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

BALANCEAMENTO DE CARGA DE SERVIDORES VIRTUALIZADOS

TRABALHO DE GRADUAÇÃO

Gustavo Foletto Mora

Santa Maria, RS, Brasil

2008

BALANCEAMENTO DE CARGA DE SERVIDORES VIRTUALIZADOS

por

Gustavo Foletto Mora

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. Benhur de Oliveira Stein

Trabalho de Graduação N. 268

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

BALANCEAMENTO DE CARGA DE SERVIDORES VIRTUALIZADOS

elaborado por
Gustavo Foletto Mora

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

COMISSÃO EXAMINADORA:

Prof. Benhur de Oliveira Stein
(Presidente/Orientador)

Profª Andrea Schwertner Charão (UFSM)

Prof. Antonio Marcos de Oliveira Candia (UFSM)

Santa Maria, 16 de Dezembro de 2008.

AGRADECIMENTOS

Agradeço:

À minha família que me apoiou durante o tempo que eu estive fazendo este trabalho e este curso.

Aos colegas e amigos que fiz durante estes quatro anos de faculdade.

Ao professor Benhur de Oliveira Stein, que me orientou neste trabalho e em tantos outros, e, sempre quando eu tinha dúvidas, ele as respondeu.

À professora Andrea Schwertner Charão e o professor Antonio Marcos de Oliveira Candia que humildemente aceitaram fazer parte da banca deste trabalho.

Aos demais professores, que ajudaram a formar meu conhecimento.

Muito obrigado a todos.

RESUMO

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

BALANCEAMENTO DE CARGA DE SERVIDORES VIRTUALIZADOS
Autor: Gustavo Foletto Mora
Orientador: Prof. Benhur de Oliveira Stein
Local e data da defesa: Santa Maria, 16 de Dezembro de 2008.

O poder computacional presente nos servidores atuais não costuma ser totalmente utilizado pelos serviços que neles são executados. A preocupação com a segurança levou os administradores a designar poucos serviços por servidor, e com isso os servidores acabam tornando-se subutilizados. Para resolver este problema utiliza-se a virtualização, capaz de disponibilizar uma boa forma de obter o isolamento de serviços juntamente com a possibilidade de ocupar melhor os dispositivos de *hardware*. Em ambientes que utilizam máquinas virtuais para execução de serviços, a distribuição de máquinas virtuais por servidor real é geralmente feita pelo administrador do sistema de forma manual, ficando geralmente estática, mesmo que, por algum motivo, um servidor real esteja mais sobrecarregado que os outros. Isso acaba tornando-se um problema para os administradores, que devem dividir as máquinas virtuais observando vários aspectos como: memória utilizada, tempo de CPU utilizado, utilização do disco e também da rede. Este trabalho desenvolve um sistema de equilíbrio de carga automatizado, que é capaz de monitorar os recursos dos servidores reais, e melhorar o uso destes recursos. Também apresenta um relato da funcionalidade do sistema, ferramentas usadas, e alguns testes que foram realizados.

Palavras-chave: Balanceamento de carga; monitor de máquinas virtuais; máquinas virtuais; monitoramento.

ABSTRACT

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

LOAD BALANCING OF VIRTUALIZED SERVERS
Author: Gustavo Foletto Mora
Advisor: Prof. Benhur de Oliveira Stein

The computational power present in servers today typically is not fully used by its services. Concerns about security have led administrators to designate fewer services for each server, ending up with under-used servers. Virtualization is a way to solve this problem, providing a good way to get the isolation of services and the opportunity to better use the hardware devices. In environments that use virtual machines to run services, the distribution of the virtual machines on real servers is usually manually made by the system administrator, generally in a static way, even if for some reason a real server becomes more loaded than the others. This has become a problem for administrators, who must initialize the virtual machines observing various aspects such as memory consumption, the CPU time used, disk and network usage. This work proposes the development of a system that automatically balances the load of real servers, monitoring the resources of actual servers, and improving their usage.

Keywords: load balance, virtual machines, virtual machine monitor, management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MMV	Monitor de Máquinas virtuais
MV	Máquina Virtual
CPU	Central Processing Unit
API	Application Programming Interface
TLS	Transport Layer Security
NFS	Network File System
LSC	Laboratório de Sistemas de Computação
SSL	Secure Sockets Layer
SSH	Secure Shell
TCP	Transmission Control Protocol
IP	Internet Protocol

SUMÁRIO

1 INTRODUÇÃO

O poder computacional presente nos servidores atuais não costuma ser totalmente utilizado pelos serviços que neles são executados. A preocupação com a segurança levou os administradores a designar poucos ou somente um único serviço por servidor, o que facilita a identificação de falhas de *software* e diminui os danos que tais falhas possam causar. No caso da presença de vários serviços, todos eles seriam comprometidos em caso de falha ou invasão no servidor. Com poucos serviços executados, menor a carga no servidor, e, conseqüentemente, o sistema torna-se subutilizado.

Com o aumento do número de serviços oferecidos, maior é o número de servidores, e, conseqüentemente, maior o trabalho de administração em uma rede. Os gastos com infra-estrutura, refrigeração e energia também aumentam, exigindo medidas que minimizem os custos e reduzam o trabalho da equipe responsável pelos servidores. Uma das soluções para estes problemas envolve o uso de virtualização de servidores, na qual várias máquinas virtuais executam sobre um *hardware* em comum e compartilham os recursos computacionais da máquina real, de forma isolada, como se estivessem executando em máquinas reais e distintas (BARHAM et al., 2003).

A virtualização no nível de sistemas operacionais utiliza uma camada de *software* responsável pelo compartilhamento, gerenciamento e alocação de recursos, chamada de Monitor de Máquinas Virtuais (MMV). Um MMV provê a abstração necessária para que as máquinas virtuais possam funcionar sobre ele, sem que percebam que há algo entre elas e o *hardware*. Alguns exemplos de MMV são: Xen (Xen, 2008), VMWare (VMware Inc., 2008a), QEMU (QEMU, 2008), OpenVZ (OpenVZ, 2008). Todos eles diferem conceitualmente em alguns pontos.

Em ambientes que utilizam máquinas virtuais para execução de serviços, é comum que haja um conjunto de servidores reais, e que cada um deles possua várias máquinas virtuais

executando diversos serviços. A distribuição de máquinas virtuais por servidores reais é geralmente feita pelo administrador do sistema de forma manual, ficando geralmente estática, mesmo que, por algum motivo, um servidor real se torne mais sobrecarregado que os outros. Isso acaba tornando-se um problema para os administradores, que devem dividir as máquinas virtuais observando vários aspectos como: memória utilizada, tempo de CPU utilizada, utilização do disco e também da rede.

Com o uso das máquinas virtuais torna-se evidente a necessidade de um balanceador de cargas que monitore a utilização dos recursos de hardware e, com isso, melhore sua distribuição entre as diversas máquinas virtuais usuárias desses recursos. O Núcleo de Ciência da Computação da UFSM possui diversos servidores virtualizados, e já foi observado que as máquinas virtuais, dependendo do uso dos serviços que elas provêm, acabam sobrecarregando parte do sistema.

Nesse sentido, este trabalho propôs o desenvolvimento de um sistema de balanceamento de carga automatizado, que fosse capaz de monitorar os recursos dos servidores reais como um todo e melhorar o uso destes recursos, a fim de evitar que algum servidor fique sobrecarregado enquanto outros estão sub-utilizados. O equilíbrio de carga é realizado migrando-se máquinas virtuais de máquinas reais mais carregadas para as menos carregadas. No desenvolvimento deste projeto, foram utilizados como ambiente de testes três servidores do Laboratório de Sistemas de Computação (LSC) do Curso de Graduação em Ciência da Computação da UFSM. Foi utilizado o MMV Xen, que oferece suporte à migração de máquinas virtuais.

O restante deste texto está organizado da seguinte forma: o capítulo 2 faz uma revisão sobre as tecnologias úteis para o entendimento do trabalho em geral; o capítulo 3 apresenta o projeto da ferramenta; o capítulo 4 apresenta uma descrição da implementação do trabalho; o capítulo 5 apresenta os testes realizados para validar a ferramenta; e por fim o capítulo 6 apresenta uma breve conclusão do trabalho.

2 CONCEITOS E TECNOLOGIAS

A virtualização vem sendo amplamente utilizada em diversas áreas, para diversos usos. É uma tecnologia bem difundida, principalmente por fatores como: isolabilidade de servidores, independência de *hardware*, facilidade de administração e capacidade de personalização (GUILHERME P. KOSLOVSKI, 2006).

Este capítulo aborda os conceitos e ferramentas relacionadas com a virtualização.

2.1 Máquina Virtual

Máquina virtual é o nome dado a uma implementação eficiente e isolada de uma máquina real. Ao invés do sistema operacional executar diretamente sobre o *hardware*, ele executa sobre uma camada de *software*, chamada monitor de máquinas virtuais. Todos os dispositivos de uma máquina virtual, como memória, processador, disco e rede, são virtuais e mapeados nos dispositivos reais pelo monitor de máquinas virtuais. Uma máquina virtual comporta-se como uma máquina real.

As principais vantagens da utilização de máquinas virtuais são:

- múltiplos ambientes podem co-existir no mesmo *hardware*, com um forte isolamento entre eles;
- facilidade de criação e configuração de um sistema novo;
- facilidade de recuperação de um sistema, no caso de falhas.

Em servidores, geralmente são executados múltiplos serviços. Isso freqüentemente causa algum tipo de interferência entre estes serviços. Um bom método para evitar esse problema é a utilização de máquinas virtuais, onde cada serviço que poderia causar alguma interferência executa em uma máquina virtual diferente, provendo uma melhor tolerância a falhas do sistema como um todo.

2.2 Monitores de Máquinas Virtuais

Monitores de máquinas virtuais são implementados por meio de uma camada de *software* que fica entre o *hardware* e a máquina virtual. Eles administram os recursos de *hardware* e permitem que diversos sistemas operacionais executem sobre eles, gerando uma transparência para as máquinas virtuais hospedeiras. Entre os monitores de máquinas virtuais estão:

- Xen (Xen, 2008) é uma plataforma de virtualização livre para as arquiteturas x86, x86-64, IA-32, IA-64 e PowerPC. Foi originalmente um projeto de pesquisa na Universidade de Cambridge, liderado por Ian Pratt, fundador da XenSource, e posteriormente foi adquirida pela Citrix System Inc. Foi desenvolvido com o modelo de virtualização clássica, em que o monitor executa diretamente sobre o *hardware*, e, por isso, possui um bom desempenho em relação a outros monitores de máquinas virtuais.

- Vmware (VMware Inc., 2008a) é um *software* proprietário desenvolvido pela empresa VMware Inc. Possui uma versão gratuita para testes a qual utiliza a técnica de virtualização hospedada, em que o MMV executa como um aplicativo sobre um sistema operacional. Porém possui também outras versões que utilizam a técnica de virtualização clássica, e conseqüentemente possuem um melhor desempenho.

- QEMU (QEMU, 2008) utiliza a técnica de virtualização hospedada e é um *software* livre desenvolvido por Fabrice Bellard.

- OpenVZ (OpenVZ, 2008) é um *software* livre desenvolvido pela empresa Parallels Inc e implementa o esquema de virtualização de sistema operacional, no qual o núcleo do sistema operacional permite múltiplas instâncias isoladas do espaço de usuário (*user-space*), em vez de somente uma.

