Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

REALOCAÇÃO DE RECURSOS EM AMBIENTES VIRTUALIZADOS

Elder de Macedo Rodrigues

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Prof. Dr. Avelino Francisco Zorzo

Porto Alegre
2009

Dados Internacionais de Catalogação na Publicação (CIP)

R696r Rodrigues, Elder de Macedo
Realocação de recursos em ambientes virtualizados / Elder de Macedo Rodrigues. – Porto Alegre, 2009.
80 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Avelino Francisco Zorzo.

1. Sistemas Operacionais (Computação). 2. Redes de Computadores – Gerência. 3. Máquinas Virtuais. I. Zorzo, Avelino Francisco. II. Título.

CDD 005.43

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada **"Realocação de Recursos em Ambientes Virtualizados"**, apresentada por Elder de Macedo Rodrigues, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 08/01/09 pela Comissão Examinadora:

Prof. Dr. Avelino Francisco Zorzo – Orientador — PPGCC/PUCRS

Prof. Dr. César Augusto Fonticielha De Rose - — PPGCC/PUCRS

Profa. Dra. Taisy Silva Weber – — UFRGS

Homologada em 05/05/2009, conforme Ata No. 007 pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS

Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

Para minha família.

Agradecimentos

À Deus.
À Minha família.
Ao Professor Avelino Zorzo.
Aos Colegas, em especial aos do Projeto de Cooperação HP/PUCRS.
À HP (Hewlett&Packard) e a PUCRS pela concessão da bolsa de estudo.

Resumo

A constante evolução das tecnologias computacionais tem possibilitado um aumento no poder de processamento dos computadores, sendo que em determinadas situações esse poder computacional não é plenamente utilizado. Para utilizar efetivamente a capacidade de processamento dos computadores, cada vez mais as empresas de processamento de dados (*data center*) se utilizam da virtualização. A virtualização é uma técnica que permite a execução de diversos Sistemas Operacionais em um único equipamento. Desta forma, com o uso da virtualização um único servidor físico pode manter diversos Sistemas Operacionais, com diferentes aplicações sendo executadas simultaneamente. Quando essas aplicações são portadas para um ambiente virtualizado, devem ter seus níveis de recursos constantemente monitorados e ajustados para se evitar situações de degradação da qualidade do serviço ou prover uma melhor utilização do *hardware*. Nestas situações é importante implementar um *Service Level Agreement* (SLA) que controle a qualidade do serviço entregue por estas aplicações. Entretanto, os Monitores de Máquinas Virtuais, como por exemplo, o Xen, não possuem a funcionalidade de se realocar recursos com base nas regras definidas nos SLAs. Com o objetivo de superar esta limitação, este trabalho propõe a realocação dinâmica de recursos em ambientes virtualizados a partir de requisitos derivados de SLAs. A realocação de recursos, busca também atender a demanda por recursos, por exemplo, em uma máquina virtual (VM) que tenham recebido uma carga de trabalho maior do que a prevista no SLA, através da realocação de recursos (processador e memória) que não estejam sendo utilizados pelas demais VMs.
Palavras-chave: Máquinas virtuais, Xen, SLA, Virtualização, Realocação de Recursos.

Abstract

The constant evolution of computing technologies has allowed an increase in computers processing power, and in certain situations, such computational power is not fully used. In order to effectively use the processing capacity of computers, companies are increasingly using virtualization. Virtualization is a technique that allows running multiple operating systems on a single device. Thus, the use of virtualization on a single physical server can keep multiple operating systems with different applications running simultaneously. When these applications are ported into a virtualized environment, their resource levels should be constantly monitored and adjusted to avoid situations in which the quality of service is deteriorated or to provide a better hardware use. In these situations it is important to implement a Service Level Agreement (SLA) that monitors the quality of service delivered by these applications. Meanwhile, Monitors of Virtual Machines, such as Xen, do not have the functionality to reallocate resources based on the rules defined by SLAs. Aiming to overcome this limitation, this work proposes to dynamically reallocate resources in virtualized environments through the use of requirements derived from SLAs. The reallocation of resources seeks to satisfy the demand for resources of a virtual machine (VM), for instance, having more users than expected in the SLA, by reallocating resources (processor and memory) that are not being used by the other VMs.
Keywords: Virtual Machines, Xen, SLA, Virtualization, Resources Reallocation.

