Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Sistema Automatizado de Gerência de Recursos para Ambientes Virtualizados

Autor:

Govinda Mohini Gonzalez Bezerra

Orientador:

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

Examinador:

Prof. Miguel Elias Mitre Campista, D.Sc.

Examinador:

Hugo Eiji Tibana Carvalho, M.Sc.

DEL

Setembro de 2013

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

AGRADECIMENTOS

Agradeço à minha família pelo apoio e carinho. Em especial, aos meus pais e à minha tia por toda dedicação e cuidado.

Ao meu orientador, professor Otto, pelos conselhos, dedicação e compreensão durante a realização deste trabalho.

Aos amigos do Grupo de Teleinformática e Automação, em especial Andrés, Brasil, Diogo, Guimarães, Lyno e Martin pela ajuda e pelo constante incentivo.

Agradeço à UFRJ e à Télécom Bretagne por me proporcionarem uma formação sólida. Agradeço também ao CNPq pelo financiamento deste trabalho.

Por fim, agradeço a todos que de alguma forma contribuíram com a minha formação.

RESUMO

A virtualização de computadores é uma tecnologia em expansão devido, principalmente, ao aumento da capacidade de processamento e armazenamento dos computadores modernos, a demanda por servidores consolidados e o sucesso da computação em nuvem. Um dos grandes desafios em ambientes virtualizados é gerenciar dinamicamente os recursos disponíveis, se adaptando às oscilações de carga de trabalho e garantindo o nível de serviço das diferentes aplicações em execução. Neste contexto, este trabalho propõe uma ferramenta capaz de monitorar máquinas físicas e virtuais, e construir perfis de uso dos diferentes recursos computacionais, como processador, memória e rede. Com base nos perfis, o sistema é capaz de detectar a escassez dos recursos monitorados e automaticamente redistribuir as cargas de trabalho, evitando a sobrecarga dos nós físicos e violações de acordos de níveis serviços. O sistema proposto realiza a distribuição de cargas através da funcionalidade de migração ao vivo de máquinas virtuais, presente nas principais tecnologias de virtualização, que permite transferir uma máquina virtual de um nó físico origem para um nó físico destino, sem a interrupção dos serviços em execução. O sistema proposto foi desenvolvido e o seu desempenho foi avaliado. Os resultados obtidos mostram a eficácia do sistema em distribuir as cargas de trabalho e eliminar sobrecargas de processamento e memória dos nós físicos monitorados.

Palavras-Chave: Virtualização, Gerenciamento de Recursos, Perfil de Uso, Migração ao Vivo, Distribuição de Cargas.

ABSTRACT

Virtualization is a growing technology that can substantially increase flexibility and manageability in data centers. It is becoming increasingly popular due to its wide adoption in cloud computing and the demand for consolidated servers. One of the major challenges in virtualized environments is to dynamically manage the resource allocation to virtual machines, reducing idles resources and avoiding Service Level Agreements (SLAs) violations. In this context, this work proposes a tool that monitors the physical and virtual machines in a virtualized environment and builds usage profiles of different computing resources, such as processor, memory and network. Based on the profiles, the system is able to detect the lack of resources and automatically redistribute workloads, avoiding bottlenecks and performance degradation due to server saturation. The workload distribution mechanism is based on the live migration feature which allows the transfer of a virtual machine from one physical server to another without interrupting the services running in the virtual machine. The results show that the developed system meets satisfactorily the proposed objectives, being able to distribute workloads and eliminate overloads on the monitored hosts.

Key-words: Virtualization, Resourse Management, Use Profiles, Live Migration, Load Distribution.

SIGLAS

FITS - Future Internet Testbed with Security

VPN - Virtual Private Network

TLS - Transport Layer Security

SLA - Service Level Agreement

VM - Virtual Machine - Máquina Virtual

PM - Physical Machine - Máquina Física

CPU - Central Processing Unit

IP - Internet Protocol

E/S - Entrada e Saída

MAC - Medium Access Control

SEDF - Simple Earliest Deadline First

BVT - Borrowed Virtual Time

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Virtualização e Gerência de Recursos

A virtualização foi desenvolvida pela IBM na década de 60 para permitir o acesso simultâneo e interativo aos computadores de grande porte (*mainframes*) [1]. Cada máquina virtual (VM) consistia em uma instância de máquina física que dava aos usuários a ilusão de acessar a máquina física diretamente. Era uma solução simples e transparente de permitir o compartilhamento dos recursos de hardware, ainda muito caros e escassos naquela época. Com a redução do preço dos computadores e o surgimento dos computadores pessoais, a virtualização praticamente desapareceu durante as décadas de 70 e 80. Entretanto, com o surgimento de uma grande variedade de computadores e sistemas operacionais na década de 90, as tecnologias de virtualização ganharam novamente força, pois permitiam a execução, em um mesmo computador, de aplicações projetadas para diferentes plataformas [2].

À medida que o portfólio de serviços aumenta e a concorrência se torna cada vez maior, as empresas de Tecnologia da Informação (TI) devem encontrar maneiras de aumentar a utilização dos servidores e reduzir os custos de operação. Estratégias de virtualização e consolidação de servidores tem sido largamente empregadas nos novos *data centers*, pois permitem um melhor gerenciamento de recursos, aumentam a flexibilidade da infraestrutura de TI e reduzem gastos com hardware e energia elétrica. Além disto, as tecnologias de virtualização também possibilitaram o surgimento de novos modelos de negócio, como a computação em nuvem que oferece

recursos computacionais como serviço, permitindo aos clientes aumentar a capacidade ou adicionar recursos aos seus serviços, sem investir numa nova infraestrutura, treinamento de novos funcionários ou novas licenças de softwares.

O baixo nível de utilização de servidores é uma grande preocupação dos administradores de centro de dados [3]. Os custos de energia são altos e a baixa utilização das máquinas se traduz em uma quantidade maior de computadores e de gastos com energia e refrigeração, além de necessitar de mais espaço físico para acomodar as máquinas. A baixa utilização dos servidores pode ter diferentes causas, a mais comum é o super dimensionamento das máquinas virtuais visando atender os requisitos das aplicações nos períodos de pico de demanda. Como a demanda por recursos normalmente é dinâmica e apresenta fortes oscilações durante o dia, muitos recursos ficam ociosos.

Um dos grandes desafios dos ambientes virtualizados é gerenciar a alocação dos recursos computacionais, de forma a otimizar a utilização dos computadores disponíveis. A grande dificuldade está em alocação de recursos de acordo com as variações da carga de trabalho e garantindo recursos suficientes às maquinas virtuais, assegurando o nível de serviço adequado a todas as aplicações em execução [4]. Um sistema de gerência automática capaz de detectar a escassez de recursos em nós do sistema e controlar o aprovisionamento desses recursos às máquinas virtuais é fundamental.

1.2 Proposta e Objetivo do Projeto

O objetivo deste trabalho é o projeto de um sistema automático de gerência de recursos que seja capaz de realocar dinamicamente as máquinas virtuais de acordo com o nível de utilização das máquinas físicas monitoradas. O sistema proposto monitora os recursos computacionais, como capacidade de processamento, memória e banda passante, em ambientes virtualizados e detecta pontos de sobrecarga. Quando há recursos ociosos suficientes nas máquinas monitoradas, o sistema é capaz de tomar decisões que eliminam a sobrecarga. A ferramenta é capaz de distribuir automaticamente as cargas de trabalho entre as máquinas físicas disponíveis, realocando as

máquinas virtuais críticas em máquinas físicas menos sobrecarregadas. Desta forma, os objetivos específicos do sistema proposto são:

1. O monitoramento do uso de recursos das máquinas físicas e virtuais;
2. A criação de perfis de uso de recursos para cada uma das máquinas;
3. A utilização de um algoritmo capaz de tomar decisões, baseado nos perfis de uso, que diminua os pontos de sobrecarga;
4. A proposição de uma interface amigável com o usuário do sistema, como um painel de controle que permita a visualização, em tempo real, da alocação e utilização dos recursos computacionais monitorados.

1.3 Metodologia

O sistema de monitoramento utiliza uma abordagem do tipo caixa preta, ou seja, todo o monitoramento é realizado através de dados visíveis a partir do exterior das máquinas virtuais. Esta abordagem foi escolhida, pois garante menor interferência e maior privacidade no uso das máquinas virtuais, o que é fortemente desejável em ambientes virtualizados, visto que em muitas aplicações o proprietário do hardware não é o proprietário das aplicações que nele são executadas e a instalação de programas de monitoramento nas máquinas virtuais é inviável.

O projeto foi implementado utilizando a plataforma de testes FITS [5] desenvolvida pelo Grupo de Teleinformática e Automação da COPPE/UFRJ. O sistema proposto monitora remotamente as máquinas físicas da plataforma de testes. Os recursos monitorados são: uso do processador, quantidade de memória alocada e tráfego de rede das máquinas virtuais. O sistema cria perfis de uso destes recursos para cada uma das máquinas físicas e virtuais e, com base nestes perfis, é capaz de detectar pontos de sobrecarga.

A redistribuição de cargas é realizada utilizando a funcionalidade de migração ao vivo presente em muitas tecnologias de virtualização. Este tipo de migração permite a transferência de máquinas virtuais entre máquinas físicas sem a interrupção dos serviços em execução. O problema de encontrar a melhor combinação

de máquinas virtuais e máquinas físicas pode ser classificado como um problema de empacotamento (*bin packing problem*) [6], o qual consiste em organizar itens de diversos volumes dentro de unidades maiores (objetos), satisfazendo um conjunto de restrições e otimizando uma determinada função. A complexidade deste tipo de problema é *np-hard* e a solução adotada neste projeto é baseada em heurísticas extraídas dos perfis das máquinas monitoradas.

1.4 Organização do Texto

O restante do texto pode ser dividido em duas partes principais: o estudo dos temas relacionados e o projeto desenvolvido. A primeira parte consiste no estudo de virtualização e ferramentas utilizadas no projeto. A segunda parte foca no desenvolvimento de um sistema automatizado de gerência de recursos para ambientes virtualizados, sua implementação e os resultados obtidos.

O Capítulo 2 é dedicado à virtualização de computadores, a tecnologia Xen e o funcionamento da divisão de recursos, como processador, memória e dispositivos de E/S, entre as máquinas virtuais.

O Capítulo 3 apresenta as principais ferramentas e bibliotecas que foram utilizadas no desenvolvimento do projeto, além de uma breve descrição dos trabalhos relacionados.

O Capítulo 4 descreve detalhadamente o sistema realizado. A arquitetura do sistema é apresentada, assim como os diagramas de caso de uso.

Os resultados são apresentados no Capítulo 5. São mostrados os testes realizados e os resultados obtidos com a implementação do sistema.

E por fim, o Capítulo 6 apresenta a conclusão do trabalho e discute melhorias que podem ser realizadas.

Capítulo 2

A tecnologia de Virtualização

A virtualização de máquinas pode ser definida como uma tecnologia que permite o compartilhamento de recursos computacionais entre diferentes ambientes de execução, chamados de máquinas virtuais. Cada máquina virtual acessa uma abstração de recursos isolada, dando ao usuário a ilusão de acessar diretamente uma máquina física com todos os recursos computacionais dedicados. Desta forma, a virtualização permite que um computador físico execute diversos computadores lógicos em paralelo, cada um com aplicações e sistema operacional próprios (Figura 2.1).

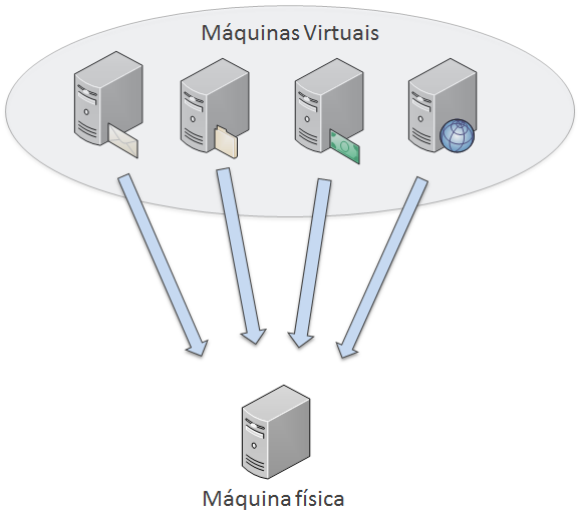


Figura 2.1: Virtualização de máquinas: quatro máquinas virtuais executando em uma única máquina física.

Atualmente, a tecnologia tem sido muito empregada nas empresas [7] por proporcionar uma infraestrutura de TI mais dinâmica, sendo capaz de aumentar o uso

das máquinas físicas disponíveis e realocar os recursos computacionais, de forma a responder às variações de demanda por recursos e melhor atender as necessidades dos diferentes serviços. A técnica facilita ainda a desassociação dos proprietários de hardware e os proprietários das aplicações.

Dentre os principais usos da virtualização podem ser citados:

- Consolidação de servidores: Permite consolidar cargas de trabalho de vários computadores subutilizados em uma quantidade menor de máquinas, aumentando o nível de utilização dos recursos computacionais disponíveis (Figura: 2.2), economizando em hardware, gerenciamento e administração da infraestrutura;

- Consolidação de aplicações: Diferentes programas podem exigir diferentes configurações de hardware e versões de softwares. Através da virtualização, é possível executá-los em uma mesma máquina física, cada um em uma máquina virtual específica que atenda os requisitos do programa. Esta técnica facilita a transição de tecnologias de uma empresa e permite executar programas desenvolvidos para hardwares e/ou sistemas operacionais obsoletos;

- *Sandboxing*: As máquinas virtuais são úteis para fornecer ambientes seguros e isolados (*sandboxes*) para a execução de aplicações suspeitas ou para testar programas ainda em desenvolvimento, sem interferir no funcionamento de outras aplicações da mesma máquina física. A virtualização também pode auxiliar na etapa de teste e validação de software, pois permite criar diferentes cenários de teste com diversos perfis de configuração em uma escala de teste difícil de se reproduzir apenas com máquinas físicas.

Realizar o particionamento de hardware para suportar a execução simultânea de vários sistemas operacionais impõe três desafios principais. O primeiro é garantir o isolamento entre as máquinas virtuais, pois não é aceitável que a execução de uma máquina virtual interfira negativamente no desempenho de outra, principalmente se as máquinas virtuais pertencerem a usuários mutuamente não confiáveis. O segundo consiste em suportar uma variedade de sistemas operacionais para conseguir

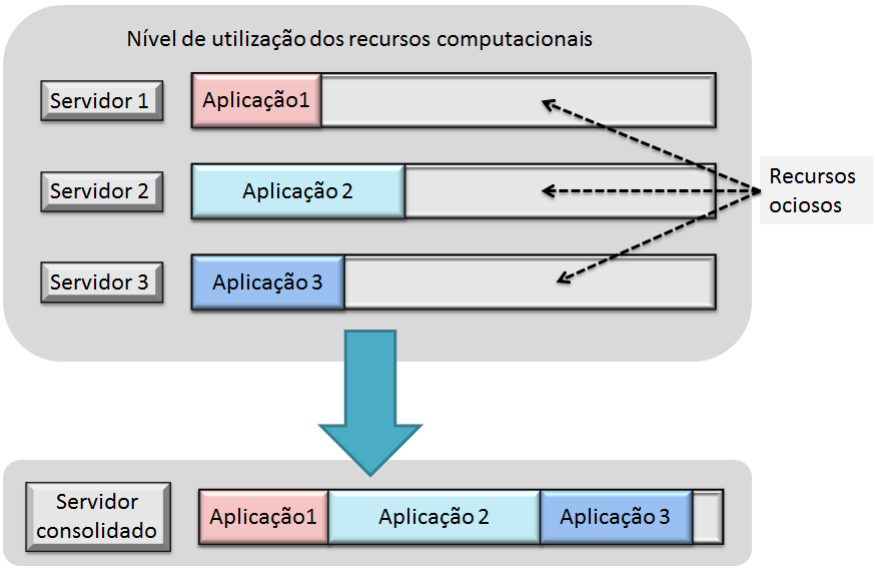


Figura 2.2: Consolidação de servidores: três servidores consolidados em um único servidor físico para aumentar a eficiência.

acomodar a diversidade de aplicações existentes. O terceiro é minimizar o impacto causado pela camada de virtualização no desempenho das aplicações. Assim, existem diversas tecnologias de virtualização, cada uma abordando o problema de uma forma diferente e oferecendo soluções distintas. Dentre as soluções mais populares podem ser citadas: VMWare, Xen, VirtualBox e QEMU.

A virtualização pode ocorrer em diferentes níveis, como no nível de hardware, do sistema operacional ou de aplicação. Qualquer que seja o nível de abstração em que ela ocorra, a ideia central continua sendo a mesma: os recursos da camada inferior são particionados, através de técnicas que permitem mapeá-los em diversos domínios virtuais na camada superior [1].

Neste projeto, é utilizada a virtualização no nível de hardware que consiste em abstrair o hardware através de uma camada de software chamada de hipervisor ou Monitor de Máquina Virtual (VMM). O hipervisor se situa entre o hardware e os sistemas operacionais convidados e é responsável por gerenciar o compartilhamento dos recursos físicos entre as máquinas virtuais. Este tipo de virtualização pode ser dividido em três categorias: virtualização completa, para-virtualização e virtualização assistida por hardware.