2.3 Balanceamento de Carga de Servidores

Quando há diversos servidores reais, cada um deles executando diversas máquinas virtuais, fica evidente a necessidade de um balanceamento de máquinas virtuais, a fim de equilibrar o desempenho entre todos os servidores.

Um equilibrador de cargas deve monitorar a utilização do processador, memória, rede

e disco de um conjunto de servidores, e ser capaz de analisar estes dados a fim de não deixar um servidor mais carregado, enquanto há servidores sendo subutilizados. Para redistribuir a carga, utiliza-se a migração de máquinas virtuais de um servidor real para outro. O processo de migração consiste em transferir, entre computadores conectados, uma máquina virtual em execução. Isso é feito de maneira que não afeta demasiadamente a execução desta máquina virtual.

Existe uma biblioteca chamada Libvirt, que facilita a manipulação das máquinas virtuais e tem como objetivo prover uma interface única para administração de domínios em um ou mais nós. Essa biblioteca possui uma interface que se comunica com diversos tipos de monitores de máquinas virtuais. Ela possui chamadas que facilitam a interação com as máquinas virtuais, como por exemplo para migração, captura de dados, inicialização e desligamento.

2.4 Libvirt

A Libvirt (Libvirt, 2008) é uma API para C que facilita a manipulação de máquinas virtuais, e tem como objetivo prover uma interface única para administração de domínios em um nó.

Possui suporte para Xen, QEMU, KVM (KVM, 2008) e OpenVZ. Pode ser usada remotamente, juntamente com o protocolo TLS.

Como a Libvirt suporta diferentes tipos de monitores de máquinas virtuais, é necessário uma forma de especificar o *driver* que a conexão deve utilizar. E no caso de uma conexão remota, precisa-se informar a localização do servidor ao qual será conectado. Para isso a Libvirt utiliza URIs como as que são utilizadas na Web. As URIs seguem o formato: *driver://host/*.

2.5 Acesso Remoto

A Libvirt permite o acesso aos monitores em máquinas remotas, através de conexões autenticadas e encriptadas.

Para isso utiliza TLS, que significa *Transport Layer Security* (Wikipedia, 2008), e é o protocolo sucessor do SSL (Webstart, 1994). Eles são protocolos de criptografia que garantem a segurança das comunicações feitas com TCP/IP na Internet. São muito utilizados em aplicações voltadas para a Internet como por exemplo navegadores Web,

correio eletrônico e mensagens instantâneas. O TLS foi baseado no SSL.

O protocolo garante a privacidade a integridade dos dados entre duas aplicações que estão comunicando-se pela Internet.

Para o acesso remoto, é preciso que o servidor que será conectado esteja rodando o *daemon* libvirtd. Para conectar o servidor remoto deve se usar uma URI apropriada com o *hypervisor* e o transportador.

Os transportadores podem ser: TLS, unix, SSH e TCP. O transportador padrão, se nenhum for especificado é o TLS.

Para o uso do TLS é necessário configurar um certificado de autoridade (CA), tanto para o servidor como para o cliente.

2.6 Trabalhos Relacionados

O problema de equilíbrio de carga entre monitores de máquinas virtuais é tratado por outros trabalhos. Todos eles possuem algum tipo de dependência de tecnologia proprietária, e isso é um grande empecilho para uma possível utilização em instituições com limitações de recursos. Neste trabalho foram utilizadas ferramentas livres de custo, e com funcionalidades semelhantes às ferramentas proprietárias. Seguem abaixo alguns trabalhos semelhantes a este.

2.6.1 VMware Vmotion

É um *software* proprietário produzido pela empresa VMware inc. (VMware Inc., 2008b), e suporta apenas o monitor VMware ESX.

Este *software* é capaz de fazer a migração de máquinas virtuais automaticamente, de acordo com a carga da CPU e da rede.

2.6.2 LBVM

LBVM é um acrônimo para *Load Balancing of Virtual Machines* (LBVM, 2008) e significa Balanceamento de carga de Máquinas Virtuais.

Consiste em vários *scripts* que permitem o balanceamento de carga de máquinas virtuais. Suporta Xen e OpenVZ.

Permite o compartilhamento de máquinas virtuais entre servidores físicos. LBM, LB LOG e LB MONITOR são os *scripts* principais.

O *script* LBM é a interface de gerenciamento para o balanceador de carga. É usado

também para ver os registros das migrações ocorridas.

O *script* LB LOG é executado regularmente em cada servidor através do *cron*, e é utilizado para coletar dados. Os dados coletados são armazenados em um local compartilhado por todos os servidores, e são analisados pelo balanceador.

O *script* LB MONITOR utiliza diversos algoritmos para decidir qual máquina virtual será migrada.

Apesar de ser um *software* livre, ele funciona apenas com o *Red Hat Cluster Suite*, que é um produto da empresa Red Hat e, também, é necessária a utilização de locais compartilhados de armazenamento de dados para as máquinas virtuais, tais como iSCSI, SAN ou DRBD.

2.6.3 Virtual Iron LiveCapacity

É um outro tipo de *software* proprietário para o gerenciamento e balanceamento de carga (Virtual Iron, 2007). Este *software* é parte do *Virtual Iron 3.1 Enterprise Edition*, e suporta apenas essa tecnologia.

2.6.4 ZENworks Virtual Machine Management

É um *software* proprietário produzido pela Novell (Novell, 2008), que gerencia e faz o balanceamento de máquinas virtuais. Suporta virtualização do VMware, Windows e Xen.

3 PROJETO DA FERRAMENTA

Visando alcançar os objetivos apresentados, inicialmente foi realizada uma avaliação de como seria feito o balanceamento de carga dos servidores. A partir desta avaliação decidiu-se que seria monitorado o uso do processador, disco, memória e rede.

Esta ferramenta monitora os servidores à distância, coletando dados das máquinas virtuais e analisando-os. Quando mostrar-se necessário, reorganizará a distribuição das máquinas virtuais dos servidores. O monitoramento acontece em intervalos de tempos pré-definidos. São gerados *logs* dos dados coletados, e também de eventos acontecidos.

Para distribuição das máquinas virtuais, existe uma técnica chamada de migração "ao vivo" (CLARK et al., 2005), com a qual é possível que uma máquina virtual, em tempo de execução, mude de servidor. A migração das máquinas virtuais acontece de forma transparente, e não prejudica o sistema como um todo. O monitor de máquinas virtuais Xen, utilizado pelo Núcleo de Ciência da Computação, possui este suporte.

A ferramenta projetada consiste basicamente de 3 componentes básicos: coleta de dados, análise dos dados e migração das máquinas virtuais. A coleta de dados, como o nome sugere, captura os dados do sistema, e os armazena. O módulo de análise processa as informações armazenadas e, a partir disso, toma decisões chamando o módulo de migração para efetuar possíveis migrações de máquinas virtuais. A figura 3.1 apresenta um diagrama da disposição destes componentes, que são descritos nas seções seguintes.

3.1 Coleta de dados

Para a coleta de dados, foi utilizada a biblioteca Libvirt, que tem chamadas que se comunicam com o *daemon* dos servidores e retornam alguns dados das máquinas virtuais. Estes dados são: tempo de CPU, memória total, memória utilizada, utilização da rede e utilização de disco. Isso facilitou a fase de coleta de dados, pois não foi necessário

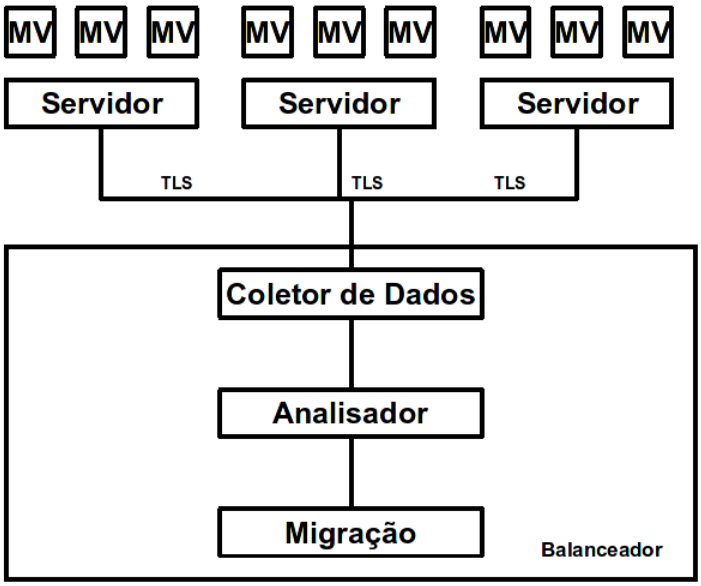

Figura 3.1: Diagrama da disposição dos componentes básicos da ferramenta

preocupar-se com a comunicação com os servidores.

O programa comunica-se com o *daemon* da Libvirt de cada servidor utilizando o protocolo TLS. Para o funcionamento do TLS é necessária a configuração de certificados de segurança para cada servidor físico.

Os dados são coletados em intervalos de tempos arbitrários, e as medições são armazenadas em disco. Estão sendo armazenados dados da utilização do processador, memória, disco e rede. A figura 3.2 ilustra a os componentes da coleta de dados. O componente *Sistema* é o que contém todos os outros componentes. O componente *Host* armazena informações sobre um servidor real. O módulo *MaquinaVirtual* encapsula uma MV, e os módulos *MemCPU*, *Rede* e *Disco* armazenam os dados coletados.

Para cada dado armazenado, é calculada uma média, que é armazenada em memória juntamente com as últimas medições, que formam a média. O número de medições armazenadas em memória é definido na inicialização do programa. Este número é muito importante pois afeta diretamente o cálculo da média, e conseqüentemente a análise dos dados.

3.2 Análise dos dados

A análise também é realizada em intervalos de tempos definidos, porém o tempo entre as análises pode ser maior que o de coleta. Apesar de estarem sendo armazenados

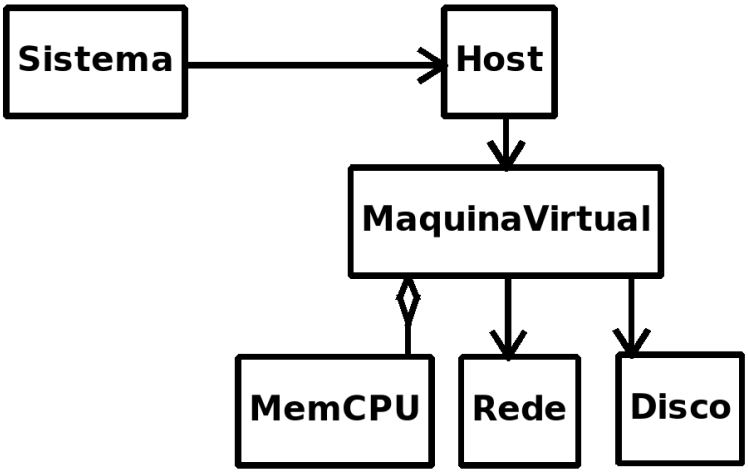

Figura 3.2: Diagrama dos componentes da coleta de dados

dados do processador, disco, rede e memória, a análise está sendo realizada **apenas** para a carga do processador. Isto deve-se ao fato de que uma análise de mais tipos de dados, simultaneamente, torna-se mais complexa e então, por isso, decidiu-se focar apenas no processador.