Lista de Figuras

Figura 1 Abstração de um computador moderno 24
Figura 2 Exemplo de abstração de um ambiente virtualizado 25
Figura 3 Estrutura do Jail 29
Figura 4 Estrutura da Emulação 30
Figura 5 Estrutura da Virtualização 31
Figura 6 Estrutura da Paravirtualização 31
Figura 7 Estrutura do Xen 33
Figura 8 Nível de privilégio do Xen [1] 34
Figura 9 Hierarquia da API do Xen 35
Figura 10 Arquitetura XenMon 37
Figura 11 Estrutura de um SLA 44
Figura 12 Definição SLA com base QoS 46
Figura 13 Estrutura de um SLA no VMware 48
Figura 14 Migração de VM com Virtual Server 49
Figura 15 Alocação de recursos no Xen 54
Figura 16 Realocação de recursos no Xen 54
Figura 17 Processo proposto para a realocação de recursos 55
Figura 18 Diagrama de casos de uso da aplicação *web* do TPC-W [2] 57
Figura 19 Diagrama de atividades da aplicação *web* do TPC-W [2] 57
Figura 20 Processo proposto para a realocação de recursos 58
Figura 21 Alocação dos recursos no Xen com Credit 59
Figura 22 Realocação de recursos com o subsistema 60
Figura 23 Algoritmo do subsistema de realocação 61
Figura 24 Algoritmo do subsistema de realocação - Módulo Monitora 61
Figura 25 Algoritmo do subsistema de realocação - Módulo Desaloca 62
Figura 26 Estrutura utilizada 65
Figura 27 Sobrecarga em 1 VM 67
Figura 28 Sobrecarga em 2 VMs 68
Figura 29 Sobrecarga em 3 VMs 69
Figura 30 Comparação do desempenho do subsistema com os escalonadores do Xen 71

Lista de Siglas

SOs	Sistemas Operacionais	19
VM	*Virtual Machine*	19
SLA	*Service Level Agreement*	19
TI	Tecnologia da informação	23
MMV	Monitor de Máquinas Virtuais	24
ISA	*Instruction Set Architecture*	25
HAL	*Hardware Abstraction Layer*	25
SCSI	*Small Computer System Interface*	27
JVM	*Java Virtual Machine*	30
API	*Application Programming Interface*	32
CPU	*Central Processing Unit*	33
XML	*eXtensible Markup Language*	35
RPC	*Remote Procedure Call*	35
I/O	*Input/Output*	36
BVT	*Borrowed Virtual Time*	38
SEDF	*Simple Earliest Deadline First*	38
SMP	*Symmetric Multi-Processor*	39
SLO	*Service Level Objectives*	43
SLI	*Service Level Indicators*	43
ISP	*Internet Service Provider*	44
QoS	*Quality of Service*	45
MTBF	*Mean Time Between Failures*	46
MTTR	*Mean Time To Recover*	46
UML	*Unified Modeling Language*	56
TPC-W	*Transactions Performance Council-Web*	63
WIPS	*Web Interactions Per Second*	64
SUT	*System Under Test*	64

Sumário

1 Introdução 19

2 Virtualização 23
2.1 Arquitetura do Conjunto de Instruções 25
2.2 Camada de Abstração do Hardware 27
2.3 Camada de Virtualização a Nível do Sistema Operacional 28
2.4 Virtualização Através de Linguagens de Alto Nível 29
2.5 Técnicas de Virtualização 30
2.6 Considerações Finais 32

3 Xen 33
3.1 Estrutura do Xen 33
3.1.1 Xen API 34
3.1.2 Ferramentas de Monitoração e Gerência no Xen 36
3.2 Escalonadores do Xen 38
3.3 Alocação de Recursos no Xen 40
3.4 Considerações Finais 41

4 *Service Level Agreement- SLA* 43
4.1 Aplicações de SLA 44
4.1.1 *Service Level Agreement* aplicado em servidores ISP 44
4.1.2 SLA aplicado em redes de computadores 45
4.1.3 SLA aplicado em *Cluster* 46
4.1.4 SLA aplicado em *Grids* de computadores 47
4.1.5 SLA orientado à segurança 47
4.1.6 SLA em máquinas virtuais 48
4.2 Considerações Finais 50

5 Realocação de Recursos: Proposta e Implementação do Subsistema de Realocação 53
5.1 Ambiente Virtual Xen com Credit Estático 53
5.2 Processo Proposto 55
5.2.1 Decomposição de SLA 55
5.2.2 Realocação de Recursos 59
5.3 Considerações Finais 62

6 Resultados 63
6.1 Ambiente de Teste 63
6.2 Casos de Teste e Resultados 65
6.2.1 Casos de teste e resultados com o TPC-W 66

6.2.2 Casos de testes e resultados com o Jmeter 69
6.3 Considerações Finais 72

7 Conclusão 73

Referências 75

1 Introdução

A constante evolução das tecnologias computacionais tem possibilitado um aumento no poder de processamento dos computadores. No entanto, este poder computacional não é plenamente utilizado, já que as aplicações que são hospedadas sobre estas estruturas se utilizam em média de apenas 8%-10% dos recursos disponíveis, sendo que o restante não é utilizado [3].