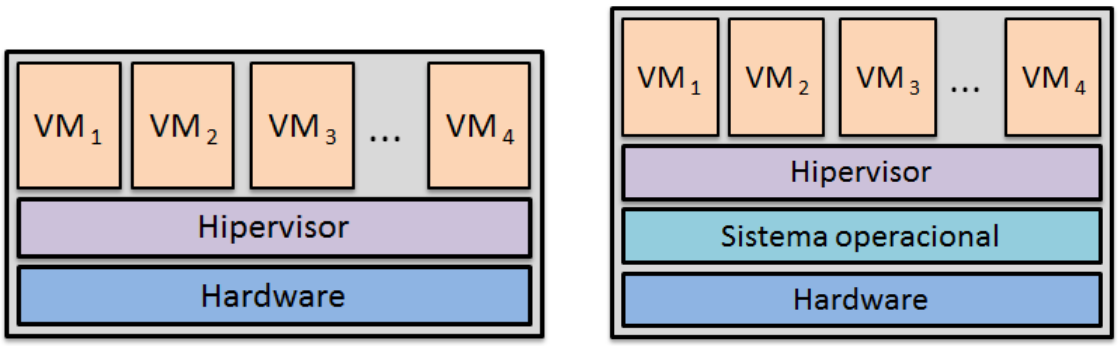
Na virtualização completa, o sistema operacional convidado não está ciente da virtualização e age como se executasse diretamente em uma máquina física dedicada. O hipervisor intercepta as instruções executadas pelo sistema operacional convidado para verificar se elas são instruções privilegiadas, o que representa um custo de processamento. A grande vantagem dessa abordagem é permitir a execução de sistemas operacionais sem nenhum tipo de adaptação para o contexto de virtualização, permitindo a virtualização de grande parte dos sistemas operacionais. A principal desvantagem é a sobrecarga de processamento e perda de desempenho causada pelo monitoramento de instruções realizado pelo hipervisor.

A segunda modalidade, conhecida como para-virtualização, visa solucionar os problemas da virtualização completa. Na para-virtualização, os sistemas operacionais convidados são modificados para o contexto da virtualização. Essas modificações permitem que os sistemas operacionais se comuniquem diretamente com o hipervisor quando precisarem executar instruções privilegiadas. Assim, o hipervisor não precisa monitorar e testar as instruções dos sistemas operacionais convidados, melhorando o desempenho do sistema. A principal desvantagem da para-virtualização é justamente a necessidade de realizar modificações no sistema operacional convidado, o que pode gerar custos extras de adaptação e atualização do software, além de limitar a portabilidade do sistema.

A terceira modalidade é a virtualização assistida por hardware [8]. Devido a popularização das tecnologias de virtualização, os fabricantes de hardware têm adaptado seus dispositivos para facilitar a virtualização. Os exemplos mais conhecidos são o AMD-V e o Intel VT. Nestes dois casos, chamadas específicas de CPU não são traduzidas pelo hipervisor, mas são enviados diretamente para a CPU. Isso reduz a carga do hipervisor e aumenta o desempenho, eliminando o tempo de tradução de das chamadas.

Os hipervisores também podem ser classificados em dois tipos: O hipervisor do tipo I é denominado nativo ou *bare metal* e é instalado diretamente no hardware (Figura 2.3(a)), semelhante à forma como um sistema operacional regular pode ser instalado em um único servidor. Já o hipervisor do tipo II é instalado em um

sistema operacional existente (Figura 2.3(b)), o que introduz uma sobrecarga de processamento maior que o tipo I, pois todos os recursos do ambiente operacional são gerenciados pelo sistema operacional, resultando em menor desempenho.



(a) Hipervisor tipo I.

(b) Hipervisor tipo II.

Figura 2.3: Tipos de hipervisores: hipervisor tipo I ou (*b*are metal), que virtualiza os recursos de hardware e hipervisor tipo II, que virtualiza o sistema operacional.

A ferramenta desenvolvida nesse trabalho é compatível com diversas tecnologias de virtualização, mas o enfoque será dado ao Xen, pois essa é a tecnologia utilizada na plataforma de teste FITS, onde a ferramenta proposta foi implementada.

2.1 Xen

O Xen [9] é um monitor de máquinas virtuais de código aberto desenvolvido pelo laboratório de computação da universidade de Cambridge e mantido como um software livre pela sua comunidade de desenvolvedores. O Xen é uma tecnologia de virtualização na camada de hardware, seu hipervisor é do tipo I, permitindo tanto a virtualização completa quanto a para-virtualização.

Um ambiente virtual Xen é composto de vários elementos que trabalham em conjunto para oferecer um ambiente de virtualização [10], são eles:

- Hipervisor: O hipervisor Xen é a camada de abstração básica de software que fica diretamente sobre o hardware e abaixo de todos os sistemas operacionais. É responsável pelo escalonamento de CPU e pelo particionamento de memória das diferentes máquinas virtuais em execução na máquina física. O hypervisor

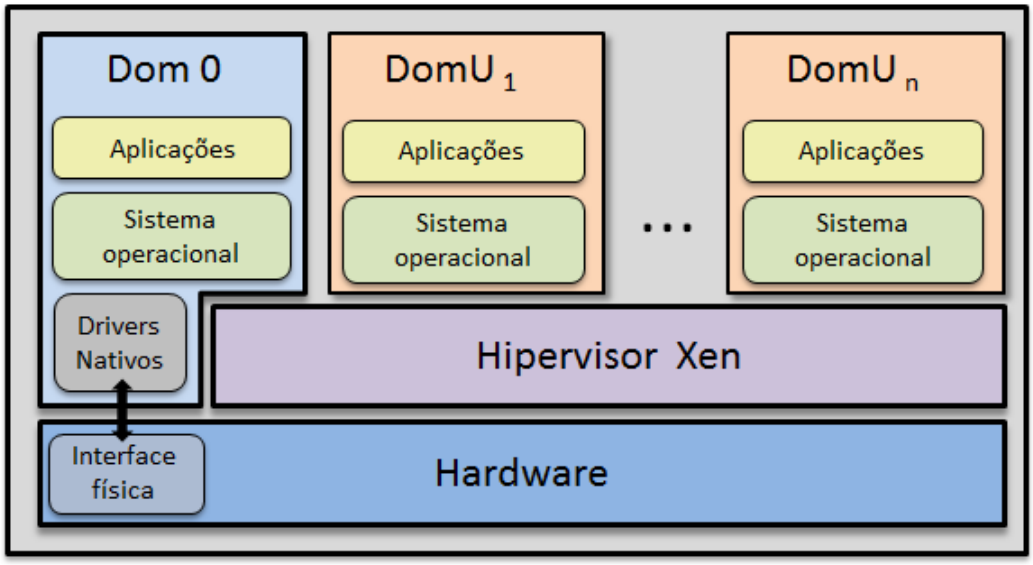


Figura 2.4: Arquitetura Xen: hipervisor virtualizando o hardware. Os domínios não privilegiados (domU), que hospedam os sistemas virtuais, e o domínio privilegiado (dom0), que centraliza as operações de E/S e tem acesso direto aos *drivers* da máquina física.

não só abstrai o hardware para as máquinas virtuais, mas também controla a execução de máquinas virtuais no ambiente de processamento compartilhado. O hipervisor não tem conhecimento de rede, de dispositivos de armazenamento externos ou de quaisquer outras funções comuns de E/S encontrados em um sistema de computação;

- Domínio-0: Este domínio é uma máquina virtual que tem privilégios especiais para acessar os recursos físicos de E/S e para interagir e gerenciar os outros domínios em execução no sistema. Todos os ambientes de virtualização Xen requerem que o Domínio-0 esteja em execução antes de quaisquer outras máquinas virtuais possam ser criadas;

- Domínios não privilegiados ou Domínios U: Estes domínios, ao contrário do Domínio-0, não tem acesso direto ao hardware da máquina. Todas as máquinas virtuais para-virtualizadas executados em um hipervisor Xen possuem sistemas operacionais modificados, tais como Solaris, FreeBSD, Debian e outros sistemas operacionais UNIX. Já as máquinas virtuais em sistemas com virtualização completa não requerem nenhuma modificação do sistema operacional

e podem, portanto, executar uma gama maior de sistemas operacionais, como o Windows padrão ou qualquer outro sistema operacional. Os domínios para-virtualizados contêm dois *drivers* para rede e acesso ao disco, o *PV Network Driver* e o *PV Block Driver*.

2.1.1 Acesso aos dispositivos de entrada e saída

Na arquitetura Xen, cada domínio convidado implementa um *driver* para sua interface de rede virtual, chamado de *netfront driver*. O Domínio-0 implementa um *netback driver* que atua como um intermediário entre os *netfront drivers* e o *driver* da interface de rede física. O *driver* de dispositivo físico está localizado no Domínio-0 e pode acessar a placa de rede física, porém interrupções da placa de rede são tratadas primeiramente pelo hipervisor, que por sua vez envia o sinal de interrupção para o Domínio-0 utilizando canais de eventos. Os canais de eventos consistem em um mecanismo de notificação assíncrona utilizado para comunicação entre domínios. Estes canais de eventos são utilizados estritamente para notificações, já para enviar mensagens inter-domínios, o Xen utiliza um mecanismo de compartilhamento de página chamado *I/O-rings* [11]. Para agilizar a transferência dos dados de E/S, o Xen utiliza mecanismo de troca de páginas *page-flipping* que permite trocar as páginas de memória contendo os dados de E/S entre os domínios virtuais e o Domínio-0.

Quando um pacote para um dos domínios chega a placa de rede física, o hipervisor recebe uma interrupção que, por sua vez, notifica Domínio-0 da chegada dos pacotes. Quando o escalonador autorizar o Domínio-0 a utilizar o processador, o *netback driver* verifica o destino dos pacotes. O Domínio-0 notifica, então, o domínio de destino e atualiza os *reception-I/O-rings* para copiar os pacotes em seus espaços de memória. Da próxima vez que domínio de destino utilizar o processador, ele verá os pacotes que foram recebidos e pode processá-los como qualquer sistema operacional padrão faria. Da mesma forma, quando os pacotes são enviados por um domínio convidado, ele notifica o Domínio-0 através do seu canal do evento e o Domínio-0, quando escalonado, entrega os pacotes para a placa de rede.

2.1.2 Escalonamento

O Xen realiza a virtualização do processador de forma semelhante ao escalonamento de processos de um sistema operacional. Para cada máquina virtual é atribuída uma quantidade de CPUs virtuais (vCPU) que é vista, pelas máquinas virtuais, como núcleos físicos do processador. O funcionamento de uma vCPU é semelhante ao funcionamento de um núcleo de CPU convencional, entretanto, uma vCPU normalmente não representa um único núcleo físico, mas sim, intervalos de tempo de todos os núcleos físicos disponíveis.

No arquivo de configuração de uma máquina virtual, é possível definir quantas vCPUs poderão ser usadas pelo domínio. A alocação de mais de uma vCPU à uma máquina virtual melhora o desempenho de aplicações multi-threaded, entretanto, alocar mais vCPUs que o necessário pode prejudicar o desempenho da máquina virtual. Por exemplo, se um computador com quatro núcleos de CPU estiver executando quatro máquinas virtuais, onde três delas tem uma única vCPU alocada e a outra com quatro vCPUs. Na melhor das hipóteses, apenas as três máquinas virtuais com uma vCPU poderão ser executadas simultaneamente. A máquina virtual com 4 vCPU deverá aguardar todos os quatro núcleos ficarem ociosos ao mesmo tempo. Este exemplo mostra que domínios com um número menor de vCPUs tendem a ser executados com maior frequência do que domínios super dimensionados.

O Xen fornece alguns parâmetros que permitem aos usuários ajustar a prioridade dos domínios no uso do processador. Para cada domínio, incluindo o Domínio-0, são atribuídos um valor de peso e outro de *cap*. O peso é utilizado para calcular a fatia de processador que o domínio terá direito e o seu valor pode variar de 1 até 65535, sendo 256 o valor padrão. Por exemplo, um domínio com peso de 512 poderá utilizar o processador duas vezes mais que um domínio com 256. Já o *cap* é um parâmetro opcional que representa a quantidade máxima de CPU que um domínio poderá usar, ainda que a CPU tenha mais recursos disponíveis. O valor de *cap* é expresso em porcentagem de CPU, onde o valor 100 representa um núcleo de CPU. Assim, um *cap* de 50 representa a metade de uma CPU e 400 indica 4 núcleos de CPU. O valor padrão é 0 (zero) e significa que não há limite superior para a utilização do

processador.

O Xen também fornece alguns parâmetros para ajustar o comportamento global do escalonador. O primeiro deles é o *timeslice* que consiste no intervalo de tempo durante o qual um processo pode ser executado. O valor padrão no Xen é de 30ms. Dependendo do tipo de aplicação em execução na máquina virtual, pode ser desejável ajustar este valor principalmente para aplicações sensíveis à latência, porém este ajuste deve ser feito com cautela, pois reduzir o valor do *timeslice*, aumenta a frequência da troca de contexto do processador, o que pode reduzir a eficácia do sistema. O segundo parâmetro é o *Schedule Rate Limiting* que consiste no intervalo de tempo mínimo durante o qual uma máquina virtual pode ser executada sem ser interrompida, ou seja, se um processo está em execução e um outro processo de maior prioridade acorda, o processo em execução poderá continuar utilizando a CPU até que o *Schedule Rate Limiting* seja atingido. Este parâmetro foi adicionado ao Xen, pois se verificou que, sob certas circunstâncias, algumas máquinas virtuais acordavam, realizavam alguns micro segundos de trabalho e voltavam a dormir. Estas máquinas virtuais continham aplicações sensíveis à latência, logo tinham prioridade para executar sempre que necessário. O problema era que elas causavam milhares de escalonamentos por segundo, o que implicava em uma quantidade muito significativa de tempo gasto pelo escalonador realizando a troca de contexto, ao invés de fazer o trabalho efetivo.

A versão do Xen utilizada neste projeto é a 4.2. Nesta versão, existem três algoritmos de escalonamento disponíveis, o Credit Scheduler, escalonador padrão do xen baseado em créditos, o SEDF (Simple Earliest Deadline First) e o BVT (Borrowed Virtual Time).

Os algoritmos de escalonamento citados acima podem ser classificados de acordo com os seguintes parâmetros [12]:

- Proporcional x Justo: Um escalonador proporcional realiza uma divisão instantânea do processador, de modo proporcional ao peso atribuído a cada uma das máquinas virtuais. Já um escalonador justo oferece uma divisão baseada

no peso e no tempo médio de utilização do processador por cada uma dos domínios;

- Conservativo x Não conservativo: Em algoritmos não conservativos, os pesos determinam a fatia de CPU correspondente a cada VM e, se alguma delas não utilizar a sua fatia, a CPU ficará ociosa durante este intervalo, mesmo que existam outros processos na fila de execução. Em escalonadores conservativos, o processador só ficará ocioso se não houver nenhum processo aguardando na fila de execução;

- Preemptivo x Não preemptivo: No escalonamento preemptivo, o algoritmo recalcula a decisão de escalonamento sempre que um processo passa ao estado "pronto". Se esse processo possuir uma prioridade maior que o processo que está sendo executado, o escalonador suspende o processo em execução e executa o processo de maior prioridade, que é a ação de preempção. Um algoritmo não-preemptivo permite a todos os processos usarem todo o tempo de CPU à eles alocados. Nesse modo, todas as decisões de alocação de CPU são feitas somente quando um processo cede a CPU, seja voluntariamente ou devido ao término da fatia de tempo alocada. Um algoritmo preemptivo é a melhor opção para aplicações intensivas em operações de entrada e saída, já que estas aplicações costumam ficar bloqueadas a espera de I/O e podem ser prejudicadas ao concorrer com aplicações intensivas em processamento.

O algoritmo BVT (*Borrowed Virtual Time*) implementa um escalonador preemptivo, justo e conservativo. O algoritmo SEDF (*Simple Earliest Deadline First*) consiste em um escalonador preemptivo que pode ser configurado para trabalhar em modo conservativo ou não conservativo.

O algoritmo *Credit Scheduler* [13], atualmente o escalonador padrão do Xen, implementa um escalonador preemptivo, de divisão justa e conservativo. O algoritmo funciona da seguinte forma: cada CPU gerencia uma fila local de execução contendo vCPUs executáveis e ordenadas pela suas prioridades. As prioridades das vCPUs podem assumir dois valores: *over* ou *under*. Se a vCPU possui créditos, a sua prioridade é *under*. A medida que ela é executada, a vCPU consome créditos. Quando

os créditos acabam e ficam negativos a sua prioridade é considerada *over*. Ao inserir uma vCPU em uma fila de execução, ela é posicionada atrás de todas as vCPUs de mesma prioridade.

Em cada CPU, as decisões de escalonamento são tomadas quando uma vCPU bloqueia, termina a sua fatia de tempo ou acorda. Se a CPU não encontrar uma vCPU de prioridade *under* na sua fila de execução, ela procura por uma na fila de execução dos outros processadores. Da mesma forma, se não houver nenhuma vCPU na sua fila de execução, ela procura vCPUs executáveis na fila dos outros processadores. Isto garante que nenhuma CPU fique ociosa enquanto há trabalho a ser executado no sistema.