A análise é dividida em duas partes: o cálculo do estado e a procura de possíveis máquinas virtuais a serem migradas. O cálculo do estado é realizado para saber se o sistema está equilibrado ou não. Se o sistema estiver desequilibrado, acontece a procura de máquinas virtuais que podem ser migradas. Segue abaixo uma descrição mais detalhada destes dois módulos.

3.2.1 Cálculo do estado

Primeiramente acontece a análise para verificar se o sistema está equilibrado ou não. Para calcular o estado do sistema, é calculada a porcentagem de utilização da CPU de cada servidor físico, que varia de 0 ao número de processadores multiplicado por 100. Por exemplo, para um servidor com 8 processadores, a porcentagem de carga varia de 0 a 800. Este cálculo utiliza a média mencionada anteriormente na fase de coleta de dados.

Após isso é verificado o desequilíbrio entre esses servidores. Se o maior desequilíbrio entre os servidores estiver maior que um limite definido, e o tempo que verifica-se este desequilíbrio for maior que um tempo de desequilíbrio definido, então o sistema entra em estado de desequilíbrio. O algoritmo de cálculo de estado do sistema está representado no diagrama de estados da figura 3.3.

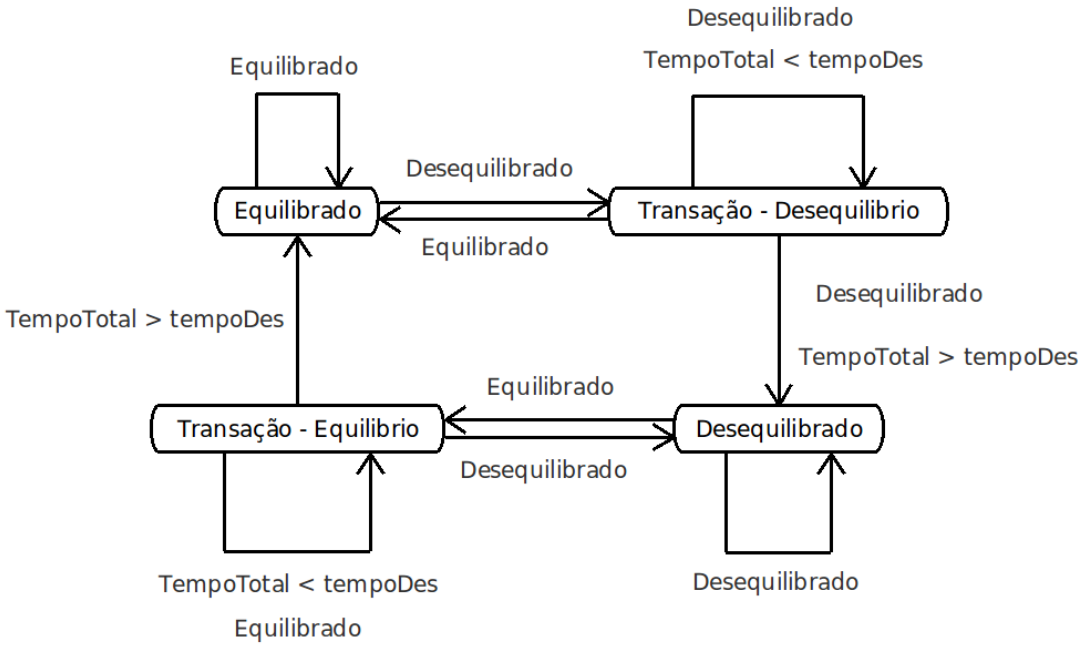

Figura 3.3: Diagrama de estados do cálculo do estado do sistema

3.2.2 Seleção de máquinas virtuais

Se o sistema está equilibrado, não acontece nada, porém se o sistema estiver desequilibrado, é passado para o próximo tipo de análise, que é descobrir qual máquina virtual e para onde deverá ser migrada. Como esta implementação está apenas focando o uso do processador, os outros dados de carga são desconsiderados.

O cálculo para resolver como será efetuada a migração é simples: o servidor que terá sua máquina virtual migrada é o que tiver maior carga; o servidor que receberá a máquina virtual migrada é o que possuir a menor carga; a máquina virtual escolhida é a que possuir a carga mais próxima do desequilíbrio entre o processador de maior carga e o de menor carga, semelhante ao algoritmo de alocação de memória em sistemas operacionais *"best fit"* (TANENBAUM, 2001). A análise limita-se em escolher apenas uma máquina virtual por vez.

Este algoritmo funciona, porém ele pode apresentar um problema de "ping-pong" de máquinas virtuais. Isto acontece quando uma máquina virtual fica alternando de servidor. Para solucionar este problema, a máquina virtual somente será escolhida se ela não passar o desequilíbrio para o outro servidor. Isto limita um pouco as escolhas, porém consegue evitar efetivamente que máquinas sejam continuamente migradas entre os servidores.

3.3 Migração

A partir da identificação da máquina virtual e do servidor de destino, deverá ser finalmente realizada a migração.

O Xen possui suporte a migração, e para ser possível que ela ocorra, a imagem do disco da máquina virtual deve ser acessível para todos os servidores. Isto é possível com a utilização do *Network File System protocol* (NFS) (SHEPLER et al., 2003). A migração, por mais que seja um processo transparente ao usuário, possui um certo *overhead*, então a migração de uma máquina virtual deve ser feita com muito cuidado, evitando migrações desnecessárias. Por este motivo, o algoritmo da análise foi implementado de forma conservadora.

A migração não é feita diretamente com chamadas do Xen. Ela é feita com a ajuda a Libvirt, que possui suporte para tal. Ao fim da migração o processo todo recomeça.

4 IMPLEMENTAÇÃO DA FERRAMENTA

Conforme apresentado anteriormente, foi utilizada a biblioteca Libvirt em conjunto com a linguagem de programação C++ (STROUSTRUP, 2000). Inicialmente este projeto estava sendo implementado com a linguagem de programação C, mas com o crescimento desordenado do código, preferiu-se fazer a troca para a linguagem C++, a qual acaba forçando a modularização do código. Esta troca, com o andamento do projeto, mostrou-se muito adequada, pois além de obrigar uma modularização, esta linguagem possui algumas estruturas que ajudam na implementação. A versão final desta ferramenta é composta de 17 classes e mais 2 módulos auxiliares.

4.1 Configuração

Para uma maior simplicidade do programa, foram implementadas as classes *BalanceConfigs* e *ConfigParser* que carregam os dados de configuração do programa a cada inicialização. Este arquivo de configuração segue este padrão: *configuração* = *opção*. Uma por linha. Existem configurações para: adicionar *host*, intervalo de medição, intervalo de análise, modo somente coleta de dados, tamanho do *buffer* de dados, salvar *log*, porcentagem de desequilíbrio e tempo de desequilíbrio. Essas configurações são utilizadas em diversas partes do programa. Também foi implementado um pequeno *shell* que serve para fazer mudanças de configurações em tempo de execução, e também para a invocação de comandos.

Utilizou-se *threads* na implementação deste trabalho, as quais garantem uma execução em paralelo do programa. Em locais que haviam necessidade de execução seqüencial, foram usados *mutexes* para garantir exclusão mútua.

Existe uma *thread* para coleta de dados, uma para análise dos dados, e outra para o *shell*. Na coleta, existe uma *thread* para cada máquina virtual. Então, o número total de

threads do programa será: três, mais o número de máquinas virtuais do sistema inteiro.

4.2 Coleta de dados

Para a parte de coleta de dados foram implementadas as seguintes as classes: *Sistema, Host, MaquinaVirtual, Disco, Rede, MemCPU e Medida*. Essas classes são responsáveis pela conexão, comunicação e armazenamento das informações coletadas. Todas as classes citadas acima possuem dados privados e sendo assim, são providos métodos de acesso aos dados, e algumas possuem ainda métodos para modificação dos mesmos. A figura 4.1 ilustra o diagrama de classes simplificado da parte de coleta de dados.

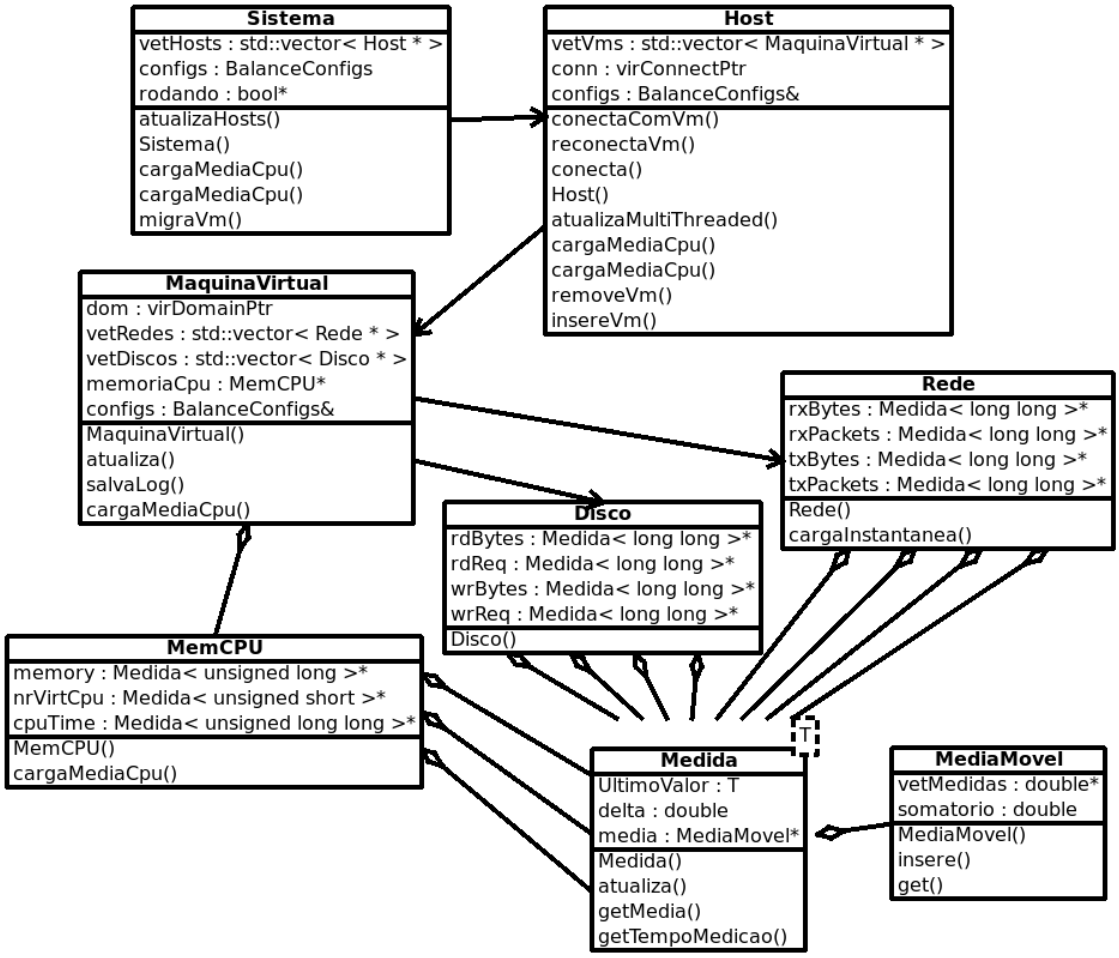

Figura 4.1: Diagrama de classes da parte de coleta de dados

Segue abaixo uma descrição das classes implementadas:

A classe *Sistema* é a classe principal da coleta de dados. Ela contém um vetor de elementos do tipo *Host*. O construtor da classe inicializa todos os servidores que estiverem definidos no arquivo de configuração. O método *atualizaHosts* encarrega-se de atualizar

de tempos em tempos os seus *Hosts*.