Motivado por este, entre outros fatores, empresas de *data centers* têm, cada vez mais, se utilizado da virtualização de servidores para um melhor aproveitamento do *hardware*. A virtualização é um processo para executar diversos Sistemas Operacionais (SOs) em um único equipamento. Cada um desses SOs é uma máquina virtual (*Virtual Machine- VM*), sendo que cada VM é um ambiente operacional completo que se comporta como se fosse um computador independente [4]. Deste modo, com o uso da virtualização um único servidor físico pode manter diversos Sistemas Operacionais, com diferentes aplicações sendo executadas simultâneamente.

No entanto, quando essas aplicações são portadas para um ambiente virtualizado, devem ter seus níveis de recursos constantemente monitorados e ajustados para se evitar situações de degradação da qualidade do serviço ou prover uma melhor utilização do hardware. Nestas situações é importante implementar um Acordo de Nível de Serviço (*SLA-Service Level Agreement*) que controle a qualidade do serviço entregue por estas aplicações e disponha de maneira clara as responsabilidades de cada participante no SLA [5].

SLA é uma declaração de expectativas e obrigações que existem no relacionamento de negócio entre duas organizações: o provedor do serviço e seu consumidor [6]. Essa declaração é chamada de contrato, pois especifica os níveis de qualidade de serviços que o fornecedor se compromete em disponibilizar, além das cláusulas legais, bem como as conseqüências para cada parte se houver descumprimento destes deveres. Ou seja, o SLA estabelece o nível de serviço requisitado e é projetado para criar uma compreensão comum sobre serviços, prioridades e responsabilidades, entre clientes e provedores de serviços.

Historicamente o SLA foi desenvolvido pelas operadoras de telefonia fixa que o usavam como parte de seus contratos com seus clientes e o utilizavam como ferramenta que determina e delimita os objetivos de desempenho e define de maneira formal as suas obrigações, mas de forma um pouco genérica e relativamente de pouco uso. No contexto atual os acordos deverão ser explícitos e exigentes, de modo que os provedores de serviços precisarão adotar algum mecanismo de gerência de SLAs, para garantir o cumprimento dos mesmos. Os provedores de serviços estão descobrindo que SLAs flexíveis e bem estruturados, podem auxiliar na instalação, a correção de falhas e a disponibilidade de classes de serviços. Quando claramente definidos e compreendidos, os SLAs serão ferramentas de gerência poderosas para seus usuários. Mais do

que apenas uma garantia, eles estimulam a comunicação em baixo custo, alocação de recursos e prioridades de negócios, incentivando mais planejamento e colaboração [7]. Atualmente as mais diversas áreas lançam mão do uso de SLAs como ferramentas para definir os índices de serviços e a expectativa do cliente com este serviço.

Na área da Ciência da Computação, os SLAs são utilizados em diversas subáreas, sendo a virtualização uma destas. Na virtualização sua utilização se deve principalmente às características destes ambientes, que são altamente dinâmicos, e seus serviços/recursos tendo que ser constantemente ajustados para uma melhor utilização do *hardware* [5]. Caso não estejam sendo monitorados, o desempenho e a utilização dos recursos, em cada uma das máquinas virtuais que compartilham o mesmo *hardware*, pode uma VM estar com 60% dos recursos alocados e estar utilizando apenas 10%, e outra VM possuir 40% dos recursos e estar utilizando 95% destes. Por este motivo, utilizam-se SLAs para definir quais as necessidades de recursos para uma VM, executando um tipo específico de aplicação, e também, para monitorar a utilização desses recursos por cada uma destas VMs com o objetivo de avaliar a distribuição dos recursos, com base nas regras declaradas no SLA.

No entanto, atualmente, os monitores de máquinas virtuais não possuem a funcionalidade de realocar os recursos do sistema, tendo como base as políticas definidas em um SLA. Alguns trabalhos propõem apenas a definição do nível inicial de recursos necessários para uma aplicação sob determinado SLA [8] [9], mas sem realocar recursos físicos (processador e memória) em tempo de execução. Assim sendo, atualmente não é possível alterar de forma dinâmica o nível de recurso atribuído a uma VM, caso o valor de uma métrica contida no SLA seja alterada ou a aplicação necessite de mais recursos dos que aqueles descritos no SLA.