Vale ressaltar que o Domínio-0 está sujeito ao mesmo algoritmo de escalonamento que os domínios não privilegiados. Logo, deve ser atribuído um peso suficientemente alto ao Domínio-0, visto que ele tem que ser capaz de responder às solicitações de E/S de todos os outros domínios.

2.1.3 Memória

O compartilhamento de memória é considerado como o recurso mais difícil de se adaptar ao contexto da para-virtualização [9], tanto em relação aos mecanismos a serem implementados no hipervisor, quanto nas modificações necessárias para adaptar o sistema operacional convidado. No Xen, a virtualização de memória é feita alocando fatias da memória RAM às máquinas virtuais.

A documentação do Xen [14] descreve como calcular a quantidade de memória que deverá ser reservada para o hipervisor Xen e para o Domínio-0. Em geral, o hipervisor Xen e suas ferramentas de sistema ocupam aproximadamente 128 MB de RAM. Já a memória utilizada pelo Domínio-0 depende da quantidade de memória física que o computador possui. Para computadores de até 32 GB de memória RAM, a memória do Domínio-0 utilizada para gerenciar os outros domínios é, normalmente inferior a 800 MB [14]. Assim, é preciso levar em conta esses valores para realizar a correta alocação de memória entre as máquinas virtuais.

A partir da versão 3, a funcionalidade de balão de memória foi adicionada ao Xen. O balão de memória é uma técnica na qual o *host* instrui a máquina virtual a liberar parte da memória que lhe foi alocada para que seja usada com outro fim. Esta técnica permite alocar ou desalocar memória das máquinas virtuais, sem a necessidade de pausar ou reiniciar os domínios. No arquivo de configuração das máquinas virtuais, é possível definir a quantidade de memória que será alocada à máquina virtual no momento da sua criação. Neste mesmo arquivo, também é possível determinar o valor máximo de memória que poderá ser alocado à máquina virtual. O intervalo entre o valor de memória e de memória máxima é a margem em que o sistema pode realizar o balão.

O *driver* do balão encontra-se na máquina virtual, entretanto ele é controlado pelo hipervisor [15]. Quando o balão infla criando pressão de memória na VM as rotinas de gerenciamento de memória do sistema operacional convidado devem liberar o espaço de memória necessário para satisfazer a solicitação de alocação do *driver*. A escassez de memória pode exigir que o sistema operacional convidado utilize memória virtual. As páginas liberadas são passadas para o hypervisor que por sua vez as disponibiliza para outras VMs. A fim de garantir a separação entre VMs, as páginas são zeradas antes de serem disponibilizadas novamente. A quantidade de memória que poderá ser liberada através do balão é obtida através do arquivo */proc/meminfo*.

A técnica de balão pode ser útil em diversas situações como, por exemplo, em ambientes que exigem uma alta densidade de máquinas virtuais, o balão de memória pode ser usado para aumentar a concentração de domínios virtuais nas máquinas físicas. A desvantagem em utilizar essa técnica é que a redução de memória de um domínio virtual pode impactar negativamente no desempenho das aplicações em execução, principalmente nas aplicações intensivas em memória.

A documentação do Xen [16] recomenda que servidores dedicados à virtualização aloquem uma quantidade fixa de memória para o Domínio-0, desabilitando a opção de balão de memória. Essa recomendação é feita por dois motivos principais. O primeiro é que o kernel do Linux calcula vários parâmetros relacionados à rede,

com base na quantidade de memória do Domínio-0 no instante de inicialização. O segundo é porque o Linux precisa de um espaço de memória para armazenar os metadados de memória e essa alocação também é baseada na quantidade de memória do Domínio-0 no momento de inicialização. Logo, se o sistema for inicializado com toda a memória física alocada ao Domínio-0, a cada vez que um novo domínio for iniciado, haverá uma redução da memória do Domínio-0. Essa redução implica que os parâmetros calculados no instante de inicialização não corresponderão mais ao cenário existente e uma parte da memória será desperdiçada no armazenamento de metadados de uma parte da memória que o Domínio-0 não possui mais.

2.1.4 Migração de máquinas virtuais

A migração é uma técnica que permite a transferência de uma máquina virtual de uma máquina física origem para outra máquina física destino, sem a necessidade de desligar ou reiniciar a máquina virtual. Para que a migração ocorra, as duas máquinas físicas envolvidas devem estar executando o Xen e o computador de destino deve ter recursos suficientes para acomodar a máquina virtual que será migrada. Além disso, quando uma máquina virtual é migrada, os seus endereços MAC e IP permanecem inalterados, logo, só é possível a migração de máquinas virtuais dentro da mesma sub-rede IP e dentro do mesmo espaço de endereçamento da camada 2.

A migração é uma importante forma de gerenciamento dos recursos computacionais em sistemas virtualizados. Com a popularização da tecnologia de virtualização, o uso da migração vem se tornando cada vez mais comum para realizar a distribuição e o balanceamento de cargas de trabalho, visando melhorar o desempenho de aplicações e otimizar o uso dos recursos em centros de dados.

No Xen, existem duas principais estratégias de migração. A abordagem convencional é baseada no princípio suspender-copiar-retomar, ou seja, a máquina virtual é pausada, o estado da máquina é copiado e transmitido ao computador de destino, onde a execução da máquina virtual é então retomada. Esta estratégia é de fácil implementação, mas pode comprometer seriamente o funcionamento de aplicações executadas nesta máquina virtual, como por exemplo, aplicações de roteamento [17][18]

e aplicações multimídias [19]. A segunda estratégia de migração é chamada de migração ao vivo e permite migrar máquinas virtuais sem causar uma longa interrupção dos serviços em execução.

Existem duas abordagens [20] usuais para realizar a migração ao vivo:

- Pré-cópia: A pré-cópia acontece em duas fases, chamadas de fase de "aquecimento" (*warm-up*) e fase "parar-e-copiar" *stop-and-copy*. Durante a fase de "aquecimento", as páginas de memória da máquina virtual são copiadas para o destino, enquanto a VM ainda está em execução na máquina de origem. Se alguma página de memória for modificada durante o processo, é dito que as páginas estão "sujas" e deverão ser reenviadas. Esta fase continua até que a taxa de páginas reenviadas seja menor do que a taxa em que as páginas são modificadas. Após a fase de "aquecimento", a VM será pausada no computador de origem, as páginas sujas restantes são copiadas para o destino e a execução da VM será retomada no computador de destino. O tempo entre a pausa da VM no computador original e retomada de execução no destino é chamado de *down-time*, podendo variar de alguns milissegundos a segundos de acordo com o tamanho da memória e aplicações em execução na VM.

- Pós-cópia: Nesta implementação de migração ao vivo, a máquina virtual é suspensa no computador de origem, um subconjunto mínimo do estado de execução da máquina virtual (registros da CPU e da memória não paginada) é transferido para o computador de destino. A execução da máquina virtual é, então, retomada no novo computador hospedeiro, ainda que a maior parte do estado da memória da VM ainda esteja na origem. No computador de destino, quando a VM tenta acessar páginas que ainda não foram transferidas são geradas falhas que são redirecionadas para o computador de origem através da rede. O *host* de origem, enviando a página que estava faltando. Uma vez que cada falha de página da máquina virtual em execução é redirecionada para a fonte, esta técnica pode degradar o desempenho de aplicativos que rodam na máquina virtual.

Durante a migração ao vivo, os Domínios-0 das duas máquinas físicas coordenam a transferência das páginas de memória e outras informações relativas à máquina virtual. O tempo de migração é influenciado pelo uso de CPU dos Domínios-0, pela quantidade de informações a serem enviadas e pela banda de rede disponível.

Capítulo 3

O Ambiente Virtualizado de Testes

O objetivo deste capítulo é descrever a plataforma de testes FITS na qual o sistema de gerência de recursos vai ser integrado, descrever algumas ferramentas utilizadas no desenvolvimento desse projeto, assim como alguns trabalhos relacionados nos quais este trabalho foi baseado.

3.1 A plataforma de Testes FITS

A Internet tem crescido sistematicamente e cada vez mais surgem novas aplicações e serviços baseados na Web. Com a diversificação de aplicações, novos requisitos de segurança, mobilidade e qualidade de serviço começam a surgir [21]. Devido à dificuldade em atender esta nova demanda com a estrutura atual da Internet, muito esforço de pesquisa [22, 23] tem sido empregado para propor novas soluções visando construir a "Internet do Futuro".

O Grupo de Teleinformática e Automação da COPPE/UFRJ desenvolveu a plataforma de testes FITS (Future Internet Testbed with Security)[5] que consiste em um ambiente de teste que permite realizar experimentos e validar propostas ligadas a Internet do Futuro. A plataforma segue a abordagem pluralista [21], cuja principal característica é a capacidade de suportar simultaneamente múltiplas pilhas de protocolos, garantindo a dinamicidade e a capacidade de evoluir da nova arquitetura.

O FITS é uma rede de teste, cujos elementos de rede estão geograficamente distribuídos em ilhas dentro de campi universitários e interligados através da Internet. Para isto, o projeto conta com a colaboração de diversas universidades brasileiras e europeias que mantêm as ilhas funcionando em suas instalações. A plataforma de testes é baseada na virtualização de redes e utiliza o hipervisor Xen para realizar a virtualização de roteadores e a plataforma OpenFlow [24] para gerenciar o fluxo de dados. O FITS possui uma interface Web que torna o gerenciamento das redes de teste intuitivo e facilita a manipulação e configuração dos elementos de rede.

O FITS permite a criação de redes virtuais isoladas entre si, com diferentes especificações de níveis de serviço, tais como: i) migração das redes virtuais entre as máquinas do FITS, sem perda de pacotes ou interrupção dos experimentos; ii) segurança, o FITS utiliza o paradigma de chaves públicas para certificar todos os servidores físicos e máquinas virtuais disponíveis no testbed; iii) autenticação de usuários baseada em cartões inteligentes *smartcards* e *OpenId*. Na rede de testes é possível instanciar, migrar e remover fluxos em redes OpenFlow, e redes virtuais formadas por roteadores virtuais Xen. A administração da rede, é realizada por uma interface Web que permite o acesso as configurações tanto de nós OpenFlow, como de nós Xen.

3.2 A Libvirt

A libvirt [25] é uma API para gerenciar, de forma segura, máquinas em ambientes virtualizados. A libvirt permite o gerenciamento completo das máquinas virtuais, tornando possível criar, modificar, controlar, migrar e parar as máquinas através de sua interface. A ferramenta é implementada em C, mas inclui suporte para Python, Ruby, Java, Perl e OCaml.

A libvirt permite o acesso a hipervisores utilizando conexões autenticadas e criptografadas, através do protocolo *Transport Layer Security* (TLS). A criptografia e a autenticação são feitas usando a criptografia assimétrica e infraestrutura de chave pública (PKI), logo, é necessário que os computadores possuam certificados válidos, além das chaves pública e privada.

3.3 Gerência de recursos em ambientes virtualizados

Através da virtualização, é possível dividir grandes servidores físicos subutilizados em diversos servidores virtuais de menor capacidade, oferecendo uma gama maior de aplicações e serviços. Normalmente, a qualidade dos serviços prestados está especificada em contratos de acordos de nível de serviço (*Service Level Agreement* - SLA), na qual a empresa se compromete a garantir recursos suficientes para atender a demanda do serviço contratado. Observa-se então, que a gerência de um *data center* é uma tarefa complexa que exige um balanço entre o nível de utilização dos computadores e o cumprimento dos SLAs, ou seja, é desejável ter o maior emprego possível dos recursos de hardware sem comprometer o nível de serviço estabelecido nos contratos.

O planejamento da alocação de recursos ainda traz outros desafios devido ao *overhead* causado pela camada de virtualização [26] que depende do tipo e da tecnologia de virtualização adotada. No Xen, por exemplo, aplicações intensivas em operações de entrada e saída podem causar um overhead de processamento considerável, visto que há um consumo de processamento por parte da máquina virtual e do Domínio-0 a cada operação de E/S. Para realizar a consolidação de servidores, é necessário, então, realizar um planejamento e uma análise das cargas de trabalho envolvidas em todos os serviços prestados. A solução mais trivial para esse problema é alocar, para cada máquina virtual, a quantidade máxima de recursos que ela está autorizada a utilizar. Entretanto, apesar da simplicidade desta implementação, ela pode implicar em um excesso de aprovisionamento de recursos, deixando grande parte da capacidade computacional subutilizada.

Uma abordagem mais eficiente para realizar a consolidação de servidores utiliza a funcionalidade de migração de máquinas virtuais. Através dessa técnica, é possível reorganizar os domínios nos nós físicos de acordo com a disponibilidade dos recursos. Assim, se houver um aumento significativo na carga de uma máquina virtual, de forma a sobrecarregar um nó físico, esse domínio poderá ser migrado para outro computador com mais recursos disponíveis. Essa abordagem permite realizar a

distribuição de cargas de uma forma eficiente, porém também trás alguns desafios. O primeiro consiste em prever o comportamento futuro das máquinas virtuais para estimar a demanda futura de recursos. O segundo consiste em encontrar a melhor combinação entre máquinas físicas e virtuais que é um problema de otimização do tipo *bin-packing*, cuja complexidade é *NP-hard*. Logo, dada a complexidade do problema, os trabalhos desenvolvidos nesta área são baseados em heurísticas e modelos estatísticos.

3.3.1 Trabalhos relacionados

Esta seção, apresenta uma breve descrição dos trabalhos relacionados que serviram de base para o desenvolvimento desse projeto. De uma forma geral, os trabalhos de balanceamento e distribuição de cargas em ambientes virtualizados se diferenciam principalmente em três características:

- Estado local ou global - Alguns algoritmos analisam a carga individual dos nós físicos ([27, 28, 29]), outros analisam a carga total do conjunto de máquinas e a diferença entre as cargas dos nós físicos ([30]);
- Afinidade entre as máquinas - Algumas das soluções utilizam como critério de decisão, o perfil das máquinas e a afinidade entre eles [31, 32]. A afinidade pode ser definida como a capacidade das máquinas virtuais em compartilharem uma mesma máquina física. Por exemplo, duas máquinas com uso intensivo de CPU possuem baixa afinidade entre si, pois disputariam a utilização do mesmo recurso. Outras soluções se preocupam somente com o nível de utilização dos recursos de hardware das máquinas físicas;
- Topologia de rede - Alguns trabalhos levam em consideração a topologia da rede e a banda disponível para realizar a migração [33].

3.3.1.1 Sandpiper

Sandpiper [27] é uma ferramenta de gerenciamento de recursos para *data centers*, composto basicamente de três componentes: um mecanismo de criação de perfis, um detector de sobrecarga e um gerenciador de migração. A criação de perfis é feita através do monitoramento e da coleta de estatísticas do uso de recursos das

máquinas físicas e virtuais. O detector de sobrecarga monitora estes perfis em busca de nós sobrecarregados que consistem em máquinas físicas cuja utilização de pelo menos um dos recursos ultrapassou um limiar ou se ocorreu a violação de SLA. O detector de sobrecarga determina quando é necessário realizar uma realocação de recursos, ou seja, quando o gerenciador deverá atuar para eliminar esta sobrecarga. Cabe ao gerenciador decidir qual máquina virtual migrar e para qual servidor irá recebê-la.

A decisão de qual máquina migrar é feita utilizando a métrica *Volume*, representado pela Equação 3.1, onde *cpu*, *net*, *mem* representam respectivamente o uso de CPU, rede e memória. O gerenciador seleciona para a migração a máquina virtual com maior relação *volume/mem* da máquina física de maior *volume*. Essa máquina é migrada para a máquina física de menor volume.

$$Vol = \frac{1}{1 - cpu} \cdot \frac{1}{1 - net} \cdot \frac{1}{1 - mem} \tag{3.1}$$

O Sandpiper utiliza o estado local de cada máquina física e não leva em conta a afinidade entre as máquinas e a topologia da rede para realizar as migrações. A proposta foi implementada utilizando o hipervisor Xen e os autores desta ferramenta apresentam duas abordagens ao problema. A primeira é uma abordagem do tipo caixa preta, onde o monitoramento é realizado sem a instalação de nenhum software específico nas máquinas virtuais, coletando apenas informações disponíveis a partir do exterior das máquinas virtuais. A segunda proposta utiliza uma abordagem do tipo caixa cinza, a qual obtém as informações através de um *daemon* de monitoramento instalado em cada máquina virtual. Os resultados mostram que a abordagem do tipo caixa preta, é capaz de eliminar sobrecargas simultâneas envolvendo múltiplas máquinas físicas, porém a utilização de um sistema baseado na abordagem de caixa cinza foi capaz de melhorar o tempo de resposta do sistema, permitindo, sobretudo um melhor monitoramento da memória.

3.3.1.2 Voltaic

Voltaic [28] é um sistema de gerência voltado para computação em nuvem que visa garantir o cumprimento de acordos de níveis de serviços (SLAs) e otimizar o uso dos

recursos computacionais. Assim como o Sandpiper, o Voltaic também é uma solução baseada em perfis de uso. Porém, ao invés da métrica *volume* utilizada no Sandpiper, o Voltaic utiliza uma métrica própria chamada *cargadosistema*, baseada em lógica fuzzy, cujo conjunto de regras de inferência é de responsabilidade do gerente da nuvem.