A classe *Host* encapsula um servidor físico e possui um vetor de elementos do tipo *MaquinaVirtual*. Na inicialização cada *Host* inicializa todas as suas máquinas virtuais. Possui também métodos de acesso, inserção, remoção e atualização de máquinas virtuais.

A classe *MaquinaVirtual* encapsula uma máquina virtual. Pode ter vários discos e várias interfaces de rede, por isso possui vetores de elementos do tipo *Disco* e *Rede*. Possui também um elemento do tipo *MemCPU*. Na inicialização esta classe procura por todas interfaces de rede e de disco da máquina virtual, e as inicializa. Há também um método para salvar os dados em disco. O método *atualiza* é quem vai fazer as chamadas para atualizar os dados armazenados.

As classes *Disco, Rede* e *MemCPU* armazenam os dados respectivamente do disco, rede, memória e CPU. Possuem diversos objetos do tipo *Medida*.

A classe *Medida* encapsula uma medida realizada. É onde está armazenado o último dado lido. Possui um objeto do tipo *MediaMovel*, onde são armazenadas as últimas medidas, juntamente com a média delas.

O uso do TLS para a comunicação com a Libvirt estava gerando alguns problemas quando efetuavam-se muitas chamadas concorrentes. Para resolver isto, foram encapsuladas todas as chamadas que eram utilizadas em momentos que o programa executava concorrentemente, e foi utilizado um *mutex* para garantir que não houvesse comunicação concorrente. Isto está implementado no módulo *Comunicacao*.

4.3 Análise de dados

Para a parte de análise dos dados e migração de máquinas virtuais, foram implementadas as seguintes as classes: *Analisador, Estado, EstadoCpu, EstadoRede, EstadoGlobal* e *ProcuraMigracao*. Essas classes são responsáveis pela tomada de decisão do sistema. A figura 4.2 ilustra o diagrama de classes simplificado da parte de análise dos dados.

Segue abaixo uma descrição das classes implementadas:

A classe *Analisador* é a classe principal da parte de análise de dados. Possui um ponteiro para o objeto da *Sistema*, pois precisa acessar os dados para tomar decisões. Possui dois objetos do tipo *Estado*. O método *run* executa a análise de tempos em tempos.

A classe *Estado* implementa o cálculo do estado descrito na seção 3.2.1. As classes *EstadoCpu* e *EstadoRede* herdam da classe *Estado*, e calculam o estado para a CPU e

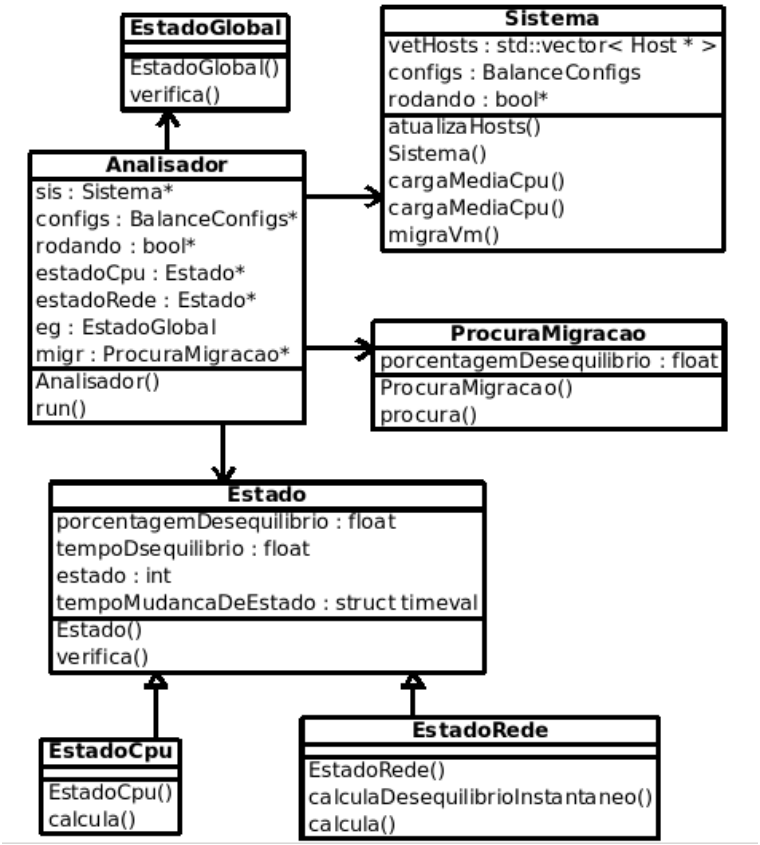

Figura 4.2: Diagrama de classes da parte de análise de dados

para a rede, respectivamente.

A classe *EstadoGlobal*, a partir do cálculo do estado realizado anteriormente, calcula se o sistema como um todo está desequilibrado ou não. Como este cálculo está sendo realizado apenas para a CPU, ele apenas repassa o resultado calculado pela classe *EstadoCpu*. Futuramente esta classe pode ser expandida para suportar um cálculo que considere os outros estados.

4.4 Migração

Para a parte de migração foi implementada a classe *ProcuraMigracao*, que após verificado o desequilíbrio, procura por uma migração que melhore o desequilíbrio do sistema. Baseia-se no algoritmo descrito na seção 3.2.2. Este cálculo também é realizado apenas para a CPU. Se for encontrado uma máquina virtual para ser migrada, será chamado o método *migraVm* contido na classe *Sistema*.

Com o fim da implementação resolveu-se dar um nome para a ferramenta, e nome escolhido foi *Vmbal*.

5 RESULTADOS

Todo projeto de *software* passa por uma fase de validação, na qual avalia-se se o que foi implementado está de acordo com os requisitos levantados.

Este capítulo apresenta os testes realizados com a ferramenta Vmbal.

5.1 Ambiente de Testes

Visando demonstrar o funcionamento prático da ferramenta Vmbal, foi criado um exemplo de utilização do *software*. Para a execução deste exemplo, utilizou-se três servidores do Laboratório de Sistemas de Computação com as seguintes configurações cada:

- **Processador**: Intel(R) Xeon(R) 2.00GHz x 8;
- **Memória Principal**: 8 GB;
- **Sistema Operacional**: GNU/Linux versão 2.6.20-xen-r6;
- **Xen**: 3.1.2.

O processo de instalação, configuração e como utilizar o Vmbal está descrito no apêndice A.

5.2 Testes

Tendo um ambiente de testes pronto, o passo seguinte foi a criação de um exemplo de utilização prático da ferramenta Vmbal.

Foram realizados dois tipos de testes: com máquinas virtuais com cargas estáticas, isto é, mantendo a porcentagem de utilização do processador constante, e com cargas dinâmicas, que variam de acordo com certos parâmetros. Para gerar a carga artificial, foi

implementado um programa capaz de gerar cargas de acordo com 4 parâmetros: carga do pico, carga do vale, tempo de carga do pico e tempo de carga do vale.

O objetivo do teste com carga estática é realizar um teste simples e controlado do funcionamento do Vmbal, pois sabe-se que as cargas estarão constantes em praticamente todo o experimento. O segundo teste, com carga dinâmica, tem por objetivo de adicionar uma variação nas cargas, e com isso dificultar um pouco a atuação do balanceador.

Em todos os testes, o sistema é considerado desequilibrado se a porcentagem de desequilíbrio valer mais de 60% e esse desequilíbrio for mantido por pelo menos 30 segundos. Esses valores foram escolhidos de modo que gerassem um grande número de migrações num tempo pequeno.

5.2.1 Carga estática

Foram realizados 2 tipos de testes com carga estática. Um com 8 e outro com 10 máquinas virtuais cada uma executando a 100% de CPU. Estes testes tiveram duração de apenas 5 minutos, pois este tempo foi suficiente para que o programa fizesse todas as migrações necessárias para equilibrar a carga entre os servidores, e a partir desse tempo o programa não faria mais nenhuma migração. O tamanho do *buffer* para as médias foi definido como 10 posições para ambos os casos.

Para os dois experimentos, foram realizados primeiramente testes de controle, com o programa Vmbal com modo de somente coleta de dados habilitado.

O objetivo do primeiro experimento é o de testar o programa da maneira mais simples possível, na qual as máquinas virtuais permanecem com a carga do processador constante em 100%.

A figura 5.1 apresenta o gráfico do teste de controle com 8 máquinas virtuais executando em apenas 1 servidor por 5 minutos.

Observa-se que a carga, como o esperado, manteve-se constante.

A figura 5.2 apresenta o gráfico de 8 máquinas virtuais executando em 3 servidores por 5 minutos, com o balanceamento ativado. O servidor *sgi3* foi inicializado com as 8 máquinas virtuais, e os outros com nenhuma.

Nota-se que há uma queda e um aumento repentino da porcentagem de utilização da CPU após uma migração. Isto deve-se ao fato de que a carga de um servidor é calculada a partir da soma das médias das cargas das máquinas virtuais. E quando uma MV passa