Visando solucionar esta limitação, esta dissertação apresentará uma proposta, implementação e avaliação de um processo para a realocação de recursos em ambientes virtualizados. Esse processo se divide em duas etapas, a primeira consiste em definir de maneira precisa, através da decomposição de SLA, qual o nível de recursos necessários para uma aplicação cumprir um SLA. A segunda também se utiliza dos resultados da decomposição, no entanto, eles são utilizados para realizar a realocação de recursos em tempo de execução. Desta forma, a utilização do subsistema permite uma melhor utilização do *hardware* e possibilita a utilização de SLAs em ambiente virtualizados.

O desenvolvimento desse trabalho é a continuação de pesquisas desenvolvidas durante o trabalho de mestrado de dois integrantes da linha de pesquisa [10] [11]. Estes dois trabalhos buscavam melhorar o desempenho das VMs, através da realocação de recursos no Xen. Para isso, se utilizavam da execução de *benchmarks* para coletar dados de configuração e desempenho das VMs, onde esses eram submetidos a algoritmos de mineração de dados. Esses algoritmos indicavam, a partir de uma determinada configuração das VMs, qual a percentagem de recursos deveria ser realocado para que a mesma apresente uma melhora no desempenho [12].

Esta dissertação está organizada da seguinte forma: o Capítulo 2 apresenta os principais conceitos sobre virtualização e alguns monitores de máquinas virtuais. No Capítulo 3 serão

apresentados os principais conceitos, utilização e alguns exemplos de SLA. No Capítulo 4 será detalhado o processo de decomposição de SLA e a realocação de recursos em ambientes virtualizados. O Capítulo 5 apresenta a metodologia utilizada para validar o processo proposto e alguns resultados. O Capítulo 6 apresenta as conclusões sobre o trabalho e indica alguns possíveis trabalhos futuros.

2 Virtualização

O conceito de máquina virtual (VM - *Virtual Machine*) surgiu no início dos anos 60, desenvolvido pela IBM para permitir uma melhor utilização de recursos em seus *mainframes* [13]. Esse conceito consistia em dividir os recursos físicos providos pelo *hardware* entre diversas máquinas virtuais. Desta forma, cada VM era um sistema independente e se comportava de forma idêntica do que quando sobre a estrutura física, só que com menos recursos que a máquina real na qual estava hospedada.

Redução de custos de TI, execução de soluções proprietárias incompatíveis com a maioria dos sistemas atuais, facilidade de implementação, manutenção e portabilidade. Estas e outras necessidades impulsionaram a utilização da virtualização, deixando de ser uma solução apenas para *mainframes* da IBM, e começando a ser utilizada pela indústria e explorada no ambiente acadêmico. O uso da virtualização permite que as empresas alcancem uma maior utilização destes recursos, o que, conseqüentemente, permitirá uma redução dos custos totais de propriedade, através da consolidação de servidores. A virtualização possibilita que uma máquina virtual apresente ao Sistema Operacional convidado um subconjunto lógico de recursos computacionais, de modo que possam ser alcançados resultados e benefícios, como se o sistema estivesse sobre a configuração nativa.

Além dos benefícios apresentados, o uso da virtualização possibilita também diversas outras vantagens, como, por exemplo:

- Consolidação de servidores: permite transferir para algumas máquinas físicas, diversos Sistemas Operacionais que estavam hospedados cada um em um único servidor físico. Desta forma, o uso da virtualização permite um melhor aproveitamento do *hardware*, economia de energia e uma maior facilidade de gerenciamento e manutenção.

- Consolidação de aplicações: pelo fato de ainda existirem diversas aplicações legadas dentro dos *data centers* das empresas e estas aplicações, em alguns casos, necessitarem de *hardware* e Sistemas Operacionais (SOs) obsoletos, é importante portar estas aplicações para um *hardware* moderno. Desta forma, pode-se fazer uso da virtualização para emular estes *hardware* e SO obsoletos.

- Balanceamento dinâmico de recursos: o fato de que os recursos físicos disponíveis para uma VM podem ser aumentados ou diminuídos, sem a necessidade de se alterar a estrutura física do computador, é especialmente importante para um sistema que precisa de uma grande capacidade de processamento, em situações específicas e temporárias. Outra

possibilidade de utilização do balanceamento de recursos seria para prover a alocação de recursos necessários para uma aplicação manter um SLA (*Service Level Agreement*).