O sistema pode ser dividido em três módulos principais: o primeiro, chamado de coletor de estatísticas, é responsável por coletar informações de uso dos recursos computacionais; o segundo consiste no analisador de perfil que utiliza os dados provenientes do coletor de estatística para gerar perfis de uso; O terceiro é o orquestrador, módulo central do Voltaic, responsável por gerenciar as máquinas, detectar escassez de recursos e orquestrar a migração.

O orquestrador monitora a carga de todas as máquinas físicas e caso a média das últimas amostras de carga de alguma das máquinas ultrapassarem um limiar de segurança, o algoritmo começa o procedimento de realocação de recursos. As máquinas físicas são ordenadas de acordo com a *cargadosistema* e pelo número de máquinas virtuais críticas que ela possui. A criticidade das máquinas virtuais é calculada de acordo com a métrica de criticidade expressa pela Equação(3.2)

$$criticidade = \alpha \cdot [carga\ virtual] + (1 - \alpha) \cdot [1 - abs(correlação)]. \tag{3.2}$$

Dada a sobrecarga de algum dos recursos monitorados, a máquina virtual escolhida para a migração será a máquina mais crítica do servidor físico mais sobrecarregado. O servidor de destino será aquele que possuir recursos disponíveis compatíveis com o perfil da máquina virtual.

Esta ferramenta foi testada utilizando um simulador de ambiente em nuvem, desenvolvido pelo próprio autor, que permite testar a proposta em um cenário com uma grande quantidade de computadores e máquinas virtuais, além da possibilidade de incluir outros parâmetros, como, por exemplo, temperatura. Os resultados obtidos mostram uma redução de mais de 10% no total de ciclos perdidos, obtida devido à correta alocação dos recursos, além de apresentar um melhor desempenho quando comparado ao algoritmo do Sandpiper.

Capítulo 4

O Sistema Proposto

Os capítulos anteriores apresentaram o contexto do projeto, as ferramentas que serão utilizadas para a sua implementação e os trabalhos relacionados. Neste capítulo, é descrito em detalhe o trabalho realizado nesse projeto que consiste em um sistema de gerência de recursos computacionais para ambientes virtualizados.

O sistema proposto visa monitorar e gerenciar as máquinas físicas e virtuais de forma a melhorar a utilização dos recursos físicos e garantir o cumprimento dos níveis de serviço dos servidores virtuais. A ideia central da proposta é criar perfis de uso de diferentes recursos para todas as máquinas físicas e virtuais de um cluster e, com base nestes perfis, detectar nós físicos sobrecarregados e distribuir a carga de trabalho através da migração ao vivo.

A proposta foi implementada e testada utilizando a plataforma de teste FITS que, como descrito na Seção 3.1, é apropriada para avaliar propostas ligadas a Internet do Futuro. Este projeto contribui com o FITS oferecendo um sistema de monitoramento em tempo real dos recursos das máquinas pertencentes à plataforma. O monitoramento de recursos é uma funcionalidade fundamental, pois permite que os usuários da plataforma possam visualizar de forma simples a utilização dos recursos computacionais, auxiliando o desenvolvimento e a validação das propostas testadas na plataforma. Apesar do sistema proposto ter sido implementado no FITS, o seu uso não é restrito a esta plataforma em particular, pois pode ser utilizado em qualquer ambiente virtualizado que utilize tecnologias de virtualização compatíveis com a API da libvirt.

4.1 A Arquitetura do Sistema Proposto

Como ilustrado na Figura 4.1, o sistema desenvolvido pode ser dividido em três módulos principais: o Gerador de perfil, responsável por monitorar o uso dos recursos físicos; o Detector de sobrecarga, capaz de detectar escassez de recursos monitorados; o Orquestrador de migração, responsável por tomar decisões visando distribuir as cargas de trabalho em caso de sobrecarga. Pode-se observar que a comunicação entre o gerenciador e as máquinas físicas é toda realizada através da libvirt.

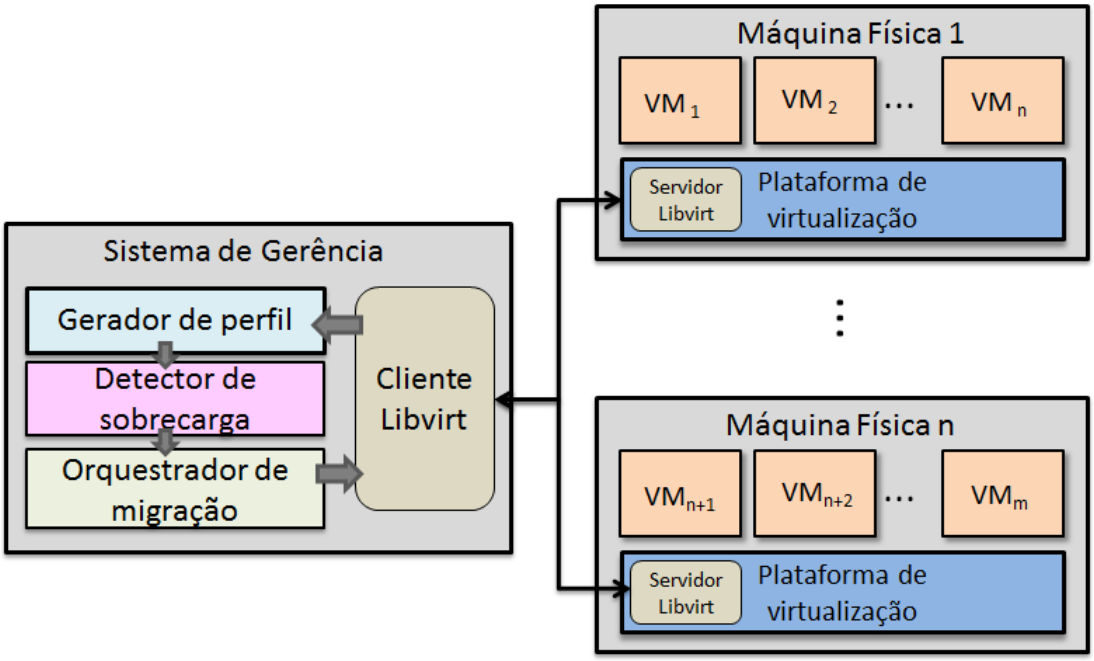


Figura 4.1: Arquitetura do sistema de gerência de recursos proposto: o gerador de perfis, o detector de sobrecarga e o orquestrador de migração. Monitoramento através da libvirt.

Para o funcionamento adequado do sistema, é necessário que os computadores monitorados possuam configurações homogêneas de hardware e software, pois, em caso de migração, configurações distintas podem implicar em perda de desempenho na execução das máquinas virtuais. Por exemplo, em um cenário onde os servidores possuem alto nível de utilização, se uma máquina virtual com uso intensivo de rede é migrada para uma máquina física com configurações de escalonamento diferentes da máquina física de origem, as aplicações sensíveis a latência podem ter o seu funcionamento fortemente prejudicado [11], [19]. Outro exemplo seria a migração entre

máquinas com processadores de modelos e frequências diferentes. Vale ressaltar que a migração entre máquinas de diferentes configurações poderiam até interferir positivamente no funcionamento da máquina virtual, porém uma solução de migração deste tipo requer uma modelagem diferente, com um maior número de variáveis, o que está fora do escopo deste trabalho.

O sistema desenvolvido é baseado na funcionalidade de migração ao vivo do Xen, logo, como detalhado na Seção 2.1.4, é necessário que todas os nós físicos monitorados estejam na mesma rede para que as imagens dos discos estejam disponíveis a todas as máquinas do sistema. Além disso, também é necessário que todas as máquinas físicas possuam a libvirt instalada assim como os certificados necessários para autenticar as conexões.

A seguir, serão apresentadas breves descrições de cada um dos módulos do sistema proposto.

4.1.1 O Gerador de Perfil

O mecanismo de monitoramento foi desenvolvido utilizando uma abordagem do tipo caixa preta. Na solução adotada, todas as informações relativas ao uso de recursos das máquinas virtuais são obtidas através da libvirt, sem a instalação de nenhum programa específico nas máquinas virtuais. Essa abordagem foi escolhida por garantir uma menor interferência no funcionamento dos domínios, assegurando maior autonomia e privacidade aos proprietários das máquinas virtuais. Estas características são importantes em ambientes virtualizados, pois, visto que em muitas soluções comerciais baseadas em virtualização, as máquinas virtuais não pertencem aos proprietários do hardware, não é viável a instalação de softwares de monitoramento nos domínios de clientes. Logo, o Gerador de perfil interage com as máquinas físicas através da libvirt, como mostrado na Figura 4.1. Essa conexão é feita utilizando o protocolo *Transport Layer Security* (TLS), que consiste numa conexão TCP/IP autenticada e criptografada, garantindo a segurança das informações. A coleta de dados é realizada através da amostragem periódica do uso de CPU, da quantidade de memória alocada e da quantidade de dados de rede enviados e re-

cebidos pela máquina. Com as amostras obtidas, são criados três perfis de uso, representando cada uma das três métricas, que consistem em séries temporais representada por uma janela deslizante contendo N amostras. O tamanho da janela a ser considerado depende das aplicações executadas, pois há aplicações cuja demanda por recursos varia de acordo com a hora do dia, outras com o dia da semana e etc. O tamanho utilizado nesse projeto é 100, pois de acordo com [28] esse número de amostras é suficiente para representar adequadamente os perfis.

A coleta dos dados de uso dos recursos é feita de modo paralelo através da utilização de processos leves, chamados *threads*. Dessa forma, é possível construir simultaneamente o perfil das diferentes máquinas monitoradas, aproveitando melhor os recursos computacionais da máquina gerenciadora e permitindo maior coerência temporal das amostras obtidas. Cada elemento do sistema possui um *thread* responsável pela coleta de dados e construção dos seus perfis de uso. Os perfis criados são apresentados em um painel de controle através de gráficos que indicam a participação de cada máquina virtual, inclusive do Domínio-0, na utilização dos recursos das máquinas físicas monitoradas.

4.1.1.1 Uso de CPU

O monitoramento de CPU das máquinas virtuais é realizado através da amostragem da medida *tempo de CPU* obtida diretamente da libvirt, que corresponde ao tempo de processador utilizado pela máquina virtual desde a sua criação. Entretanto, não é possível obter a medida *tempo de CPU* das máquinas físicas através da libvirt, assim o perfil de uso do processador dos nós físicos é calculado indiretamente, através da soma dos perfis de todos os domínios que nele executam. Esta medida possui uma imprecisão associada, pois as amostras das máquinas virtuais podem não corresponder a exatamente o mesmo instante de tempo, logo é esperado que o erro da medida aumente a medida que o número de máquinas virtuais na mesma máquina física aumente.

4.1.1.2 Memória

Os perfis de memória das máquinas virtuais são construídos a partir da amostragem da quantidade de memória alocada para cada domínio virtual. Para os perfis das máquinas físicas, é utilizada a quantidade de memória livre. Ambas as medidas são obtidas diretamente da libvirt.

4.1.1.3 Rede

Como apresentado na Seção 2.1.1, o Domínio-0 é responsável por realizar as operações de rede pelos outros domínios não privilegiados. Logo, uma forma simples de obter informações que representem a atividade de rede das máquinas virtuais é medindo a quantidade de dados enviados e recebidos através de suas interfaces virtuais, facilmente obtida através da libvirt. Essa métrica é útil para identificar a contribuição de cada máquina virtual no tráfego total gerenciado pelo Domínio-0. Então, é possível identificar se a causa desse consumo provém da utilização intensiva de rede e qual a participação de cada máquina virtual no tráfego total gerenciado pelo Domínio-0.

4.1.2 Detector de sobrecarga

No sistema desenvolvido são considerados dois tipos de sobrecarga, a de processamento e a de memória. A sobrecarga de CPU acontece quando a utilização do processador ultrapassa um determinado limiar durante as k últimas amostras. O conceito de sobrecarga de memória é similar e ocorre quando a quantidade de memória alocada de um nó físico ultrapassa o limiar definido para memória durante as últimas k amostras. Os valores de k podem ser escolhidos de forma a obter uma política de detecção de sobrecarga “agressiva” ou “conservadora”. Pequenos valores de k representam uma política agressiva porque utilizam apenas poucas amostras para classificar um elemento como sobrecarregado, enquanto grandes valores representam uma abordagem mais conservadora, pois um nó só será considerado sobrecarregado se o consumo de recursos permanecer acima do limiar durante um grande intervalo de tempo.

Apesar do sistema monitorar a intensidade do tráfego de rede, não há uma sobrecarga associada a esse perfil, uma vez que a métrica de rede implementada não leva em conta os parâmetros nem a topologia da rede de computadores. A medida é utilizada, então, para verificar o impacto que o tráfego de rede dos domínios causa no uso de processamento do Domínio-0.

Um nó físico é considerado sobrecarregado se pelo menos um dos recursos monitorados estiver escasso naquele nó. A detecção de sobrecarga funciona como descrito no Algoritmo 1. A cada intervalo de tempo T, o algoritmo percorre a lista de máquinas físicas, analisando os perfis e procurando sobrecargas no sistema. O algoritmo constrói duas listas, uma contendo máquinas físicas sobrecarregadas e outra contendo máquinas com recursos disponíveis e ordena-as de acordo com o nível de utilização de cada um dos recursos monitorados.

Algorithm 1: Detecção de sobrecarga.

```
while monitorando do
    for PM in máquinasFísicas do
        if PM.sobrecarregada == True then
            listaMáquinasSobrecarregadas.append(PM);
        else
            listaMáquinasDisponíveis.append(PM);
        end if
    end for
    listaMáquinasSobrecarregadas.sort();
    listaMáquinasDisponíveis.sort();
    sleep(T);
end while
```

4.1.3 Orquestrador de migração

Quando é detectada a escassez de algum dos recursos monitorados, o Orquestrador de migração é acionado e deve tomar decisões visando eliminar a sobrecarga e melhorar a distribuição das cargas de trabalho. O algoritmo realiza, no máximo, uma migração a cada iteração, então é preciso: i) selecionar uma máquina física

sobrecarregada que é candidata a ter sua carga diminuída, ii) um domínio virtual desta máquina física que deve ser migrado e iii) uma máquina física de destino com recursos disponíveis para receber esse domínio. Assim, a escolha dos elementos envolvidos na migração é feita da seguinte forma:

- Seleção da máquina física sobrecarregada - Como descrito anteriormente, as máquinas físicas são classificadas de acordo com o nível de utilização de CPU e alocação de memória. Assim, se acontecer de diversos nós estarem sobrecarregados com os dois tipos de sobrecarga, a máquina física selecionada será a com maior nível de utilização do processador. Caso a sobrecarga seja apenas de memória, a máquina escolhida é aquela com menos memória disponível;

- Seleção da máquina física de destino - A máquina física candidata a receber a máquina virtual migrada é aquela que possuir a maior quantidade de recursos disponíveis, suficientes para alocar a máquina virtual sem ficar sobrecarregada;

- Seleção da máquina virtual - Caso haja uma sobrecarga de CPU, é escolhida, dentre os domínios da máquina física sobrecarregada, a máquina virtual mais crítica de acordo com a Expressão 4.1, onde *cpu* representa o uso de CPU, *net* indica o tráfego de rede, $cpuDomain0$ representa o uso do processador pelo domínio-0 e memória a quantidade de memória alocada ao domínio. O termo $net \cdot cpuDomain0$ visa incluir na fórmula de criticidade o *overhead* causado no Domínio-0 pelo uso da rede [34].

  Para migrar uma máquina virtual entre dois computadores, é preciso transmitir as páginas usadas de memória à máquina de destino, logo o tempo de migração é diretamente influenciado pela quantidade de dados a serem transmitidos e consumindo recursos de processamento tanto na máquina física de origem e quanto na de destino [35]. Por isto, optou-se pela relação $\frac{cpu}{\text{memória}}$ na expressão de criticidade.

  Caso a sobrecarga seja de memória, é selecionada como candidata a migração a máquina virtual que possuir a menor quantidade de memória alocada do nó físico sobrecarregado. Com esta solução pode ser necessário a migração de mais de uma máquina virtual para eliminar a sobrecarga, entretanto ela

evita migrações longas e, com isso, pode diminuir o processamento total a ser realizado pelo Domínio-0 das máquinas físicas envolvidas na migração.

$$criticidade = \frac{cpu + net \cdot cpuDomain0}{\text{memória}} \tag{4.1}$$

O algoritmo do Orquestrador é ilustrado no Algoritmo 2. O algoritmo percorre a lista de máquinas virtuais do nó físico mais sobrecarregado buscando a máquina virtual a ser migrada. É selecionada a máquina virtual mais crítica que consiga ser migrada para a máquina física com mais recursos disponíveis. É possível notar que a sobrecarga de CPU é tratada com prioridade maior do que a sobrecarga de memória. Esta escolha foi feita porque o uso de CPU é mais dinâmico, já que o monitoramento de memória é realizado através da quantidade de memória alocada e não pela quantidade de memória usada. Nota-se também que se todas as máquinas físicas estiverem saturadas, ou se se a migração puder causar sobrecarga na máquina de destino, a migração não é realizada.