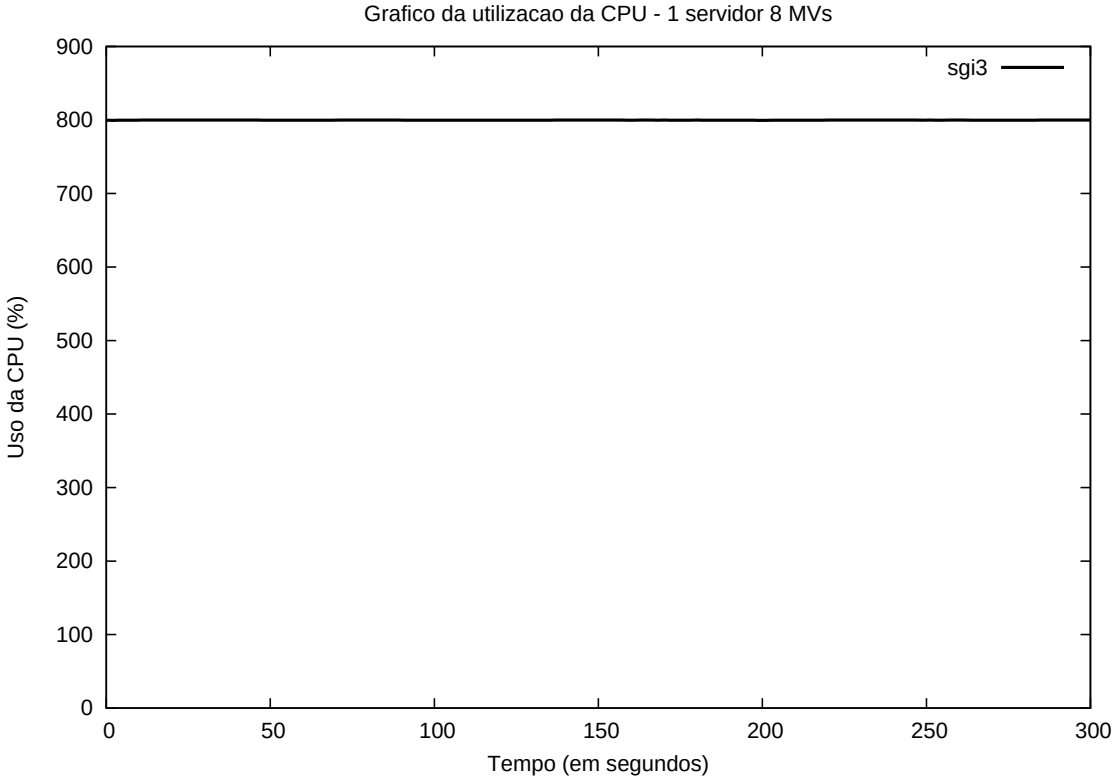

Figura 5.1: Carga de 1 servidor com 8 máquinas virtuais

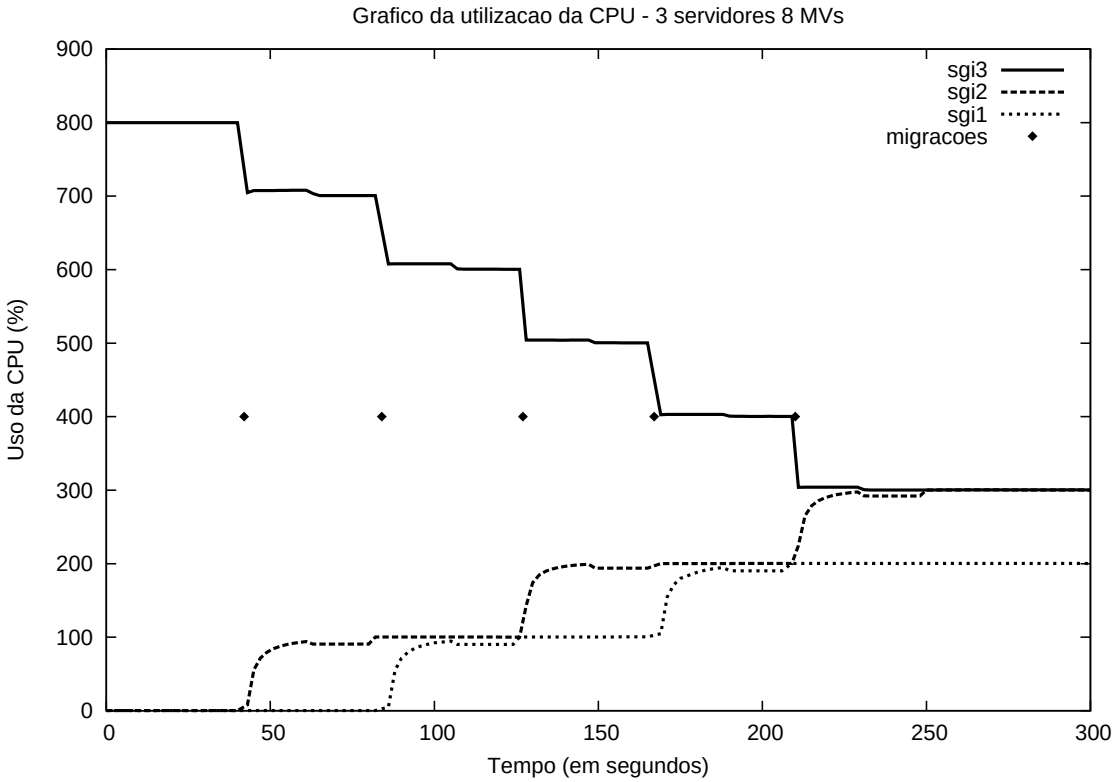

Figura 5.2: Carga de 3 servidores com 8 máquinas virtuais

de um servidor para outro, ela deixa de estar no cálculo do servidor de origem, e passa a fazer parte do servidor de destino. Pode-se observar também que o programa migrou as máquinas virtuais aos poucos, até atingir uma distribuição que fosse mais equilibrada. Neste caso foi uma distribuição ideal.

Os dois testes seguintes foram realizados com 10 máquinas virtuais. A figura 5.3 apresenta o gráfico com apenas 1 servidor e o balanceamento desligado. Como esperado, obteve-se uma carga constante.

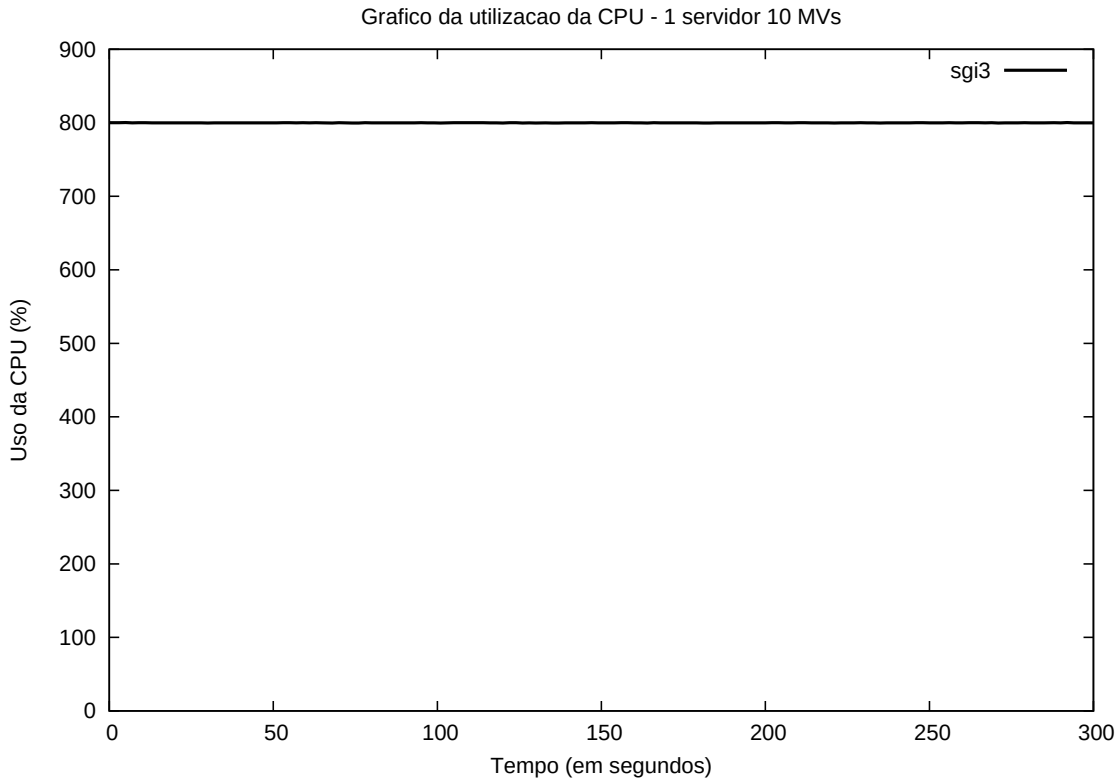

Figura 5.3: Carga de 1 servidor com 10 máquinas virtuais

A figura 5.4 apresenta o gráfico com 3 servidores e o balanceamento ligado. Logo percebe-se que também houve uma distribuição equilibrada no fim do teste.

O objetivo deste segundo experimento é aumentar um pouco a complexidade do teste, pois as cargas das máquinas virtuais variam levemente após uma migração.

Isto acontece porque inicialmente há 10 máquinas virtuais executando no servidor sgi3, e como ele possui apenas 8 processadores, só consegue processar a 800 %. Então antes da primeira migração, há uma divisão dessa capacidade de processamento entre todas as máquinas virtuais, e cada uma processa à 80 %. Após a primeira migração a máquina virtual migrada começa a executar à 100 % e as outras 9 ficam processando à

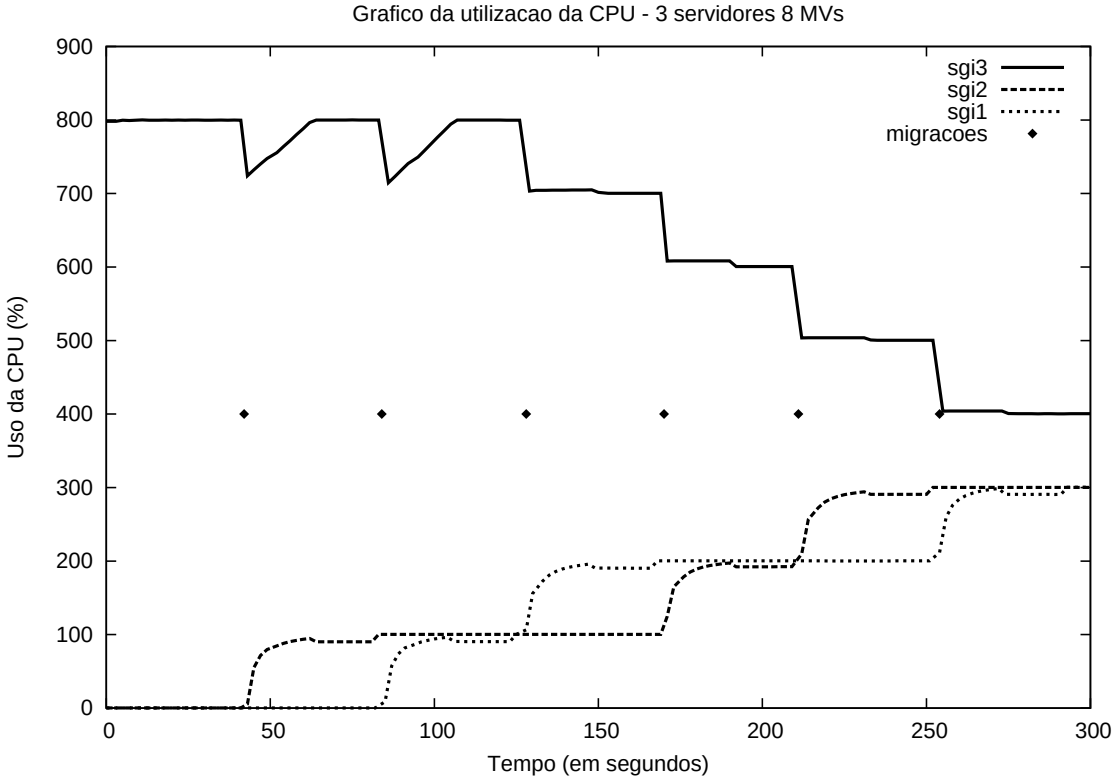

Figura 5.4: Carga de 3 servidores com 10 máquinas virtuais

88,88 %. Após a segunda migração restarão 8 máquinas virtuais no servidor sgi3 e a partir desse momento, todas as máquinas virtuais passarão a processar a 100 %. Com isso, percebe-se claramente que há um ganho de desempenho nesse experimento.

Nota-se na figura 5.4 que há alguns "dentes" na linha do servidor sgi3 antes da terceira migração, e isso deve-se ao fato que após uma migração há mudança na carga de cada máquina virtual desse servidor, e também por esse gráfico ser calculado com a soma das médias das cargas de cada uma das máquinas virtuais executada naquele servidor. Vale lembrar que esse é um comportamento esperado pois está sendo utilizando as médias das cargas, e não os valores instantâneos.

