquivo.

Conforme citado na seção 4.3.6, a manutenção de um serviço pode levar à reinicialização da máquina virtual que o executa, caso o serviço permaneça inativo por um determinado tempo. Para evitar que isso aconteça, *LauXen* possui duas rotinas administrativas, *lmm pause* e *lmm unpause*, cujas funções são pausar e retomar a execução do *Monitor* passado como argumento, respectivamente. Deste modo, antes de executar operações que interrompam um determinado serviço de alguma MV por mais de 20 segundos, por exemplo, primeiramente paralize a execução de seu *Monitor*, e ao terminar desfaça esta operação.

A adição de novas máquinas virtuais, mesmo depois que as existentes já estiverem executando, também é permitida. Este processo consiste em adicionar as informações relacionadas à nova máquina virtual no *Arquivo de Configuração*. Feito isso, *LauXen* possui uma rotina de validação da nova configuração, chamada *lauxen rebuild*. Esta rotina realiza os testes necessários para garantir que a nova máquina virtual não traga problemas à configuração existente, como a criação de um ciclo entre as MVs existentes e ela. No caso de problemas não serem encontrados, a execução do comando *lauxen launch* realiza o lançamento de todas as máquinas virtuais que não estiverem executando, entretanto, é possível lançar somente a máquina nova, passando seu nome como argumento para este comando.

Para desligar as máquinas virtuais de forma ordenada, basta executar o comando “*lauxen shutdown*”. O desligamento de uma única máquina virtual também é permitido, bastando apenas passar o seu nome como argumento para esta função. É importante lembrar que no caso de outras máquinas virtuais dependerem de algum serviço oferecido pela máquina desligada, estas ficarão pausadas até que a mesma seja relançada.