- *Sandboxing*: o uso de máquinas virtuais permite a criação de ambientes isolados para serem utilizados por aplicações inseguras. Desta forma, a virtualização funciona como uma camada a mais entre a aplicação e os seus usuários. *Sandboxing* também são definidos como locais ideais para se utilizar, como ambientes de teste de aplicações, pois caso uma aplicação provoque uma instabilidade no sistema, essa instabilidade afetará apenas a máquina virtual na qual está hospedada, sem comprometer as demais VMs que compartilham a estrutura. Do mesmo modo, a utilização de VMs é um ambiente ideal para o isolamento de falhas, pois caso uma falha, por exemplo, do Sistema Operacional venha a ocorrer em ambiente, a mesma permanece contida a este, sem afetar as demais VMs que compartilham a estrutura.

- Migração de VMs: a possibilidade de migração de VMs, entre diversas máquinas físicas, possibilita uma significativa redução do tempo de *down time* de uma aplicação, no caso de uma falha do *hardware* ou do *software*.

Segundo [14], para prover as funcionalidades de criação, execução e gerenciamento destes ambientes virtuais, uma camada virtualizada deve ser adicionada em algum nível sobre o *hardware*. Um sistema computacional moderno pode ser representado por diversas camadas, sendo o *hardware* o nível mais baixo, tendo acima dele diversas camadas que representam o Sistema Operacional e programas aplicativos, como pode-se observar na Figura 1.

Figura 1 – Abstração de um computador moderno

A camada de virtualização permite que as diversas máquinas virtuais (com diferentes Sistemas Operacionais ou não) que executam sobre ela, se utilizem dos mesmos recursos providos pela máquina física. A gerência, abstração do *hardware* e atribuição dos recursos destas máquinas virtuais é executada por um Monitor de Máquinas Virtuais (MMV), que pode executar sobre um Sistema Operacional, ou executar diretamente sobre o *hardware* (Figura 2). A camada

de virtualização pode estar em diversos níveis de abstração, dentro de um sistema computacional. Em [4] são apresentados alguns desses níveis, por exemplo, a virtualização da Arquitetura do Conjunto de Instruções (*Instruction Set Architecture Level-ISA*), virtualização em nível de *hardware* ou camada de abstração do *hardware* (*Hardware Abstraction Layer-HAL*), virtualização ao nível do Sistema Operacional (*OS-level Virtualization*) e virtualização em nível de aplicação. As próximas seções descrevem como cada um desses níveis funciona.

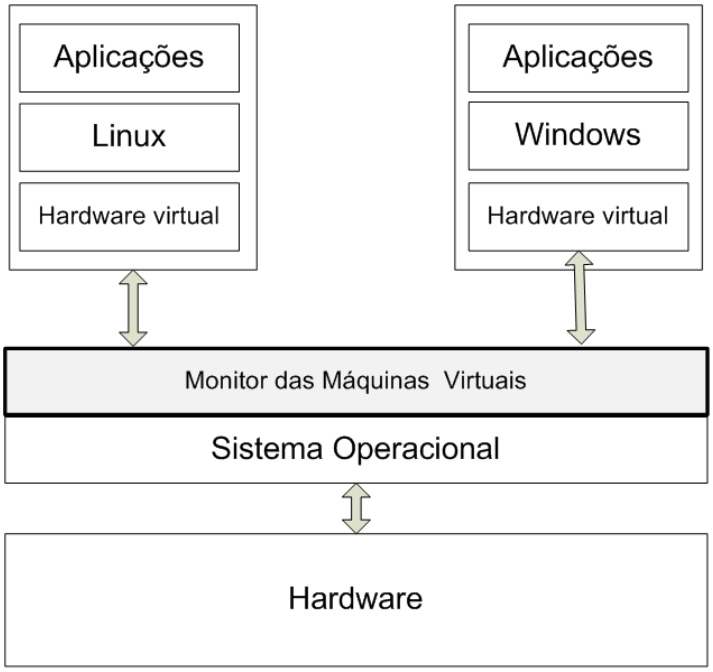

Figura 2 – Exemplo de abstração de um ambiente virtualizado

2.1 Arquitetura do Conjunto de Instruções

A arquitetura do Conjunto de Instruções (*Instruction Set Architecture-ISA*) é a interface básica entre o *hardware* e o *software*, sendo constituída pelas instruções, em código de máquina, aceitas pelo processador e todas as operações de acesso aos recursos do *hardware* (acesso físico à memória, às operações de entrada/saída, ao relógio do sistema, etc.) [15].