5.2.2 Carga dinâmica

Foram realizados testes com carga dinâmica, para verificar o comportamento do sistema com uma configuração menos estática. Realizou-se 2 testes com 10 máquinas virtuais.

O primeiro teste foi executado com um *buffer* de 10 posições, e teve duração de 12 minutos. Este teste tem como o objetivo aumentar a complexidade em comparação com os testes anteriores.

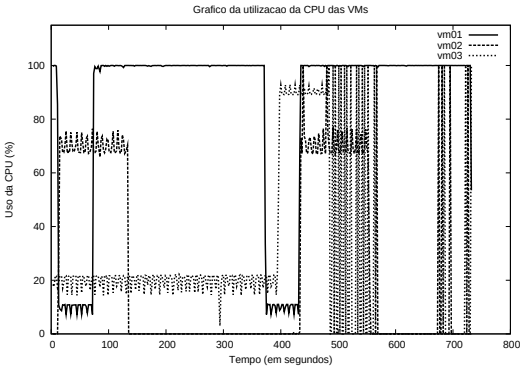

Figura 5.5: Carga das máquinas virtuais vm01, vm02 e vm03

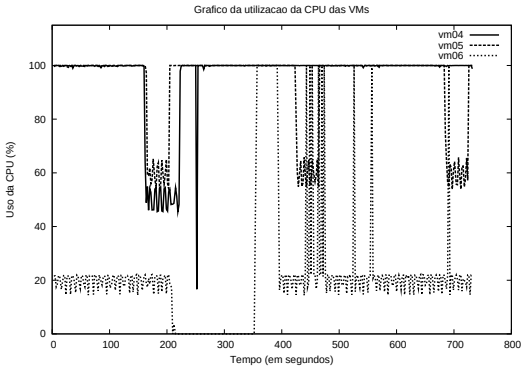

Figura 5.6: Carga das máquinas virtuais vm04, vm05 e vm06

As figuras 5.5, 5.6, 5.7, 5.8 e 5.9 representam o comportamento da carga da CPU das máquinas virtuais, e a figura 5.10 apresenta o gráfico da carga dos servidores. A ordem das migrações é representada pela tabela 5.1.

Tabela 5.1: Tabela das migrações do primeiro teste

segundo	VM	Origem	Destino
41	vm10	sgi3	sgi2
83	vm09	sgi3	sgi1
124	vm08	sgi3	sgi2
166	vm07	sgi3	sgi1
208	vm06	sgi3	sgi1
250	vm04	sgi3	sgi2
291	vm03	sgi3	sgi1
355	vm02	sgi3	sgi1
617	vm10	sgi2	sgi1

Percebe-se que as migrações aconteceram mais ou menos com o intervalo de 40 segundos, isto é devido ao tempo de desequilíbrio que foi fixado em 30 segundos, e o tempo de intervalo da análise de dados que foi fixado em 10 segundos. Com isso chega-se a este tempo de 40 segundos que representa o tempo de reação do sistema para este teste.

Nota-se que houve uma maior variação de cargas, e que em alguns momentos, o sistema ficou desbalanceado por mais tempo do que o previsto. Isto deve-se ao fato de ter ocorrido erros na hora de efetuar a migração. Estes erros não acontecem com frequência, porém após um certo tempo de execução eles têm a tendência de ocorrer. Eles ocorrem quando a Libvirt, por algum motivo desconhecido, não consegue migrar uma máquina virtual, aborta a migração e a chamada retorna um erro. Percebe-se que as máquinas virtuais vm02 e vm03 apresentaram problemas a partir do tempo de 480 segundos.

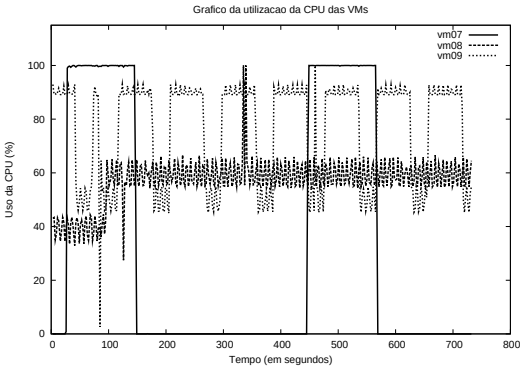

Figura 5.7: Carga das máquinas virtuais vm09, vm08 e vm07

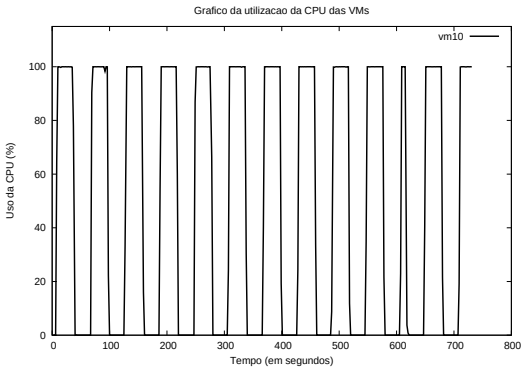

Figura 5.8: Carga da máquina virtual vm10

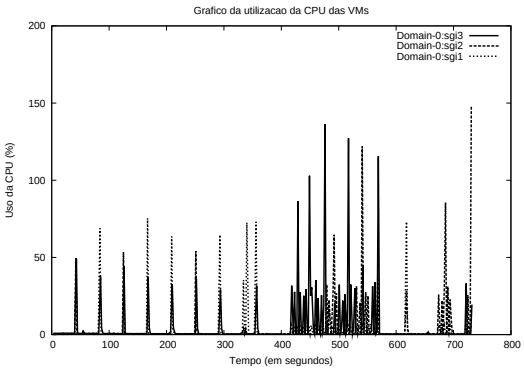

Figura 5.9: Carga das máquinas virtuais Domain0-sgi1, Domain0-sgi2 e Domain0-sgi3

O segundo teste foi executado com um *buffer* de 100 posições, e teve duração de 62 minutos. Este experimento tem como o objetivo aumentar ainda mais a complexidade em comparação com os testes anteriores e também testar o programa Vmbal com um *buffer* de médias 10 vezes maior que o dos outros testes.

As figuras 5.11, 5.12, 5.13, 5.14 e 5.15 representam o comportamento da carga da CPU das máquinas virtuais, e a figura 5.16 apresenta o gráfico da carga dos servidores. A ordem das migrações é representada pela tabela 5.2.

Também observa-se que após um certo tempo ocorreram erros na migração. As máquinas virtuais vm06 e vm07 apresentaram problemas a partir do tempo de 1700 segundos. Observa-se nos gráficos 5.9 e 5.15 o *overhead* apresentado nas migrações, representado pelos picos no gráfico. O *overhead* aconteceu também nos momentos que as migrações foram abortadas. O xen ao fazer uma migração tem que copiar a memória de um servidor para outro. Isso gera um certo *overhead* de CPU e também de rede para cada tentativa de migração.

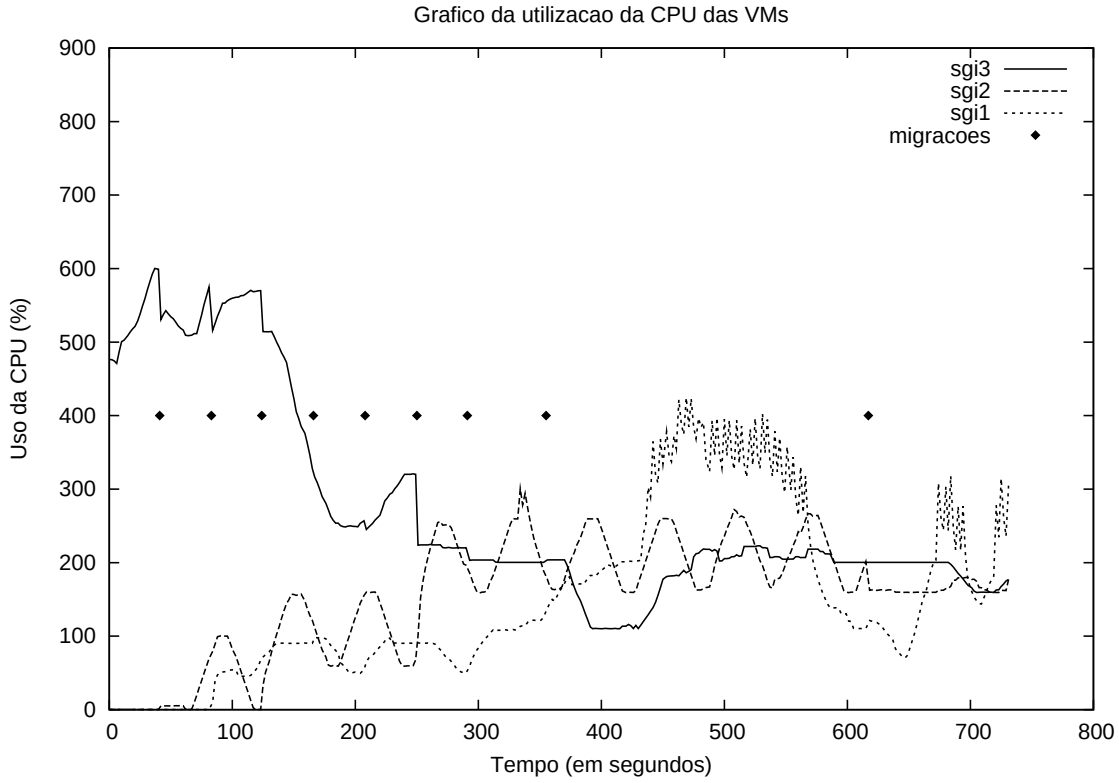

Figura 5.10: Carga de 3 servidores com 10 máquinas virtuais, e balanceador com *buffer* de 10 posições

Tabela 5.2: Tabela das migrações do segundo teste

segundo	VM	Origem	Destino
51	vm10	sgi3	sgi2
102	vm09	sgi3	sgi1
155	vm07	sgi3	sgi1
207	vm06	sgi3	sgi1
259	vm02	sgi3	sgi1
312	vm03	sgi3	sgi1
363	vm08	sgi3	sgi2
1585	vm01	sgi3	sgi1
1677	vm03	sgi1	sgi2
1729	vm06	sgi1	sgi2
1782	vm07	sgi1	sgi2