Algorithm 2: O Algoritmo do Orquestrador

```
if cpuOverchargedPms.lenght > 0 and cpuIdlePms.lenght > 0 then
    mostOverchargedPm = cpuOverchargedPms[0] ;
    leastOverchargedPm = cpuIdlePms [-1] ;
    for VM in mostOverchargedPm.VirtualMachines do
        if VM.memory + leastOverchargedPm.memory <
        memoryThreshold then
            if VM.cpuPrediction + leastOverchargedPm.cpuPrediction <
            cpuThreshold then
                migrate(VM, mostOverchargedPm, leastOverchargedPm);
                return ;
            end if
        end if
    end for
else
    if memOverchargedPms.lenght > 0 and memIdlePms.lenght > 0
    then
        mostOverchargedPm = memOverchargedPms[0] ;
        leastOverchargedPm = memIdlePms [-1] ;
        for VM in mostOverchargedPm.VirtualMachines do
            if VM.memory + leastOverchargedPm.memory <
            memoryThreshold then
                if VM.cpuPrediction + leastOverchargedPm.cpuPrediction
                < cpuThreshold then
                    migrate(VM, mostOverchargedPm,
                    leastOverchargedPm);
                    return ;
                end if
            end if
        end for
    end if
end if
```

O sistema proposto se diferencia do Voltaic e do Sandpiper em diversos aspectos, mas a principal diferença está nas decisões de migração. O Voltaic utiliza uma métrica chamada $cargadosistema$, enquanto o Sandpiper utiliza a métrica $Volume$, ambas usadas como parâmetro para realizar a distribuição de cargas e selecionar quais máquinas físicas participarão da migração. Essas duas medidas visam expressar, em um único valor, o nível de utilização dos diferentes recursos monitorados. No sistema proposto, as sobrecargas são tratadas isoladamente, de acordo com a sua prioridade. Sobrecargas de processamento são tratadas com maior prioridade que as de memória e as máquinas físicas escolhidas para migração são aquelas com o maior e o menor uso do recurso sobrecarregado.

4.2 Implementação do Sistema Proposto

A implementação de cada um dos módulos desenvolvidos foi realizada de acordo com as diretrizes da orientação a objetos, na linguagem de programação Python. Foram utilizados computadores padrão de mercado com placas de rede gigabit Ethernet.

O escalonador utilizado no Xen foi o *Credit Scheduler* e o *timeslice* de 30 ms, ambos são configurações padrão do Xen. A memória de inicialização do Domínio-0 foi definida como 2048 MB, mas a opção de balão foi habilitada e a memória mínima para este domínio foi configurada como 1024 MB. O balão de memória foi ativado nesse projeto, pois apesar de o sistema operacional debian weezy necessitar de mais de 1GB de memória para iniciar o sistema, o total de memória gasto pelo Domínio-0 e pelo hipervisor para gerenciar as máquinas virtuais está na ordem de 1GB. Então, visto que uma parte da memória alocada exclusivamente para o Domínio-0 ficaria ociosa durante todo o funcionamento do nó físico, optou-se por habilitar o balão de memória visando melhor aproveitar a memória disponível.

Para implementar todas as funcionalidades proposta para o sistema, foram utilizadas algumas bibliotecas em Python. A biblioteca *Threading* foi utilizada para a criação de *threads* e para gerenciar a concorrência dos processos através de mecanismos de exclusão mútua. Outra biblioteca importante é a *Matplotlib* que consiste

numa biblioteca de plotagem de duas dimensões (2D), de código aberto distribuída sob licença PSF. Com ela é possível diversos tipos de gráfico, como histogramas, espectros de potência, gráficos de barras, gráficos de dispersão e etc. Essa biblioteca foi utilizada para implementar o painel de controle.

O diagrama de classes da ferramenta desenvolvida é apresentado na Figura 4.2. Como pode ser observado, o sistema é composto por três classes:

- a classe MáquinaVirtual representa uma máquina virtual e contém todas as informações relativas ao domínio. Os métodos dessa classe permitem o monitoramento, a coleta de dados estatísticos e a criação do perfil de uso de recursos da máquina virtual;
- a classe MáquinaFísica representa uma máquina física e contém todas as informações relativa ao nó físico, como a conexão com o servidor da libvirt, as informações de CPU e memória, a lista de todos os domínios em execução na máquina. A classe também possui métodos que permitem criar o perfil da máquina física e gerenciar as máquinas virtuais;
- a classe Gerenciador é a classe central onde os algoritmos de gerência estão implementados. Ela é responsável por lançar o monitoramento das máquinas, por realizar a detecção de sobrecarga, por gerenciar o conjunto de máquinas e tomar decisões de migração.

Como mencionado anteriormente, a ferramenta desenvolvida utiliza paradigmas de computação paralela, sendo baseada na utilização de diversos processos leves executados paralelamente, cada um com uma tarefa específica. Cada objeto do tipo *MáquinaVirtual* possui um *thread* que monitora o consumo de recursos computacionais e cria os perfis de uso da máquina virtual, assim como cada cada instância da classe *MáquinaFísica* possui um processo leve para a construção do perfil da máquina física.

Os parâmetros do sistema que devem ser configurados, são:

1. o tamanho da janela de observação, ou seja, o número de amostras contidos no perfil;

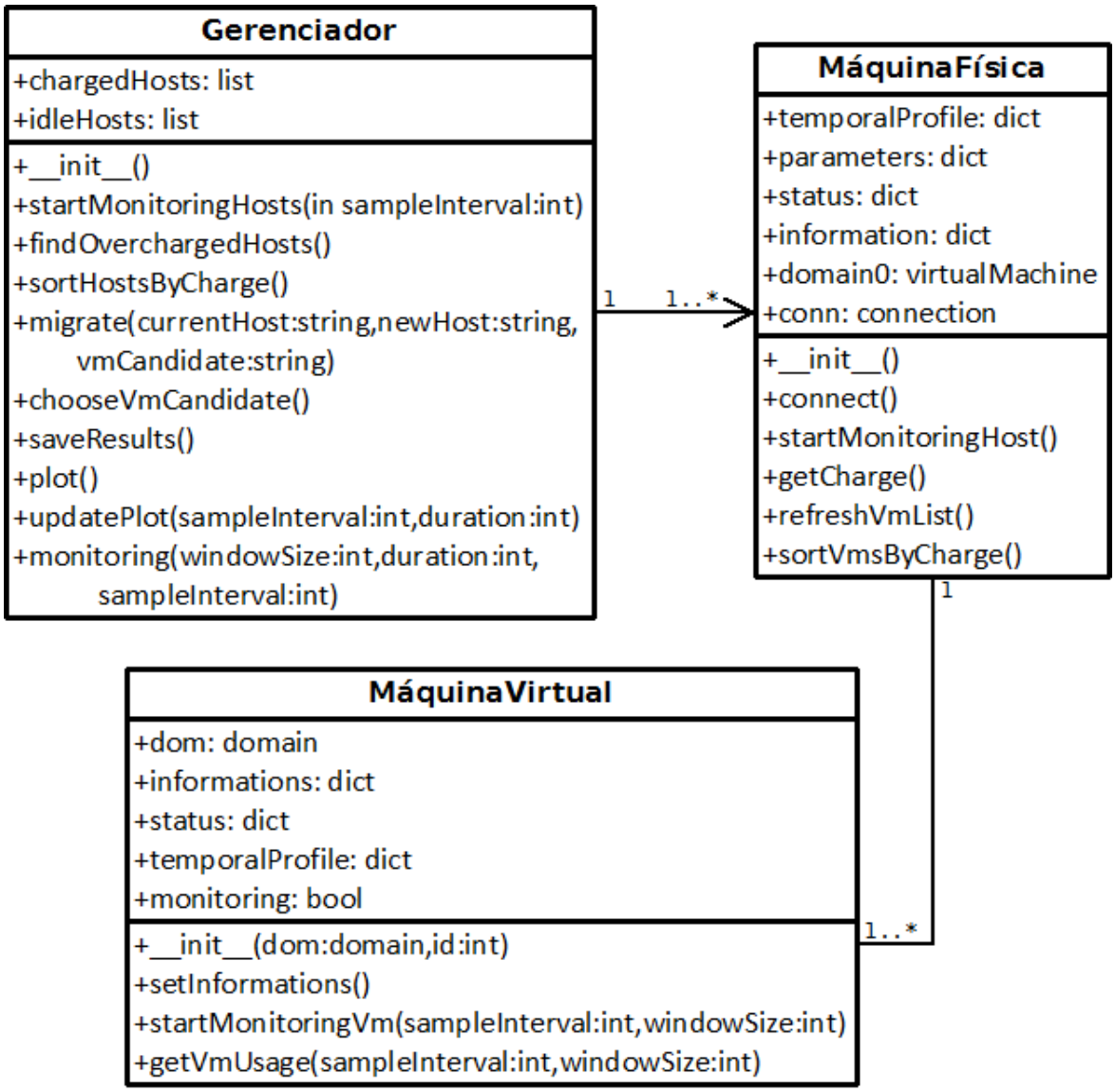


Figura 4.2: Diagrama de classe.

2. os limiares de sobrecarga de CPU e de memória;

3. o período de amostragem;

4. o número de amostras acima do limiar que devem ser consideradas antes de realizar a migração;

5. o nome das máquinas físicas a serem monitoradas.

4.2.1 Painel de controle

Foi desenvolvido, um painel de controle que permite uma fácil visualização do uso de recursos das máquinas físicas e virtuais em tempo real. O painel consiste em

gráficos mostrando a utilização de CPU, memória e rede. Este painel foi desenvolvido utilizando a biblioteca *Matplotlib*. A Figura 4.3 mostra o painel de controle em funcionamento com algumas máquinas virtuais de teste, utilizando duas máquinas da plataforma FITS.

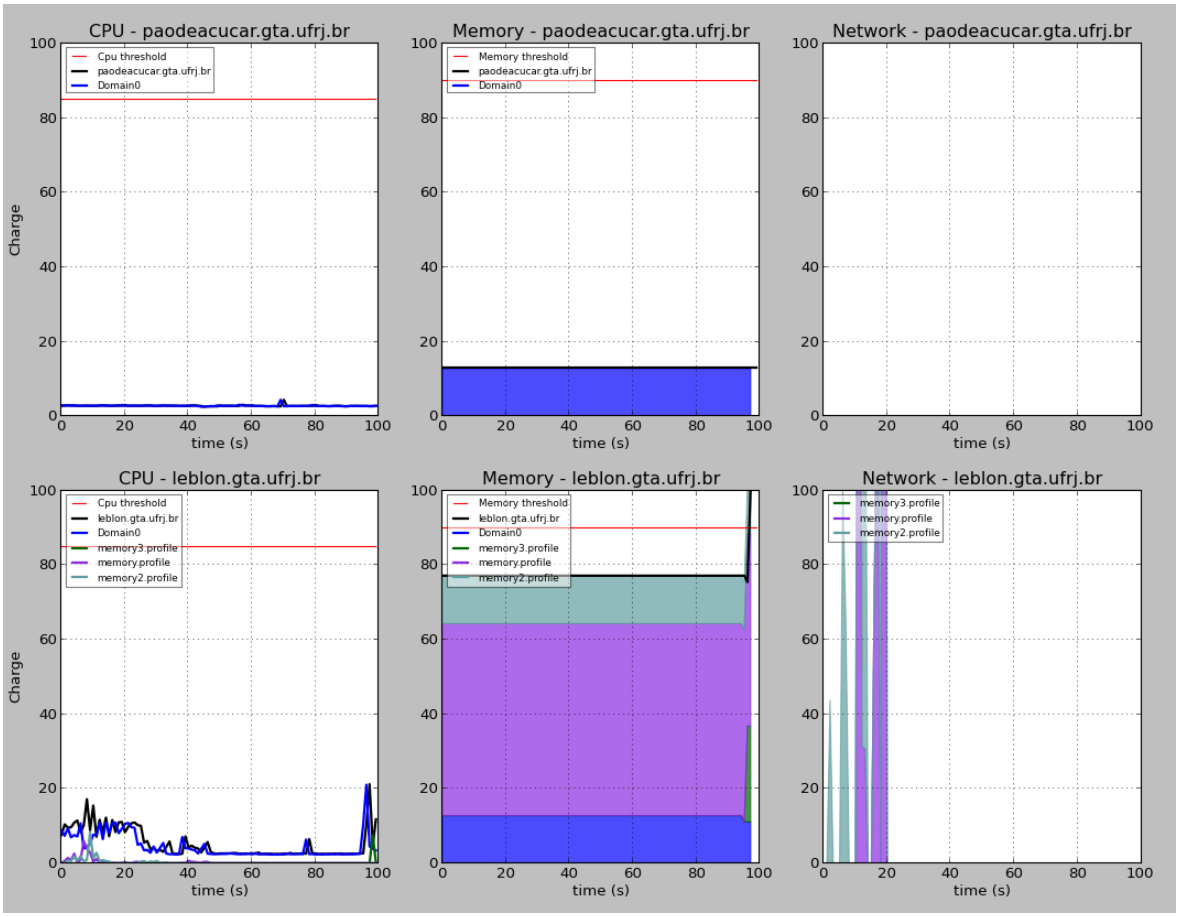


Figura 4.3: Painel de controle de visualização do uso de recursos.

Capítulo 5

Resultados

Para validar o funcionamento do sistema, foi desenvolvido um cenário de teste que permite avaliar o comportamento e a eficiência do algoritmo em diferentes casos de sobrecarga. Através deste cenário de teste também foi possível observar o comportamento do Domínio-0 durante as migrações e atividades de rede. A carga de CPU foi simulada utilizando o programa Stress [36], de licença livre, que permite gerar cargas de trabalho de forma controlada.

5.1 Cenário de teste

Conforme especificado na Seção 4.1, as máquinas monitoradas devem possuir configurações homogêneas de hardware. Dentre os computadores disponíveis na plataforma de teste FITS, foram selecionadas duas máquinas equipadas com processador Intel I7 de 3.2 GHz e 16 GB de memória RAM. O sistema operacional instalado nas máquinas é o Debian Weezy e o hipervisor Xen utilizado é a versão 4.1.3. As imagens dos discos e os arquivos de configuração encontram-se em um nó central da arquitetura do FITS e são acessíveis por qualquer máquina do testbed. O sistema de gerência foi instalado em uma máquina fora da plataforma FITS, mas com configurações de hardware idênticas às máquinas monitoradas e dentro da mesma sub-rede IP. A escolha de colocar o sistema inicialmente fora da plataforma de teste foi feita com o intuito de não interferir nos experimentos sendo realizados no FITS durante o desenvolvimento e testes desse sistema de gerência.

Tabela 5.1: Configurações das máquinas virtuais.

Nome	vCPUs	Memória (MB)
cpu1	8	512
cpu2	8	512
cpu3	8	512
memory1	1	8192
memory2	1	2048
memory3	1	4096

Para criar diferentes situações de sobrecarga foram utilizadas máquinas virtuais com diferentes configurações, de modo a consumir cada um dos recursos monitorados. A lista dos domínios virtuais utilizados e as suas respectivas configurações podem ser vistas na Tabela 5.1, onde os nomes dos domínios indicam qual recurso é mais utilizado por eles.

O cenário de teste é dividido nas seguintes etapas:

1. criação das máquinas memory1, memory2 na máquina física *leblon.gta.ufrj.br*;
2. criação da máquina memory3 na máquina *leblon.gta.ufrj.br*;
3. criação das máquinas cpu1 e cpu2 na máquina física *paodeacucar.gta.ufrj.br* e da máquina cpu3 na *leblon.gta.ufrj.br*;
4. destruição da máquina memory1 na máquina física *leblon.gta.ufrj.br* e da máquina memory2 na *paodeacucar.gta.ufrj.br*;
5. geração de carga em 5 vCPUs da máquina cpu2 e em 2 vCPUs do domínio cpu3.

A figura 5.1 mostra a configuração das máquinas *leblon.gta.ufrj.br* e *paodeacucar.gta.ufrj.br* em cada uma das etapas do cenário de testes.

A Figura 5.2 mostra o uso de memória e processador pelas duas máquinas físicas monitoradas, sem nenhuma máquina virtual em execução além do Domínio-0. Através desta figura, é possível notar que o uso do processador pelo domínio de controle é

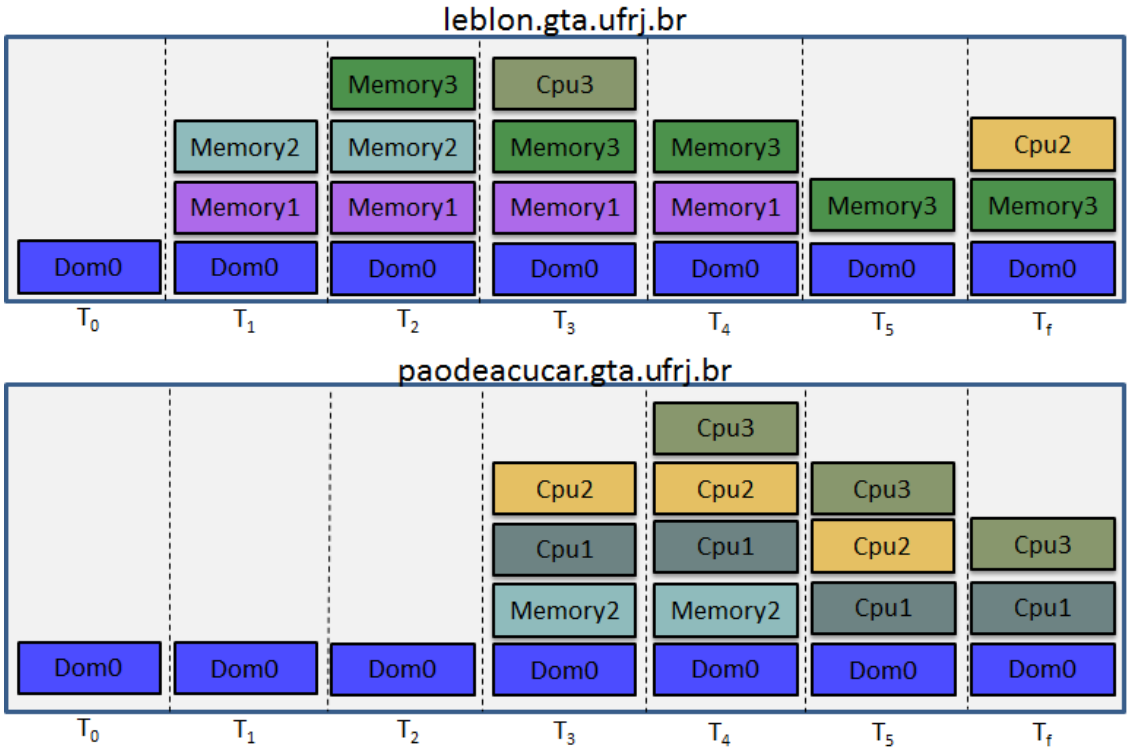


Figura 5.1: Configurações das máquinas monitoradas em cada uma das etapas do cenário de testes.

muito pequeno e a sua memória alocada está na ordem de 12% da quantidade de memória total. As linhas vermelhas representam os limiares de segurança utilizados.