A virtualização da Arquitetura do Conjunto de Instruções (ISA) consiste basicamente na emulação, por *software*, de todas as instruções da arquitetura de um computador. Um computador consiste tipicamente de processadores, memórias, discos rígidos, controladores de disco, dispositivos de entrada/saída, etc. Para possibilitar a virtualização, um emulador deve executar todas as instruções emitidas pelas máquinas virtuais, traduzindo-as para o conjunto de instruções do *hardware* nativo, e, em seguida, executando-o com o *hardware* disponível. No entanto,

para um emulador emular com sucesso um computador, ele tem de ser capaz de emular tudo o que um computador real executaria [14]. Assim como todas as demais arquiteturas de máquinas virtuais, a virtualização das ISA possui algumas vantagens e desvantagens. Como vantagens, pode-se citar sua simplicidade, robustez e facilidade de execução em múltiplas plataformas. Este fato em especial se deve ao modo como ela é implementada. Como o seu funcionamento é através da emulação das instruções da máquina virtual em instruções da máquina física, ela é portável para diversas arquiteturas, como por exemplo, x86, Sparc, Alpha, etc. Entretanto, esta portabilidade também é um fator negativo, pois como todas as instruções da máquina virtual precisam ser emuladas, o desempenho sofre uma forte degradação. Como exemplo de *software*, que se utilizam da emulação para executar diversos Sistemas Operacionais sobre uma mesma estrutura física pode-se citar o Bochs [16] e o QEMU [17].

Bochs [18] é um software escrito na linguagem de programação C++, que emula arquiteturas x86 e x86-64, incluindo as extensões MMX, SSEx e 3DNow sobre diversas outras arquiteturas. Como ele emula todo o funcionamento de um computador, o *software* rodando na emulação "acredita"que está sendo executado em uma máquina física. Essa abordagem permite que o Bochs execute uma grande variedade de SO, sem ser necessário qualquer modificação, incluindo, por exemplo, Windows 95/98/NT/2000/XP/Vista e todas as distribuições Linux.

Para ser portável para uma grande variedade de arquiteturas, ele se utiliza da emulação de instruções para simular cada instrução da arquitetura x86, assim sendo, ele pode, por exemplo, executar um Sistema Operacional Windows sobre um servidor Alpha ou Sun. No entanto, por utilizar-se da emulação de instruções, o Bochs deve converter todas as instruções da arquitetura onde está hospedado para instruções x86, o que ocasiona uma forte degradação no desempenho das máquinas emuladas. Um outro *software* que se utiliza da emulação para prover virtualização é o QEMU.

QEMU é um emulador similar ao Bochs, que assim como ele pode ser utilizado sobre diversos tipos de arquiteturas, no entanto ele também é capaz de emular arquiteturas diferentes, como x86, PowerPC e Sparc. O QEMU também pode emular diversas máquinas virtuais com diferentes Sistemas Operacionais, como, Windows, Linux e Mac OS X. Apesar de similar ao Bochs, ele apresenta uma diferença significativa quanto ao desempenho. Isso se deve ao fato do QEMU utilizar-se de um tradutor dinâmico. O tradutor dinâmico realiza, em tempo de execução, a tradução das instruções da arquitetura emulada para as instruções da arquitetura real, sendo o resultado dessa tradução armazenado em uma *cache* para posterior reutilização. Deste modo, uma instrução da arquitetura emulada é traduzida apenas uma vez, durante a execução do QEMU. Apesar de o desempenho apresentando pelo QEMU ser superior ao de outros emuladores, o seu desempenho é inferior quando comparado com *softwares* que se utilizam de outras técnicas de virtualização.

2.2 Camada de Abstração do Hardware

Diversas ferramentas de virtualização se utilizam de uma camada de abstração do *hardware* (*Hardware Abstraction Layer-HAL*), que possibilita simular completamente ou apenas partes do *hardware* para rodar um Sistema Operacional convidado sem ou com modificações sobre um único *hardware* físico. Utilizando da camada de abstração do *hardware* (HAL), os Sistemas Operacionais convidados têm acesso direto aos recursos físicos do computador (processador, memória, etc.), deste modo, evita-se a forte degradação no desempenho provocado pela emulação [19]. Entretanto, quando a máquina virtual necessita acessar recursos físicos críticos, esta requisição não pode ser realizada diretamente ao *hardware*, mas sim, à camada de abstração (*Virtual Machine Monitor-VMM*) [20]. A utilização do VMM é necessária, por exemplo, para impedir que uma máquina virtual, que apresente um defeito durante a execução de uma instrução crítica, provoque a quebra das demais VMs que compartilham o *hardware*. Desta forma o VMM tem o controle das VMs e mantém cada uma isolada das demais. Diversos *softwares* de virtualização, como o VMWare [21], o Virtual PC [22], o Denali [23] e Xen [24], se utilizam desta técnica para proverem máquinas virtuais eficientes quanto ao desempenho, mas pouco portáveis quanto aos tipos de arquitetura emulada.