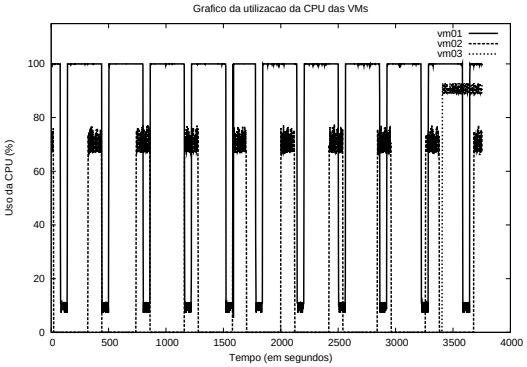

Figura 5.11: Carga das máquinas virtuais vm01, vm02 e vm03

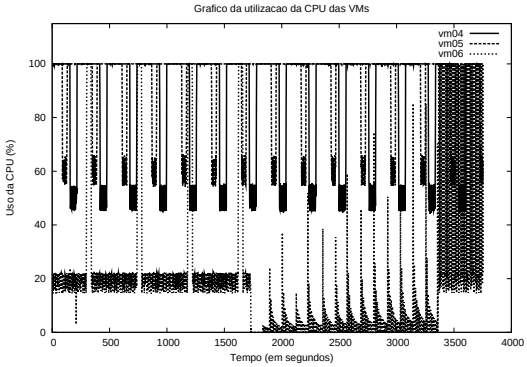

Figura 5.12: Carga das máquinas virtuais vm04, vm05 e vm06

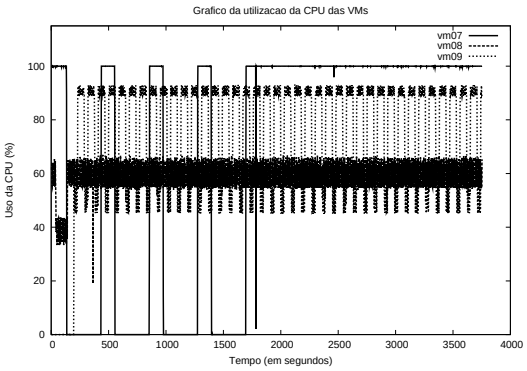

Figura 5.13: Carga das máquinas virtuais vm09, vm08 e vm07

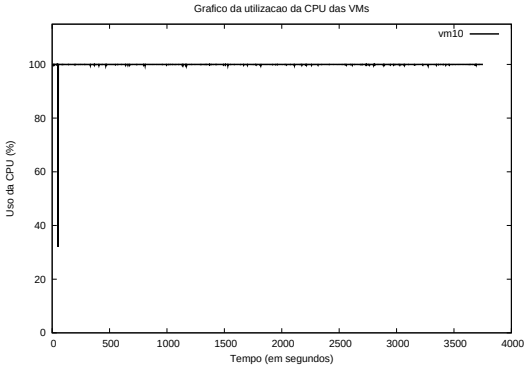

Figura 5.14: Carga da máquina virtual vm10

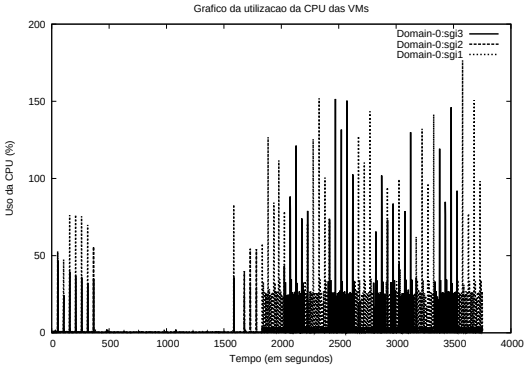

Figura 5.15: Carga das máquinas virtuais Domain0-sgi1, Domain0-sgi2 e Domain0-sgi3

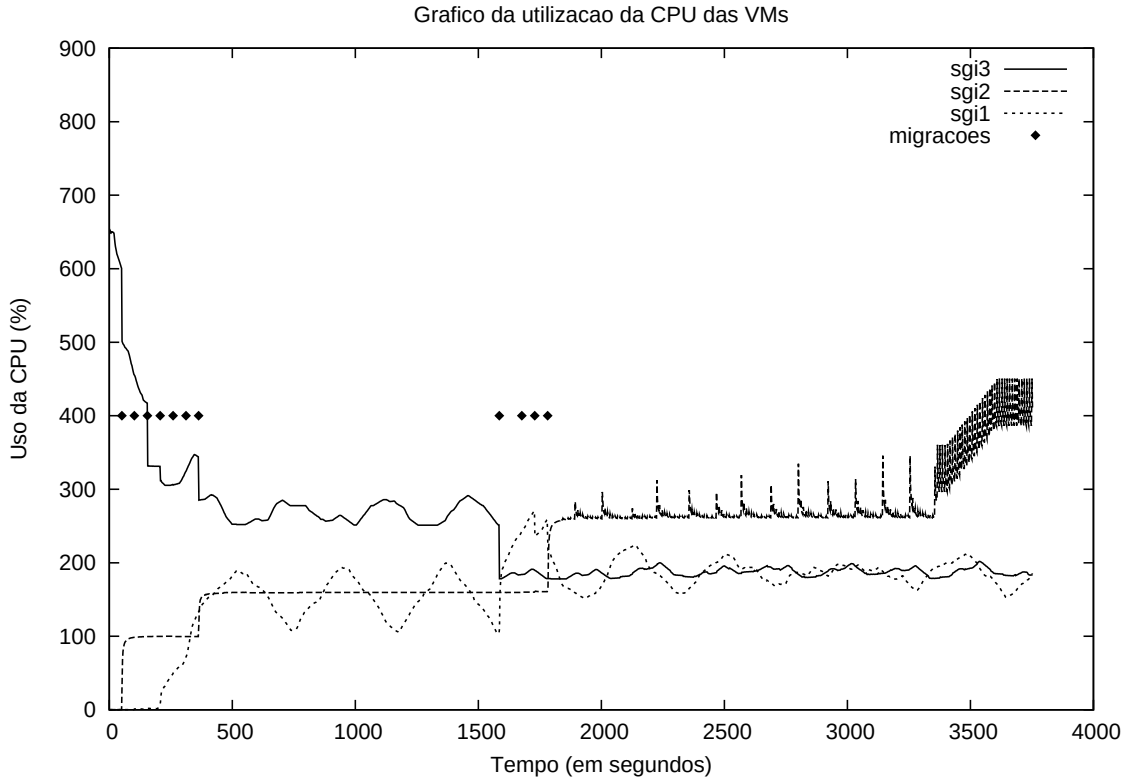

Figura 5.16: Carga de 3 servidores com 10 máquinas virtuais, e balanceador com *buffer* de 100 posições

5.3 Análise dos testes

Através do exemplo realizado pôde-se verificar que a ferramenta desenvolvida cumpre os requisitos determinados na seção 3.

Os resultados dos testes com cargas estáticas foram bons, porém os de carga dinâmica nem tanto.

No início o programa consegue equilibrar as cargas. Porém como mencionado na seção anterior, após um tempo houve uma certa instabilidade nas chamadas para a migração. A chamada da Libvirt para a migração retornou erro em alguns momentos. Esses erros não são erros da Libvirt em si, mas sim do Xen, e não afetam o sistema.

Quando eles ocorrem, a migração não acontece, e a máquina virtual que seria migrada continua no servidor antigo. Porém em algumas tentativas de migrações ocorreram problemas menos sutis, os quais acabam gerando uma instabilidade na máquina virtual a qual estava-se tentando migrar. Após uma pesquisa mais profunda, constatou-se que a máquina virtual continua normalmente, e que as anomalias geradas nos gráficos são problemas na hora da coleta dos dados, no momento da instabilidade.

A versão do Xen utilizada para esses testes é um pouco antiga, e é provável que a versão mais atual não apresente estes problemas.

No resto, a ferramenta Vmbal poderia ter algumas melhorias, como por exemplo, um maior número de comandos para o *shell*, mais opções de configurações e a análise de outros tipos de dados.

6 CONCLUSÃO

Neste trabalho, desenvolveu-se uma ferramenta para balanceamento automático de carga de servidores virtualizados, utilizando a API Libvirt em conjunto com o MMV Xen. O *software* desenvolvido tem como principal objetivo prover uma forma de melhorar a distribuição das máquinas virtuais em um conjunto de servidores. A ferramenta Vmbal possui basicamente dois usos: atuar passivamente apenas coletando dados de um conjunto de servidores; ou atuar ativamente, coletando dados e tomando decisões a respeito de como melhorar a distribuição de carga da CPU de um conjunto de servidores virtualizados.

Para o desenvolvimento deste trabalho foram pesquisadas e testadas diversas ferramentas que tornassem possível a implementação deste *software*, tais como, a API Libvirt, o MMV Xen, a linguagem de programação C++, bash *script*, além de outras ferramentas presentes no ambiente Linux.

Pôde-se verificar na prática o funcionamento da ferramenta desenvolvida, através de testes realizados em servidores do LSC, nos quais a ferramenta mostrou cumprir com os requisitos determinados na fase de projeto. Houveram alguns problemas relacionados com a migração, que devem ser pesquisados mais profundamente.

O balanceador de cargas gerado neste trabalho pode ser de grande importância futuramente, pois, com a necessidade de corte de custos em TI gerados em parte pela recessão presente, a tendência é o aumento do uso de virtualização nos servidores, e conseqüentemente o aumento da necessidade de programas gerenciadores de conjuntos de máquinas virtuais. Este trabalho é uma implementação básica de um balanceador de cargas, porém possibilita futuras expansões das funcionalidades deste produto. Alguns itens propostos como trabalhos futuros são:

- Correção de eventuais *bugs* presente nesta implementação;

- Implementar algoritmos para análise de dados da memória, rede e disco;
- Adicionar diferentes algoritmos de análise dos dados;
- Adicionar outros tipos de dados como por exemplo a temperatura;
- Implementar um modo de economia de energia, o qual buscasse uma menor utilização de energia elétrica, em vez do equilíbrio de cargas;
- Adicionar novos tipos de configurações para o programa;
- Melhorar a interatividade com o usuário, provendo uma interface em linha de comando com mais opções, ou até mesmo uma interface gráfica;
- Aplicação da ferramenta Vmbal aos servidores do Curso de Ciência da Computação da UFSM.

REFERÊNCIAS

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: SOSP '03: PROCEEDINGS OF THE NINETEENTH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 2003, New York, NY, USA. **Anais...** ACM, 2003. p.164–177.

CLARK, C.; FRASER, K.; H, S.; HANSEN, J. G.; JUL, E.; LIMPACH, C.; PRATT, I.; WARFIELD, A. Live Migration of Virtual Machines. In: IN PROCEEDINGS OF THE 2ND ACM/USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI, 2005. **Anais...** [S.l.: s.n.], 2005. p.273–286.

Gnuplot. **gnuplot homepage**. Disponível em: http://www.gnuplot.info/ . Acesso em: novembro de 2008.

GUILHERME P. KOSLOVSKI, M. P. B. e. A. S. C. Uso de Virtualização de Recursos Computacionais na Administração de Redes. **ERRC**, [S.l.], 2006.

KVM. **Kernel Based Virtual Machine**. Disponível em: http://kvm.qumranet.com/kvmwiki . Acesso em: dezembro de 2008.

LBVM. **Load Balancing of Virtual Machines**. Disponível em: http://lbvm.sourceforge.net/ . Acesso em: outubro de 2008.

Libvirt. **Libvirt - The virtualization API**. Disponível em: http://libvirt.org/ . Acesso em: outubro de 2008.

Novell. **ZENworks Virtual Machine Management**. Disponível em: http://www.novell.com/products/zenworks/virtualmachinemanagement/ . Acesso em: outubro de 2008.

OpenVZ. **OpenVZ wiki**. Disponível em: http://wiki.openvz.org/Main_Page . Acesso em: outubro de 2008.

QEMU. **Open source processor emulator**. Disponível em: http://bellard.org/qemu/ . Acesso em: outubro de 2008.

SHEPLER, S.; CALLAGHAN, B.; ROBINSON, D.; THURLOW, R.; BEAME, C.; EISLER, M.; NOVECK, D. **Network File System (NFS) version 4 Protocol**. United States: RFC Editor, 2003.