O primeiro item do cenário de testes visa criar um conjunto inicial de máquinas virtuais na máquina física *leblon.gta.ufrj.br*. A Figura 5.3(a) mostra a utilização de CPU durante esta etapa, na qual é possível notar o consumo de CPU do Domínio-0 para realizar a criação das duas máquinas virtuais no intervalo entre 20 e 60 segundos. Também é possível observar, na Figura 5.3(b) o aumento da quantidade de memória alocada da máquina física devido a criação das máquinas virtuais.

O item 2 do cenário de testes cria a máquina memory3 na máquina *leblon.ufrj.br* e está ilustrado na Figura 5.4. É possível notar na Figura 5.4(b) que a quantidade de memória alocada atinge 100% na amostra 83, obrigando o Domínio-0 a realizar o balão de memória e ceder uma parte da sua memória ao domínio recém criado. Como o limiar de segurança de memória foi ultrapassado, o Orquestrador de migração foi acionado (Figura 5.5). Conforme o esperado, a máquina escolhida para

a migração foi a memory2 que possui menor quantidade de memória dentre as 3 máquinas virtuais em execução e, portanto, tem o menor custo de migração. A Figura 5.6 representa as duas máquinas físicas após a migração da máquina memory2. Através da Figura 5.6(b), é possível observar que a migração resolveu a sobrecarga de memória da máquina *leblon.gta.ufrj.br* e alocou corretamente o domínio memory2 na máquina *paodeacucar.gta.ufrj.br* (Figura 5.6(d)). Também é possível observar nas Figuras 5.6(a) e 5.6(c) o consumo de processamento do Domínio-0 das duas máquinas físicas para realizar a migração.

No terceiro item do cenário de testes são criadas as máquinas virtuais cpu1, cpu2 e cpu3. Essa etapa é ilustrada na Figura 5.7. Conforme pode ser observado na Figura 5.7(b), a criação da máquina cpu3 gera uma nova sobrecarga de memória em *leblon.gta.ufrj.br*. De acordo com o algoritmo de decisão, a máquina cpu3 é selecionada para a migração, pois apresenta a menor quantidade de memória alocada. O resultado da migração pode ser observado na Figura 5.8.

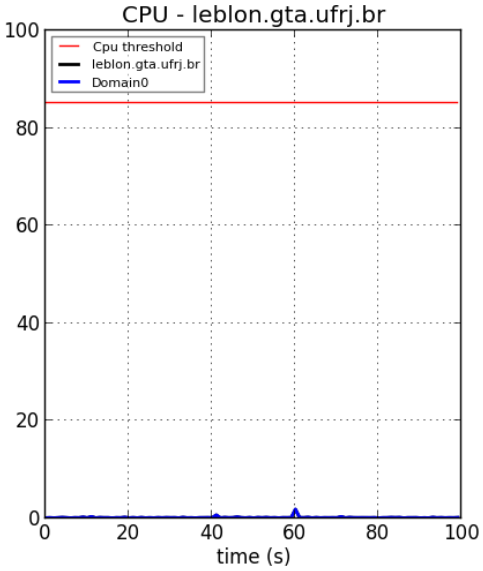
Comparando a migração da máquina cpu3 (Figura 5.8(a)) e da máquina memory2 (Figura 5.6(a)), verifica-se que a duração da migração da cpu3 é menor que a duração da migração da máquina memory2. Isso se deve à diferença da quantidade de memória alocada as duas máquinas: a cpu3 possui 512 MB de memória, enquanto a memory2 possui 2048 MB. Esta constatação valida o critério de decisão adotado neste projeto que opta por migrar máquinas com menor memória alocada, reduzindo o tempo de migração e o uso de CPU do Domínio-0.

Observando o uso do processador pelo Domínio-0 da máquina *leblon.gta.ufrj.br* na Figura 5.8(a) é possível verificar que a migração causou um consumo de CPU durante um intervalo de aproximadamente 60 segundos nessa máquina, enquanto na máquina *paodeacucar.gta.ufrj.br* (Figura 5.8(c)) a utilização de CPU pelo Domínio-0 foi de aproximadamente 20 segundos. É possível, então, verificar que, para realizar a migração de uma máquina virtual, o Domínio-0 da máquina de origem utiliza o processador durante um intervalo de tempo mais longo que o Domínio-0 da máquina física de destino, ou seja, o *overhead* causado pela migração é maior na origem do que no destino.

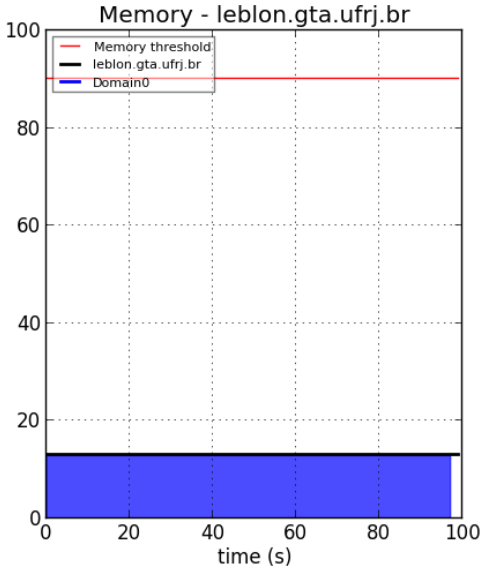
Na etapa 4, as máquinas memory1 e memory2 das máquinas *leblon.gta.ufrj.br* e *paodeacucar.gta.ufrj.br* são desligadas (Figura 5.9), permitindo verificar o comportamento do Domínio-0 durante o desligamento de domínios virtuais. Nota-se que esse processo não causa um grande consumo de processador, apenas um pico de curta duração, como pode ser observado no instante 50s da Figura 5.9(a) e no instante 60s da Figura 5.9(c).

No item 5, é simulada uma carga de trabalho, através do programa Stress, nas máquinas cpu1, cpu2 e cpu3, como mostra a Figura 5.10(a). Nesta mesma figura, é possível observar que a utilização do processador da máquina *paodeacucar.gta.ufrj.br* ultrapassa o limiar de segurança, acionando o algoritmo de migração. De acordo com o algoritmo de decisão, a máquina selecionada para a migração é a cpu2, pois é a que apresenta a maior relação $\frac{processamento}{\text{memória}}$. O resultado da migração pode ser visto na figura 5.11, onde é possível verificar a diminuição da memória alocada da máquina *paodeacucar.gta.ufrj.br* e um aumento na quantidade de memória alocada na máquina *leblon.gta.ufrj.br*.

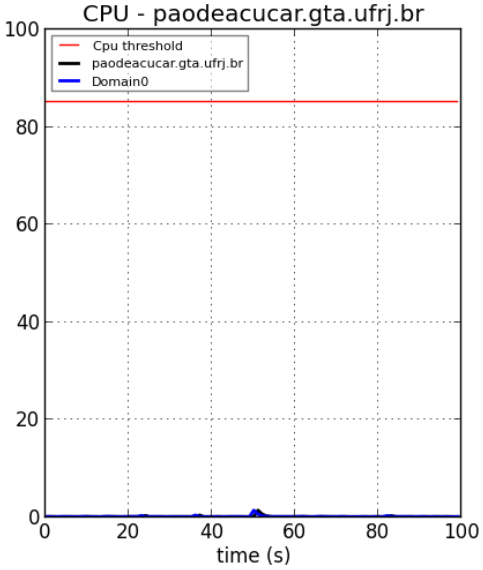
Os resultados obtidos nos testes demonstram que a ferramenta proposta é capaz de realizar o monitoramento de CPU, memória e rede, detectar a escassez de recursos e distribuir as cargas de trabalho entre as máquinas físicas monitoradas. Também foi possível observar o comportamento do Domínio-0 em diferentes situações, como criação, migração e desligamento de máquinas virtuais. O sistema desenvolvido cumpre, então, todos os objetivos descritos no Capítulo 4.



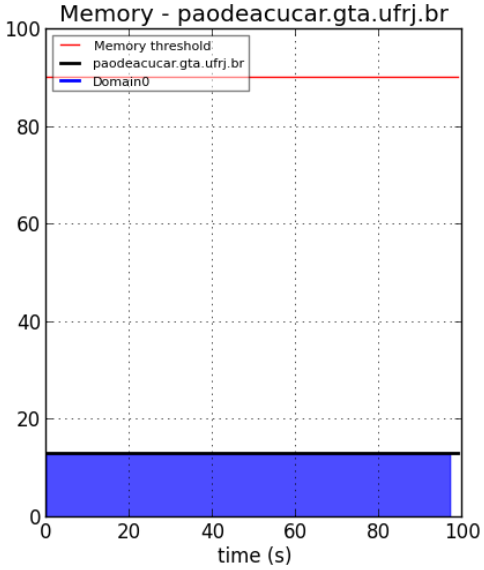
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

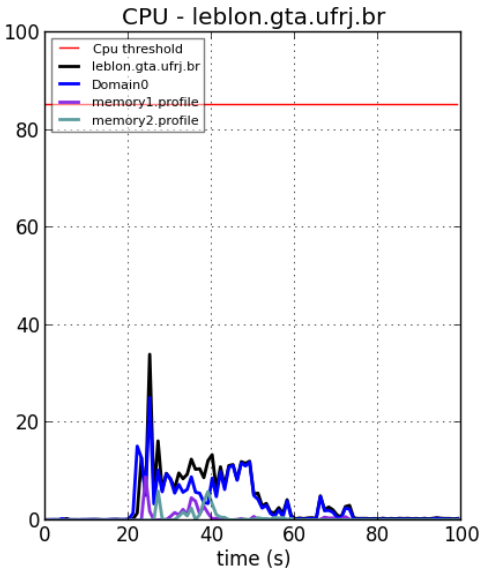


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

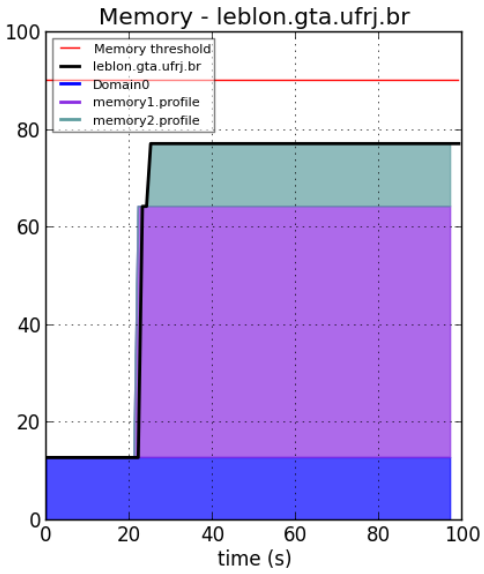


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

Figura 5.2: Estado inicial das máquinas monitoradas.

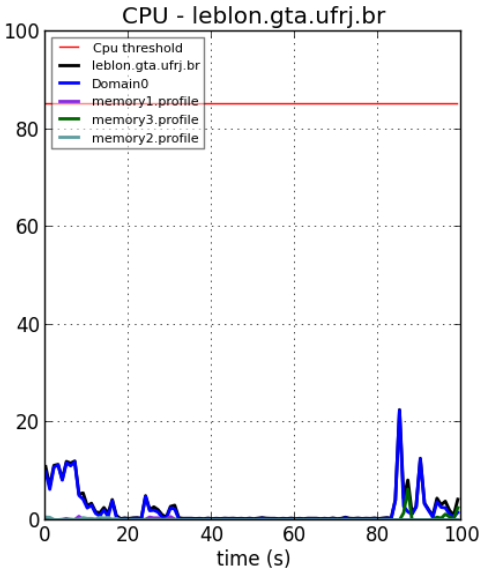


(a) Uso de processador da máquina leblon.gta.ufrj.br.

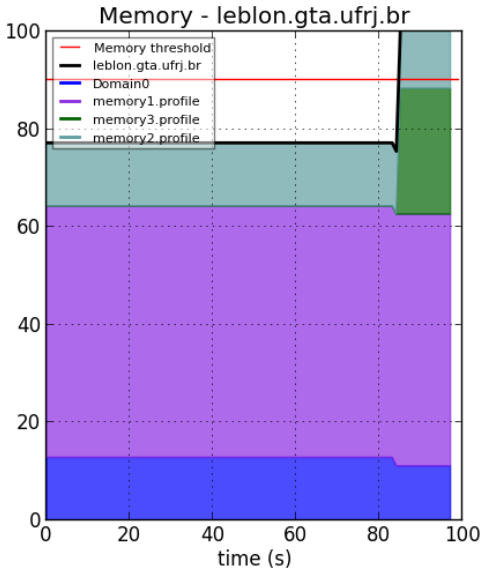


(b) Memória alocada da máquina leblon.gta.ufrj.br.

Figura 5.3: Configuração da máquina leblon.gta.ufrj.br após a criação das máquinas virtuais memory1 e memory2.

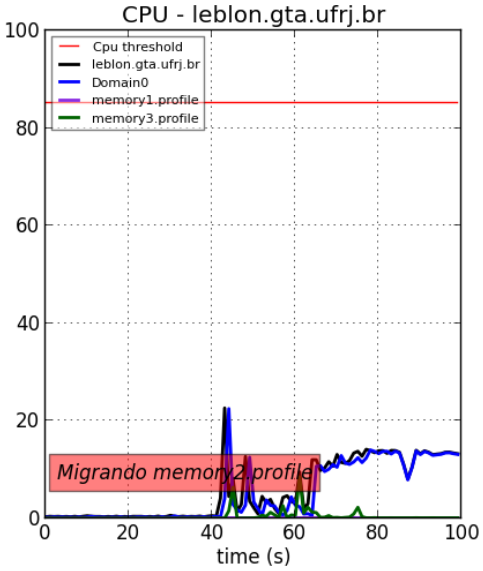


(a) Uso de processador da máquina leblon.gta.ufrj.br.

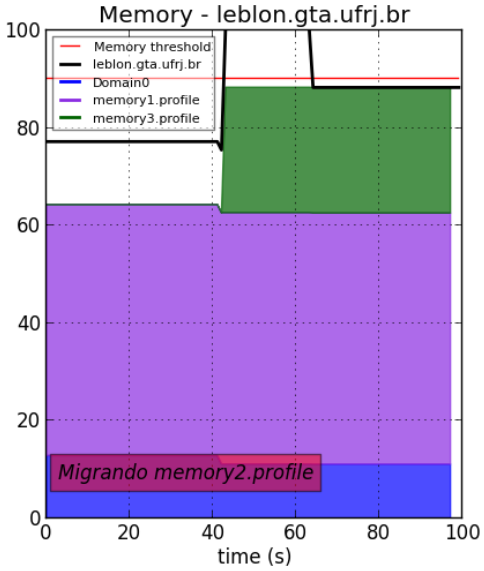


(b) Memória alocada da máquina leblon.gta.ufrj.br.

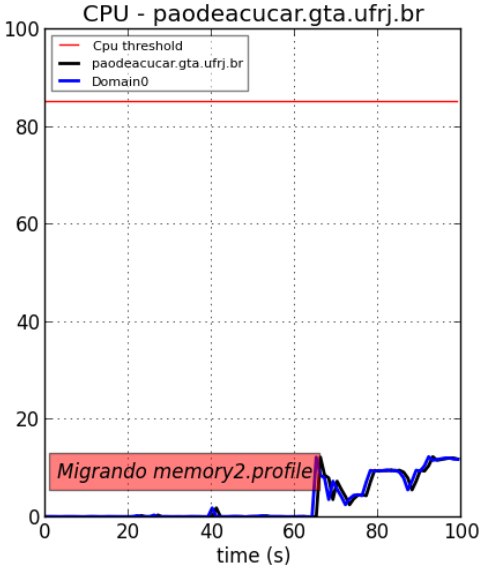
Figura 5.4: Configuração da máquina leblon.gta.ufrj.br após a criação da máquina virtual memory3.