O VMWare é uma ferramenta de emulação comercial composta de três versões, VMware Workstation, VMware GSX server e VMware ESX server [25]. O VMWare possibilita a emulação completa de diversos computadores virtuais, através da emulação do processador, rede, placa gráfica, memória, som, etc., da máquina física. Cada uma destas máquinas virtuais pode portar um SO, diferentes dos demais ou não, sobre um mesmo *hardware* físico. Um diferencial apresentado pelo VMWare, na versão ESX server, é a possibilidade de instalação do emulador diretamente sobre o *hardware* em uma máquina sem SO. Sendo que, nesse caso, o próprio emulador provê um console, onde são realizadas as tarefas de criação e manutenção das VMs. O VMWare também suporta uma grande variedade de configurações de rede, como a capacidade de emulação de um *hub* virtual que conecta todas as interfaces virtuais de rede das VMs, criando assim uma LAN virtual dentro de um computador físico. No entanto, para a criação de conexões entre as placas de redes virtuais das VMs e a interface de rede física, a comunicação deve ser realizada através de uma *brigde* virtual.

Outra ferramenta de emulação, com funcionalidades e características similares ao VMWare, é o Virtual PC. Ele foi inicialmente desenvolvido pela Connectix Corp. e, posteriormente adquirido para a Microsoft. Em 2003, a Microsoft lançou uma nova versão do Virtual PC, que faz uso da tecnologia Intel Virtualization, com o objetivo de melhorar o desempenho das VMs. Quando comparado com o VMWare, Virtual PC possui algumas desvantagens, como a falta de suporte para a emulação de dispositivos SCSI (*Small Computer System Interface*), uma restrição que não permite alterar o *hardware* emulado após a criação das VMs, e o fato de suportar apenas Sistemas Operacionais Windows como sistema convidado.

Apesar do VMWare e do Microsoft Virtual PC serem ferramentas eficientes e práticas para proverem ambientes virtualizados, eles possuem limitações quanto à criação e gerência de ambientes com grandes números de máquinas virtuais. A forma como a virtualização é implementada pelo VMM destas ferramentas torna difícil, por exemplo, gerenciar a memória de um elevado número de máquinas virtuais ao mesmo tempo. Para suprir esta deficiência e aumentar a escalabilidade e o desempenho, a Universidade de Washington desenvolveu o projeto Denali [23]. No desenvolvimento da nova ferramenta, que compartilha o mesmo nome do projeto, foi implementada uma nova técnica, chamada de paravirtualização [26].

A paravirtualização consiste basicamente em modificar a técnica tradicional de virtualização, onde o VMM intermedia todas as requisições feitas pelas VMs para o *hardware*, para um modelo onde o VMM intermedia apenas as requisições críticas, sendo permitido que as demais requisições sejam feitas diretamente ao *hardware*. Desta forma, a utilização desta técnica permite aos ambientes virtualizados serem altamente escaláveis, e a obterem um desempenho próximo ao obtido caso estivessem diretamente sobre o *hardware*. O ponto negativo, é que o Sistema Operacional base precisa ser modificado para suportar a paravirtualização.

2.3 Camada de Virtualização a Nível do Sistema Operacional

A camada de virtualização a nível do Sistema Operacional situa-se entre o Sistema Operacional e os programas aplicativos. Acima dessa camada está a máquina virtual, que executa aplicações, ou conjuntos de aplicações, desenvolvidas especificamente para um determinado ambiente operacional. Esse ambiente, necessário para uma aplicação, é composto por um Sistema Operacional, de bibliotecas, um sistema de arquivos e outras definições de ambiente. Caso a camada de virtualização, a nível do Sistema Operacional, produza um ambiente operacional idêntico ao ambiente operacional quando executado diretamente sobre o *hardware*, e considere o compartilhamento de recursos, esse ambiente operacional é considerado uma máquina virtual [19]. Basicamente uma máquina virtual, ao nível do Sistema Operacional, pode ser definida como um serviço, onde diversos serviços podem ser executados no sistema ao mesmo tempo, mas de forma independente. Assim, todos os serviços executam na mesma instância do Sistema Operacional, mas cada um deles não pode acessar qualquer arquivo ou ver processos dos demais, provendo, desta forma isolamento entre cada uma das máquinas virtuais. Como exemplos de Monitores de Máquinas Virtuais que se utilizam desta técnica, pode-se citar o Jail [27] e o User-Mode Linux [28].