STROUSTRUP, B. **The C++ Programming Language**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

TANENBAUM, A. S. **Modern Operating Systems**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

Virtual Iron. **Live Capacity**. Disponível em: http://blog.virtualiron.com/2007/01/11/virtual-irons-livecapacity-exp.html . Acesso em: outubro de 2008.

VMware Inc. **Virtualization via hypervisor, virtual machine & server consolidation - VMWare**. Disponível em: http://www.vmware.com . Acesso em: outubro de 2008.

VMware Inc. **VMware VMotion Features**. Disponível em: http://www.vmware.com/products/vi/vc/vmotion_features.html . Acesso em: outubro de 2008.

Webstart. **The SSL Protocol**. Disponível em: http://www.webstart.com/jed/papers/HRM/references/ssl.html . Acesso em: outubro de 2008.

Wikipedia. **Transport Layer Security**. Disponível em: http://en.wikipedia.org/wiki/Trans port_Layer_Security . Acesso em: outubro de 2008.

Xen. **Xen 3.2 data sheet**. Disponível em: http://www.xen.org/files/Xen_3.2Datasheet.pdf . Acesso em: outubro de 2008.

APÊNDICE A MANUAL DE INSTALAÇÃO

Este documento visa descrever as atividades necessárias para a instalação e configuração da ferramenta **vmbal** versão 0.1. Inicialmente serão listados os *softwares* dos quais a ferramenta vmbal é dependente; em seguida serão listados os passos necessários para instalação dos *softwares* necessários em ambiente Linux.

A.1 Requisitos

Para instalação e utilização do vmbal, é necessário que os seguintes *softwares* estejam instalados e corretamente configurados:

- **Sistema operacional GNU/Linux**
- **Xen**
- **NFS**
- **Libvirt**

As próximas seções especificarão a instalação dos *softwares* necessários para o funcionamento do vmbal.

A.2 Instalação da Libvirt

É necessário instalar a Libvirt em cada servidor físico que irá comunicar-se com o programa de balanceamento de carga. A sua instalação pode ser feita de 2 maneiras: automaticamente, a partir dos repositórios do sistema, ou manualmente, baixando e compilando a mão.

A Libvirt por ser uma API que está em constante desenvolvimento, ela evolui rapidamente, então por este motivo, a versão que geralmente está nos repositórios não é a mais atual. Por isso, preferiu-se fazer a instalação manual.

Para a instalação manual, primeiramente deve-se fazer o *download* do arquivo empacotado contendo o código fonte da biblioteca. Este pode ser encontrado nesta página da Web:

http://libvirt.org/downloads.html

Após o *download* deve ser feito o desempacotamento e entrar na pasta criada.

$ tar xzf libvirt-0.5.0.tar.gz

$ cd libvirt

Após isto a Libvirt deve ser compilada.

$ ./configure –prefix=local_instalacao

$ make

$ make install

A.3 Configuração do libvirtd

Após a compilação e instalação da Libvirt, é necessário configurar seu *daemon*.

O arquivo de configuração é o *libvirtd.conf*, e sua localização varia de acordo com o local de instalação.

A configuração desse arquivo deve ser realizada com muita atenção, e dependendo da necessidade de cada sistema, pode variar um pouco. Não entrarei em detalhes sobre cada opção, porém o arquivo de configuração está bem documentado. A configuração padrão deste arquivo geralmente já está funcionando.

A.4 Criação dos certificados

O próximo passo após a configuração do *daemon* é a criação dos certificados. Esta parte é composta de 3 etapas: criação do certificado de autoridade, certificado para cada servidor e certificado para cada cliente. Estes passos estão descritos nas próximas seções.

A.4.1 Certificado de autoridade

Para a criação do certificado de autoridade, primeiramente deve ser gerado uma chave privada única.

$ certtool –generate-privkey > cakey.pem

Após isso, criar um arquivo com o nome *ca.info* com o seguinte conteúdo, observando que deve ser alterado o campo *cn*:

cn = nome da organização

ca

cert_signing_key

O seguinte comando gera dois arquivos: *cakey.pem* e *cacert.pem*.

$ certtool –generate-self-signed –load-privkey cakey.pem –template ca.info –outfile cacert.pem

O arquivo *cakey.pem* é a chave privada, e deve ser mantido em segredo. O arquivo *cacert.pem* é o certificado, e este é publico, e deve ser instalado em todos os servidores e clientes. A instalação normal para este arquivo é o seguinte local: */etc/pki/CA/cacert.pem*.

A.4.2 Certificado para servidores

Para cada servidor, é necessário a criação de um certificado. Primeiramente deve ser gerada uma chave privada para o servidor:

$ certtool –generate-privkey > serverkey.pem

Após isso deve ser criado um arquivo com o nome *server.info*, com o seguinte conteúdo:

organization = nome da organização

cn = host

tls_www_server

encryption_key

signing_key

Observar que é necessário alterar o campo *organization* e o campo *cn*. O campo *cn* deve ser o nome que os clientes usarão para comunicar-se com o servidor. Por exemplo, se o cliente usar a URI *xen://servidor1/* para comunicar-se com o servidor, o *cn* deve ser **servidor1**.

Feito isso, usar o seguinte comando para criar os certificados:

$ certtool –generate-certificate –load-privkey serverkey.pem –load-ca-certificate cacert.pem –load-ca-privkey cakey.pem –template server.info –outfile servercert.pem

Esse processo irá gerar 2 arquivos: *serverkey.pem* e *servercert.pem*. O arquivo *serverkey.pem* é a chave privada do servidor e deve ser copiado para a pasta */etc/pki/libvirt/private/serverkey.pem* do servidor. O arquivo *servercert.pem* é o certificado e deve ser copiado para a pasta */etc/pki/libvirt/servercert.pem* do servidor.

A.4.3 Certificado para os clientes

O computador que irá executar o programa de balanceamento de carga precisa ter o certificado de cliente configurado. Para isso, primeiramente deve ser gerado uma chave única com o seguinte comando:

$ certtool –generate-privkey > clientkey.pem

Após isso, é necessário a criação de um arquivo com o nome *client.info*, e editar os parâmetros *country, state, locality, organization* e *cn*.

country = BR

state = Rio Grande do Sul

locality = Santa Maria

organization = organização

cn = client1

tls_www_client

encryption_key

signing_key

Para gerar o certificado usar o comando:

$ certtool –generate-certificate –load-privkey clientkey.pem –load-ca-certificate cacert.pem –load-ca-privkey cakey.pem –template client.info –outfile clientcert.pem

Instalar o arquivo *clientkey.pem* na pasta */etc/pki/libvirt/private/clientkey.pem* e o arquivo *clientcert.pem* na pasta */etc/pki/libvirt/clientcert.pem,* ambos no cliente.

A.5 Configuração do xend

Para o suporte a migração é necessário algumas alterações no arquivo */etc/xen/xend-config.sxp*. As seguintes opções devem estar descomentadas:

(xen-api-server ((unix)))

(xend-unix-server yes)

(xend-relocation-server yes)

(xend-unix-path /var/lib/xend/xend-socket)

(xend-relocation-port 8002)

(xend-address ’ ’)

(xend-relocation-address ’ ’)

Vale lembrar que dependendo da configuração do sistema pode ser que haja necessidade de alterar alguns campos.

Outra necessidade para a migração é que as imagens das máquinas virtuais estejam acessíveis a todos os servidores. Isto pode ser feito com a ajuda do NFS, por exemplo.

A.6 Instalação do Vmbal

O código fonte da ferramenta Vmbal está disponível livremente no seguinte endereço eletrônico: http://vmbal.sourceforge.net/ . Após fazer o *download* é necessário alguns passos para a compilação da ferramenta.

Primeiramente deve-se extraí-la do pacote compactado. Para isso digite o comando:

tar -xzf vmbal-0.1.tar.gz

Isso criará uma pasta chamada vmbal. Depois disso, entrar na pasta e compilar, utilizando os seguintes comandos:

cd vmbal

./autogen.sh

make

Após esses comandos, se tudo ocorrer sem erros, o programa estará compilado e o executável ficará dentro da pasta *src*.

A.7 Funcionamento do Vmbal

De acordo com o que foi descrito no capítulo 4, para o funcionamento da ferramenta Vmbal é necessário que o arquivo de configuração *vmbal.conf* esteja configurado adequadamente. Este arquivo deve estar na mesma pasta do programa compilado.

As opções são definidas com um caractere = após cada opção, e deve conter apenas uma por linha.

Para adicionar um servidor, utiliza-se a opção *host*. Por exemplo: *host=xen://servidor1/*. Pode ter vários *hosts* definidos.

Os tempos de intervalo de medição e intervalo de análise são definidos respectivamente com as opções *intervalo-medicao* e *intervalo-analise*. Os valores devem ser um número inteiro maior que zero. Por exemplo: *intervalo-medicao=5*.

Foi implementado um modo para fazer somente a coleta de dados e não fazer a análise. Utiliza-se a opção *modo-somente-coleta* para isso. Os valores aceitos são 0 ou 1, 0 para

falso e 1 para verdadeiro. Por exemplo: *modo-somente-coleta=1*.

O tamanho do *buffer* de dados é definido com a opção *tamanho-buffer-dados*. Os valores aceitos são inteiros maior que zero, e representam o tamanho do *buffer* em posições. Por exemplo: *tamanho-buffer-dados=100*.

É possível salvar registro de todos os dados lidos com a opção *salvar-log*. Os valores aceitos são somente 0 ou 1. Por exemplo: *salvar-log=1*.

Se a opção *salvar-log* estiver definida como verdadeira, é possível definir a pasta de saída destes registros com a opção *caminho-logs*. Por exemplo: *caminho-logs=logs/*.

Para definir a porcentagem de desequilíbrio, utiliza-se a opção *porcentagem-desequilibrio*. Os valores aceitos são inteiros maiores que 0. Por exemplo *porcentagem-desequilibrio=60*.

O tempo que um desequilíbrio deve ser mantido para que haja mudança de estado no programa é definido com a opção *tempo-desequilibrio*. Os valores aceitos são inteiros maiores que 0. Por exemplo *tempo-desequilibrio=30*.

Todas essas opções exceto a opção *host* possuem valores padrões caso não estejam definidas. Após a configuração, a ferramenta já pode ser executada. Como a interface deste programa é em linha de comandos, é recomendável um terminal para a inicialização deste programa.

Depois de inicializar, o programa irá inicialmente mostrar todos os servidores, máquinas virtuais, interfaces de rede e interfaces de disco que foram encontrados. Após isso o programa executa silenciosamente, porém quando há algum erro, o programa imprime mensagens de erro no console. Se o erro não for fatal, o programa segue.

Como mencionado no capítulo 4, foi implementado também um pequeno *shell*, e com isso pode ser passado alguns comandos em tempo de execução. Atualmente só possui 3 comandos úteis: *carga-cpu, lista* e *gera-graficos*. O comando *carga-cpu* imprime na tela a porcentagem de ocupação de cada servidor. O comando *lista* imprime no console todas as máquinas virtuais e em que servidores elas estão. O comando *gera-graficos*, cria gráficos da utilização dos dispositivos baseados nos registros gerados. Estes gráficos são gerados com a ajuda da ferramenta gnuplot (Gnuplot, 2008). Para sair do programa utilizar o comando *exit*.