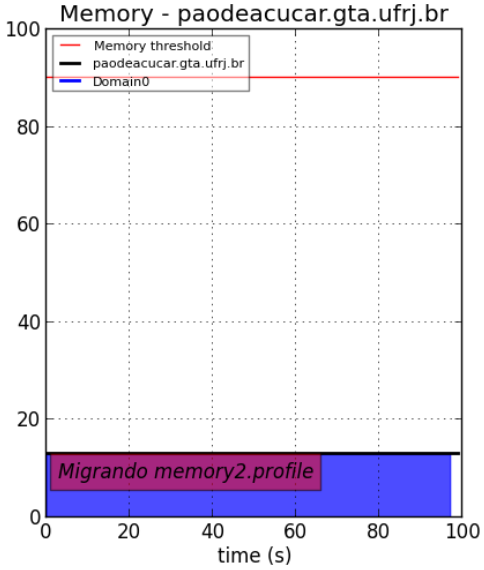
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

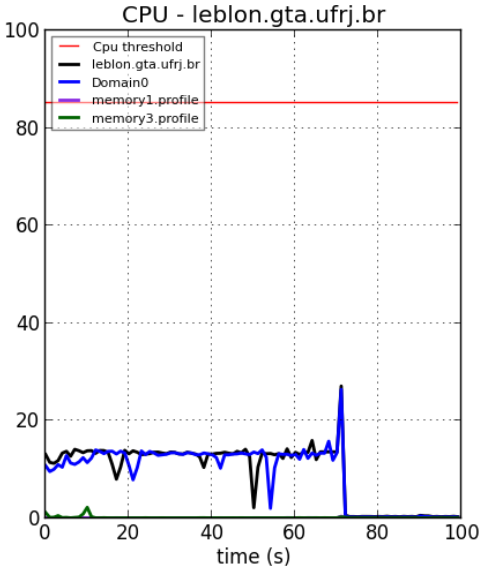


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

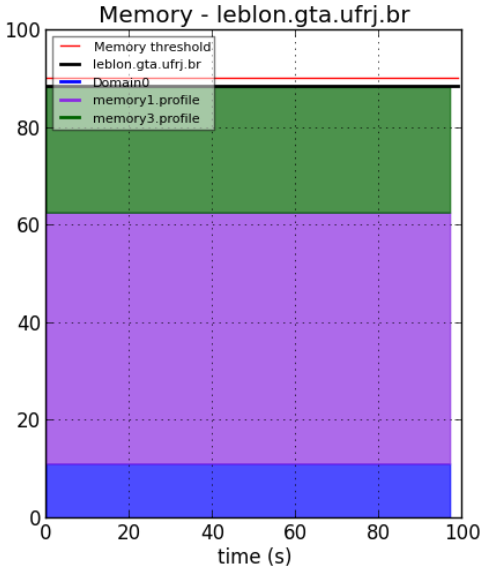


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

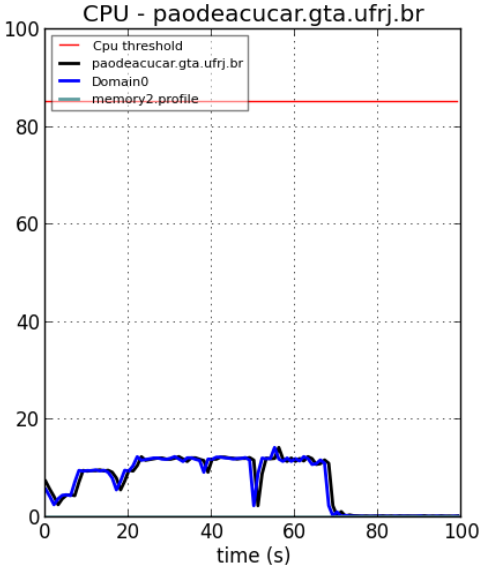
Figura 5.5: Migração da máquina virtual memory2.



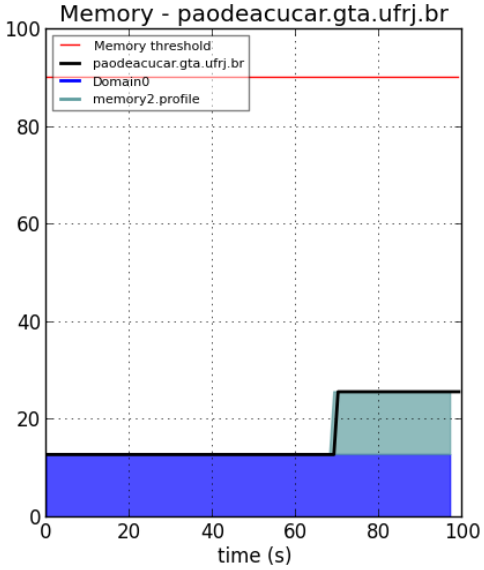
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

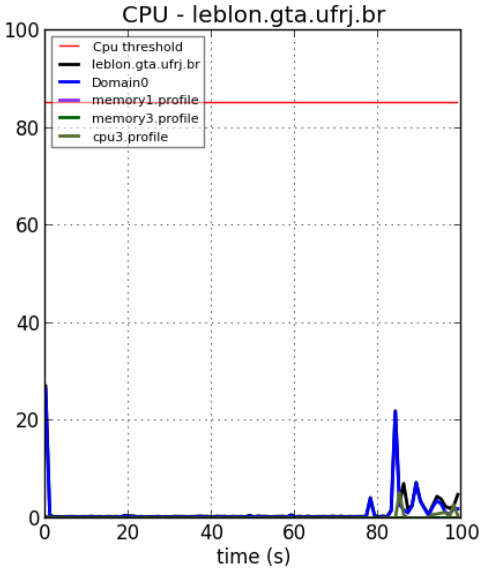


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

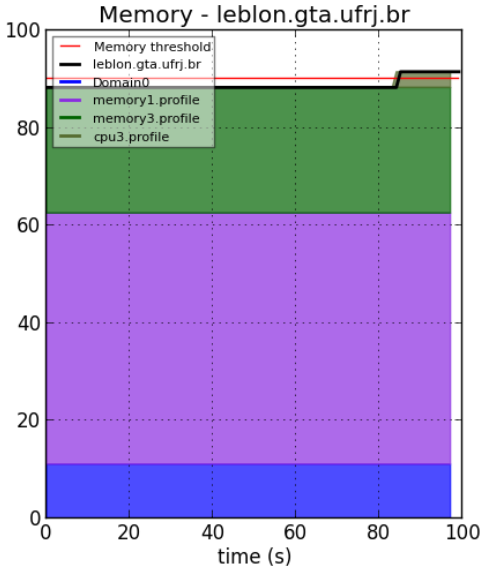


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

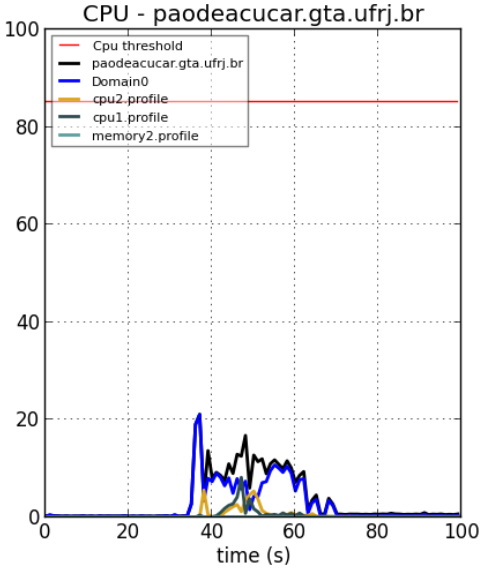
Figura 5.6: Configuração das máquinas após a migração da máquina memory2.



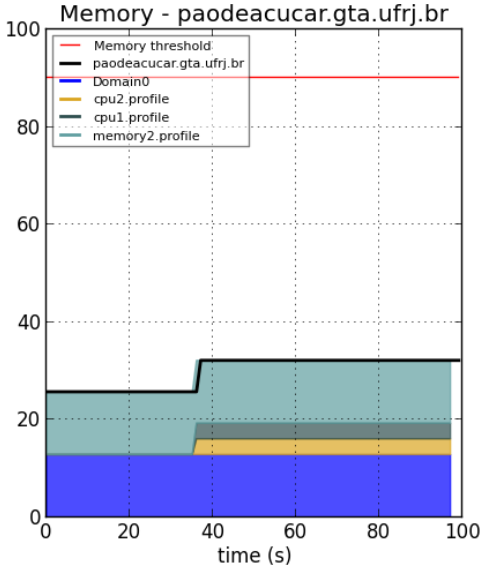
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

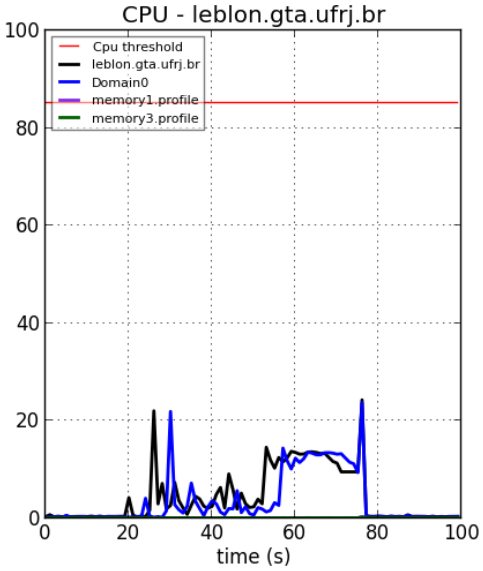


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

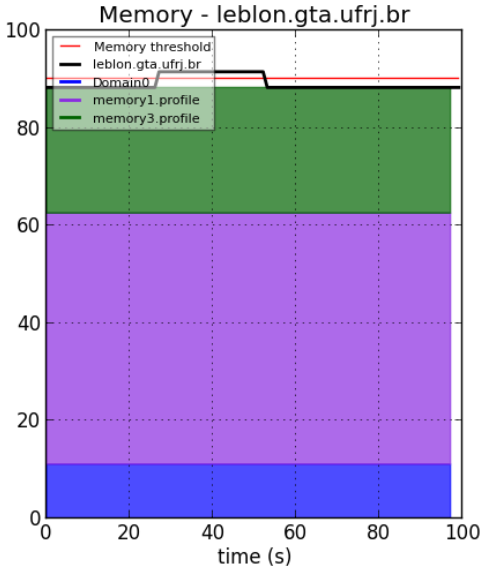


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

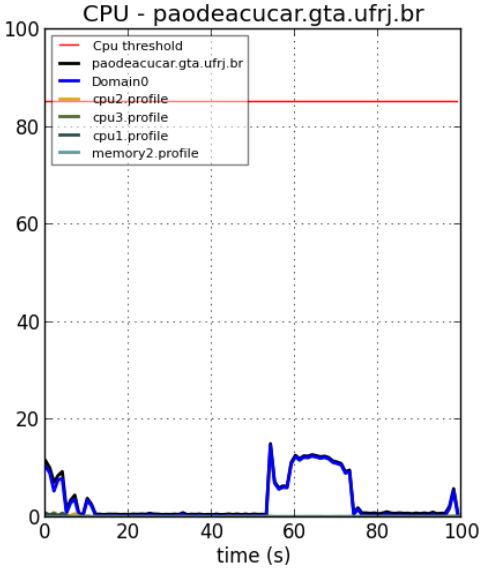
Figura 5.7: Máquinas após a criação das máquinas cpu1, cpu2 e cpu3.



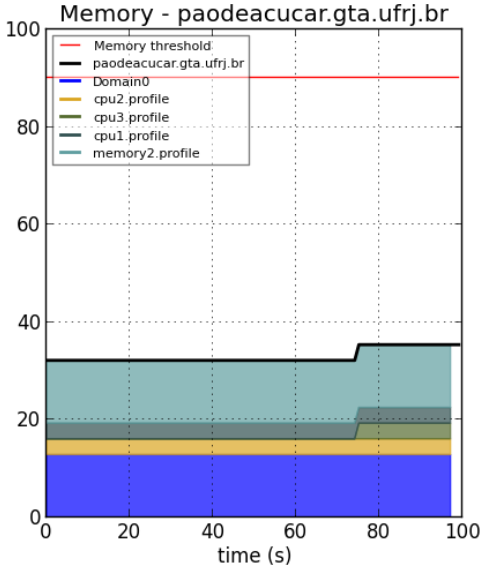
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

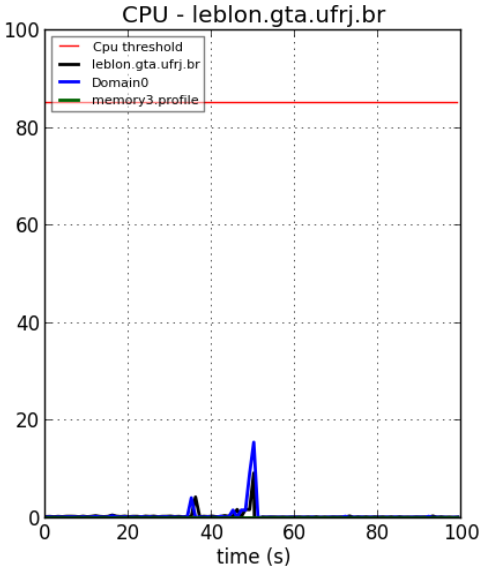


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

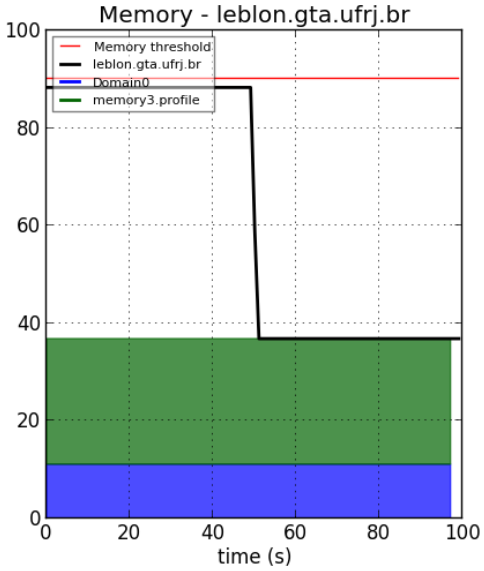


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

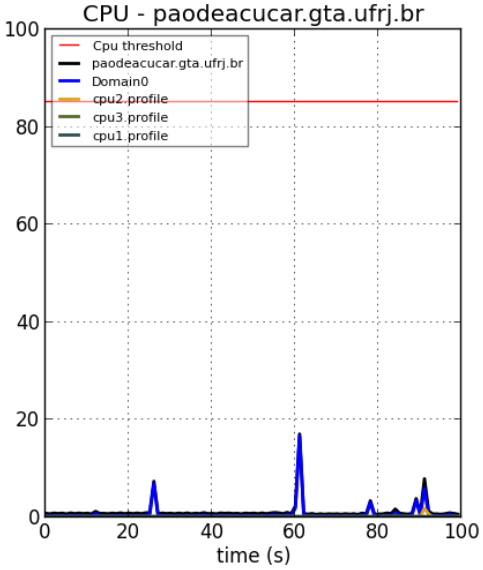
Figura 5.8: Máquinas após a migração da máquina cpu3.



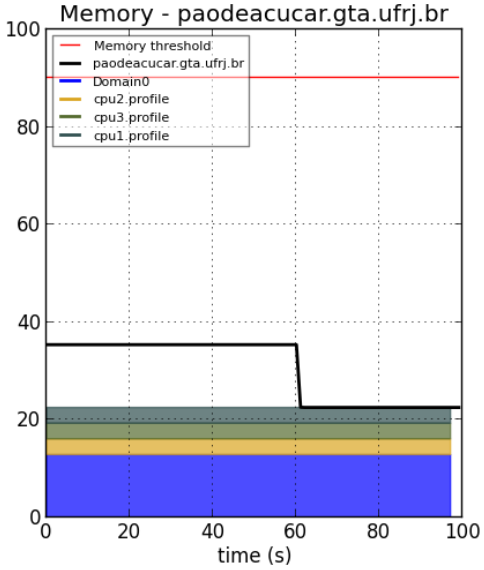
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

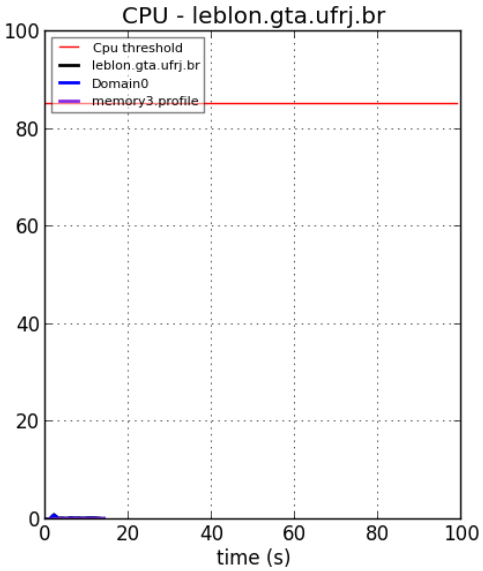


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

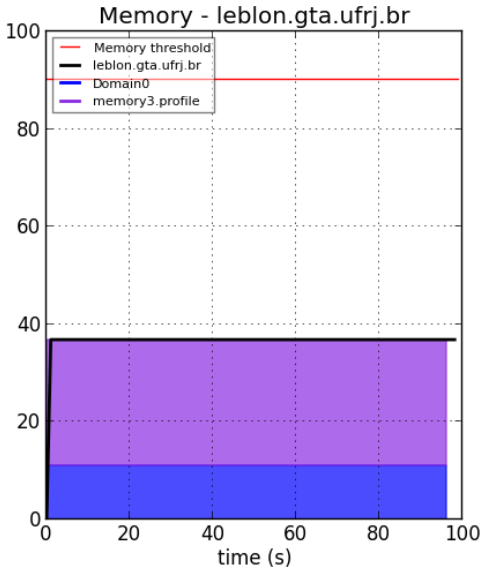


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

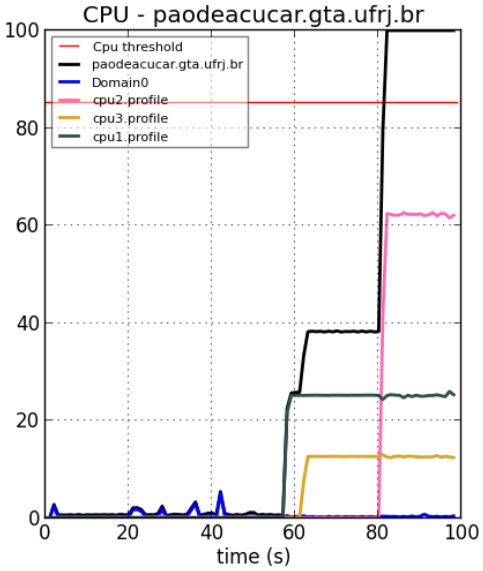
Figura 5.9: Máquinas após a destruição das máquinas memory1 e memory2.



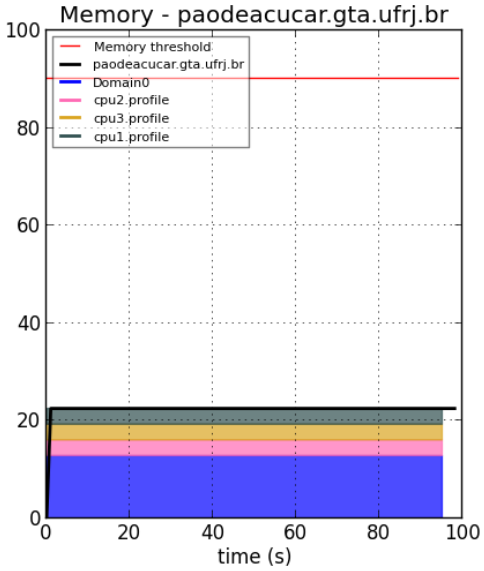
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.

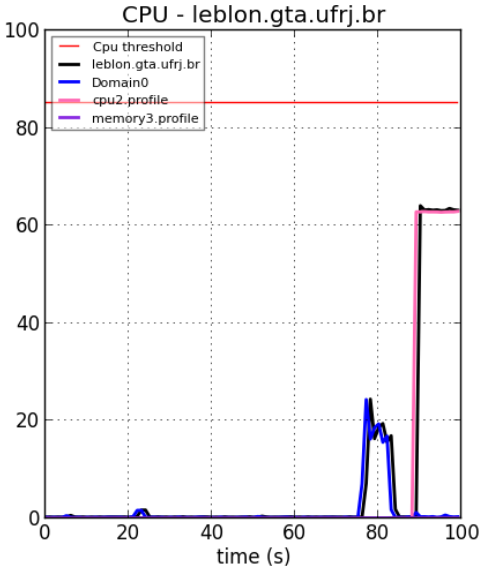


(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

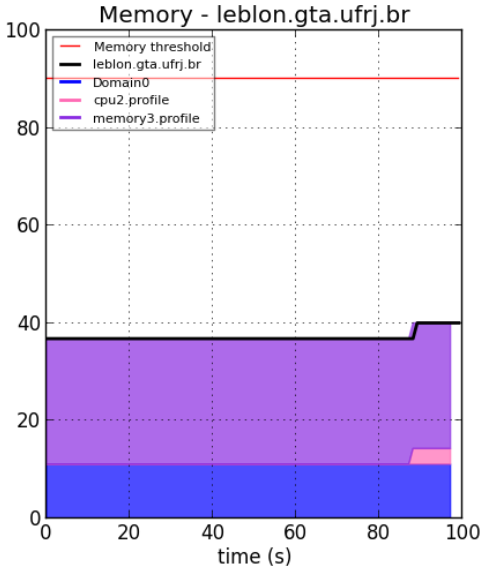


(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

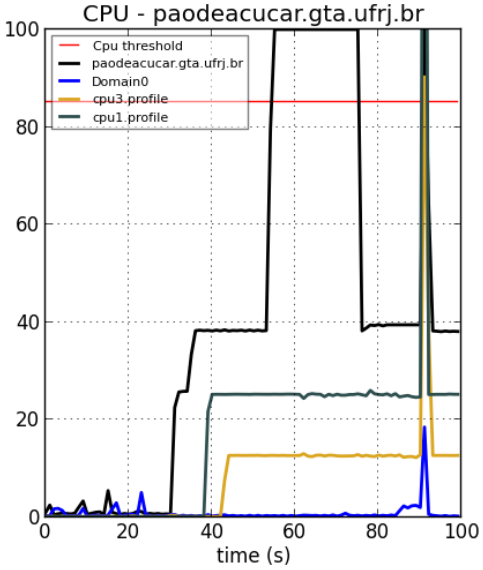
Figura 5.10: Carga de trabalho nas máquinas cpu1, cpu2 e cpu3 gerada com o programa Stress.



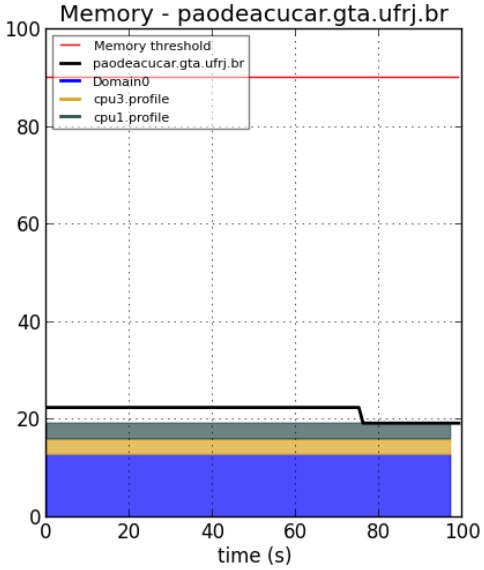
(a) Uso do processador da máquina leblon.gta.ufrj.br.



(b) Memória alocada da máquina leblon.gta.ufrj.br.



(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.



(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

Figura 5.11: Estado final das máquinas físicas.

Capítulo 6

Conclusão

A virtualização de computadores é uma tecnologia que se tornou popular nos últimos anos e tem sido largamente utilizada nos novos *data centers*, pois permite a consolidação de servidores e aplicações, além de proporcionar novos modelos de negócio, como serviços de computação em nuvem. A virtualização também tem sido empregada em rede de computadores e é uma das tecnologias de base nas propostas de "Internet do Futuro" que utilizam a abordagem pluralista.

Apesar de trazer muitas vantagens, a virtualização também trás novos desafios, dentre os quais estão a gerência dos recursos físicos e o aprovisionamento adequado destes recursos às máquinas virtuais. Dado que as máquinas virtuais executam diferentes aplicações e possuem perfis diferentes de utilização de recursos que podem variar ao longo do tempo, é necessário que a gerência das máquinas virtuais seja realizada considerando a elasticidade na demanda e assegure o cumprimento dos acordos de níveis de serviço (SLA) das aplicações em execução nas máquinas virtuais.

O trabalho desenvolvido propõe um sistema de gerência automatizado de recursos para ambientes virtualizados, no qual é possível realizar o monitoramento, a detecção de sobrecargas e a distribuição de cargas de trabalho. O sistema é capaz de monitorar a utilização do processador, a quantidade de memória alocada e o tráfego de rede de um conjunto de computadores, detectando escassez desses recursos e distribuindo cargas, de forma a evitar perda de desempenho nas aplicações em execução. O mecanismo de distribuição de cargas foi baseado na técnica de migração ao vivo fornecida pelo Xen, que permite a migração de máquinas virtuais entre nós físicos,

sem a interrupção dos serviços em execução.

O sistema foi testado utilizando máquinas do testbed FITS, permitindo a validação da proposta. Foi elaborado um cenário de teste que permite testar o sistema nos diferentes cenários de sobrecarga e os resultados obtidos através dele mostram que o sistema implementado é capaz de atender os objetivos propostos. Também foi implementado um painel de controle que permite a visualização do uso dos recursos computacionais em tempo real.

6.1 Trabalhos Futuros

A implementação realizada no escopo deste projeto é um protótipo do sistema proposto. Como trabalho futuro, pretende-se integrar os mecanismos de monitoramento e distribuição de carga nas máquinas do FITS e disponibilizá-los como serviço na interface Web da plataforma de teste. Para isso, pretende-se alterar a arquitetura do sistema, de forma que os perfis de uso sejam gerados de modo distribuído em cada uma das máquinas físicas monitoradas. Esta abordagem reduz o trabalho a ser realizado no nó físico de gerência e torna as medidas mais confiáveis e precisas em ambientes com uma grande quantidade de computadores e domínios virtuais, como é o caso do FITS.

A ferramenta de monitoramento desenvolvida poderá auxiliar na validação de propostas para Internet do Futuro testadas no FITS, já que permite a visualização em tempo real do comportamento das máquinas presentes na plataforma de teste. A parte de monitoramento também pode ser utilizada como base para trabalhos futuros ligados a detecção de intrusão baseados em anomalia. O monitoramento das máquinas e a distribuição de cargas serão implementados no FITS como serviços diferentes, pois, apesar de ser interessante monitorar o uso dos recursos de todas as máquinas virtuais, nem todas as máquinas podem ser migradas, já que a localidade fixa da máquina virtual pode ser indispensável em determinados testes de rede de computadores.

Referências Bibliográficas

[1] WEN, Y., ZHAO, J., ZHAO, G., *et al.*, "A Survey of Virtualization Technologies Focusing on Untrusted Code Execution." In: You, I., Barolli, L., Gentile, A., *et al.* (eds.), *IMIS*, pp. 378–383, 2012.

[2] FIGUEIREDO, R. J. O., DINDA, P. A., FORTES, J. A. B., "Guest Editors' Introduction: Resource Virtualization Renaissance.", *IEEE Computer*, v. 38, n. 5, pp. 28–31, 2005.

[3] MEISNER, D., GOLD, B. T., WENISCH, T. F., "PowerNap: eliminating server idle power." In: Soffa, M. L., Irwin, M. J. (eds.), *ASPLOS*, pp. 205–216, 2009.

[4] BOBROFF, N., KOCHUT, A., BEATY, K., "Dynamic Placement of Virtual Machines for Managing SLA Violations." In: *Integrated Network Management*, pp. 119–128, 2007.

[5] MATTOS, D. M. F., MAURICIO, L. H., CARDOSO, L. P., *et al.*, "Uma Rede de Testes Interuniversitária com Técnicas de Virtualizaçao Híbridas". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2012.

[6] WOOD, T., CHERKASOVA, L., OZONAT, K., *et al.*, "Profiling and modeling resource usage of virtualized applications". In: *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pp. 366–387, 2008.

[7] "The road map from virtualization to cloud computing. Acessado em: Maio 2013. Disponível em: http://easystreet.com/wp-content/uploads/2012/12/Gartner_The-Road-Map-From-Virtualization-to-Cloud-Computing-G00210845_English.pdf".

[8] TAN, L., CHAN, E. M., FARIVAR, R., *et al.*, "iKernel: Isolating Buggy and Malicious Device Drivers Using Hardware Virtualization Support". In: *Proceedings of the Third IEEE International Symposium on Dependable, Autonomic and Secure Computing*, DASC '07, pp. 134–144, 2007.

[9] BARHAM, P., DRAGOVIC, B., FRASER, K., *et al.*, "Xen and the art of virtualization". In: *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pp. 164–177, 2003.

[10] XEN.ORG, "How Does Xen Work? Acesso em: Maio de 2013. Disponível em http://www-archive.xenproject.org/files/Marketing/HowDoesXenWork.pdf".

[11] GOVINDAN, S., NATH, A. R., DAS, A., *et al.*, "Xen and co.: communication-aware CPU scheduling for consolidated xen-based hosting platforms". In: *Proceedings of the 3rd international conference on Virtual execution environments*, VEE '07, pp. 126–136, New York, NY, USA, 2007.

[12] CHERKASOVA, L., GUPTA, D., VAHDAT, A., "Comparison of the three CPU schedulers in Xen.", *SIGMETRICS Performance Evaluation Review*, v. 35, n. 2, pp. 42–51, 2007.

[13] "Credit Scheduler. Acesso em: Junho de 2013. Disponível em: http://wiki.xensource.com/xenwiki/CreditScheduler."

[14] "XenMemoryUsage. Acesso em: Julho de 2013. Disponível em: http://docs.vmd.citrix.com/XenServer/4.0.1/installation/apd.html".

[15] J.H. SCHOPP, K. F., SILBERMANN, M., "Resizing Memory with Balloons and Hotplug", *In Proceedings of the Linux Symposium*, , 2006.

[16] XEN.ORG, "Xen Best Practices. Acesso em: Maio de 2013. Disponível em: `http://wiki.xen.org/wiki/Xen_Best_Practices`", Janeiro 2013.

[17] EGI, N., GREENHALGH, A., HANDLEY, M., *et al.*, "Evaluating Xen for Router Virtualization." In: *ICCCN*, pp. 1256–1261, 2007.

[18] MATTOS, D. M. F., FERRAZ, L. H. G., COSTA, L. H. M. K., *et al.*, "Evaluating virtual router performance for a pluralist future internet". In: *Proceedings*

of the 3rd International Conference on Information and Communication Systems, ICICS '12, pp. 4:1–4:7, 2012.

[19] LEE, M., KRISHNAKUMAR, A. S., KRISHNAN, P., *et al.*, "XenTune: Detecting Xen Scheduling Bottlenecks for Media Applications." In: *GLOBECOM*, pp. 1–6, 2010.

[20] CLARK, C., FRASER, K., HAND, S., *et al.*, "Live migration of virtual machines". In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pp. 273–286, 2005.

[21] MOREIRA, M. D. D., FERNANDES, N. C., COSTA, L. H. M. K., *et al.*, "Internet do Futuro: Um Novo Horizonte". In: *"Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*, 2009.

[22] JACOBSON, V., SMETTERS, D. K., THORNTON, J. D., *et al.*, "Networking named content". In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, 2009.

[23] FEAMSTER, N., GAO, L., REXFORD, J., "How to lease the internet in your spare time", *SIGCOMM Comput. Commun. Rev.*, v. 37, n. 1, pp. 61–64, Jan. 2007.

[24] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., *et al.*, "OpenFlow: enabling innovation in campus networks", *SIGCOMM Comput. Commun. Rev.*, v. 38, n. 2, pp. 69–74, Mar. 2008.

[25] "libvirt - The virtualization API. Acesso em: Março 2013. Disponível em: http://libvirt.org/index.html."

[26] GMACH, D., ROLIA, J., CHERKASOVA, L., *et al.*, "Capacity Management and Demand Prediction for Next Generation Data Centers." In: *ICWS*, pp. 43–50, 2007.

[27] WOOD, T., SHENOY, P. J., VENKATARAMANI, A., *et al.*, "Sandpiper: Black-box and gray-box resource management for virtual machines.", *Computer Networks*, v. 53, n. 17, pp. 2923–2938, 2009.

[28] CARVALHO, H. E. T., DUARTE, O. C. M. B., "VOLTAIC: volume optimization layer to assign cloud resources". In: *Proceedings of the 3rd International Conference on Information and Communication Systems*, ICICS '12, pp. 3:1–3:7, 2012.

[29] KHANNA, G., BEATY, K., KAR, G., *et al.*, "Application Performance Management in Virtualized Server Environments". In: *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pp. 373–381, 2006.

[30] ARZUAGA, E., KAELI, D. R., "Quantifying load imbalance on virtualized enterprise servers". In: *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, WOSP/SIPEW '10, pp. 235–242, 2010.

[31] SONNEK, J., GREENSKY, J., REUTIMAN, R., *et al.*, "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration". In: *Proceedings of the 2010 39th International Conference on Parallel Processing*, ICPP '10, pp. 228–237, 2010.

[32] WOOD, T., TARASUK-LEVIN, G., SHENOY, P., *et al.*, "Memory buddies: exploiting page sharing for smart colocation in virtualized data centers". In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pp. 31–40, 2009.

[33] SINGH, A., KORUPOLU, M., MOHAPATRA, D., "Server-storage virtualization: integration and load balancing in data centers". In: *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pp. 53:1–53:12, 2008.

[34] CHERKASOVA, L., GARDNER, R., "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor". In: *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pp. 24–24, 2005.

[35] WU, Y., ZHAO, M., "Performance Modeling of Virtual Machine Live Migration", *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 492–499, 2011.

[36] “Stress. Acessado em: Maio 2013. Disponível em: http://weather.ou.edu/ apw/projects/stress/”.