Jail é um virtualizador do FreeBSD [29], que se utiliza do mesmo *Kernel* e sistemas de arquivos do sistema hospedeiro para prover diversos ambientes virtuais [30]. Diferente dos demais monitores de máquinas virtuais, o Jail não virtualiza uma máquina virtual completa, de modo que possa hospedar um Sistema Operacional com suas aplicações, mas, sim, cria um pequeno ambiente que se limita a replicar a estrutura de arquivos do Sistema Operacional sobre

o qual está portado (Figura 3). No entanto, se utilizando desta estratégia é possível portar diversas aplicações sobre o Jail, como por exemplo, um servidor *web*, que será completamente independente de outras aplicações que estejam hospedadas em outras instâncias do Jail, ou até mesmo no Sistema Operacional hospedeiro. Diversos outros *softwares*, como por exemplo, o User-Mode Linux, se utilizam da virtualização, no nível de Sistema Operacional, para prover a criação de ambientes virtuais de forma isolada entre si.

Os ambientes virtualizados providos por estas ferramentas apresentam algumas vantagens e desvantagens. Como principais vantagens, podemos citar a consolidação de servidores e melhor aproveitamento do *hardware*, pois como estas ferramentas não portam um Sistema Operacional completo, o consumo de recursos físicos é menor. Como desvantagem, estas ferramentas falham em não fornecer portabilidade para diferentes SOs, pois como virtualizam partes do Sistema Operacional hospedeiro, não podem virtualizar um Sistema Operacional diferente do mesmo.

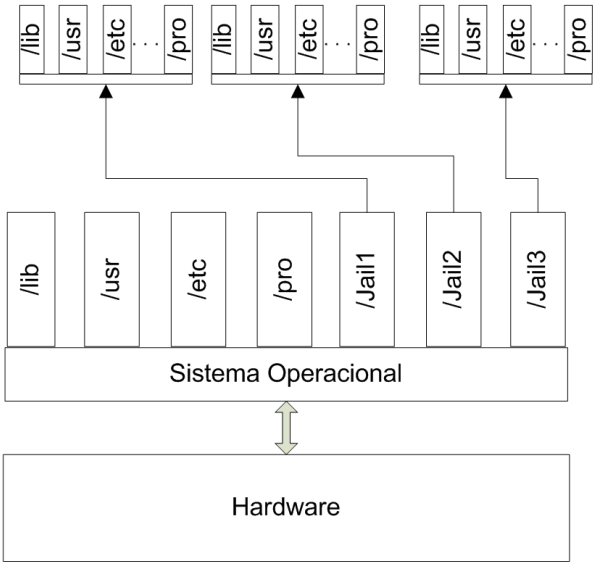

Figura 3 – Estrutura do Jail

2.4 Virtualização Através de Linguagens de Alto Nível

Em máquinas virtuais de alto nível, a camada de virtualização é inserida sobre o Sistema Operacional como se fosse um aplicativo. Este tipo de máquina virtual se utiliza basicamente da camada de virtualização para executar qualquer programa, sobre diferentes Sistemas Operacionais e *hardwares*, desde que tenham sido escritos e compilados conforme os seus padrões [31]. O Java [32] é um exemplo de linguagem de alto nível que se utiliza desta técnica.

A linguagem de programação Java surgiu em 1992 com o nome de Oak, cujo foco prin-

cipal era a portabilidade. Por este motivo, diferentemente das demais linguagens existentes, que são compiladas para código nativo da estrutura, a linguagem Java é compilada para um código intermediário, o *bytecode* que então é executado pela Máquina Virtual Java (*Java Virtual Machine-JVM*). Desta forma, um aplicativo escrito em Java pode ser portado, sem qualquer modificação, para qualquer Sistema Operacional, desde que exista uma JVM para o mesmo [33].

2.5 Técnicas de Virtualização

Em uma visão de alto nível, pode-se ainda classificar os VMMs quanto à técnica que utilizam para prover virtualização. Estas técnicas são: emulação, virtualização e paravirtualização [26].

Na emulação um *software* simula o comportamento de um computador, ou seja, ele traduz as instruções de um tipo de arquitetura para um formato de instruções suportado pela aplicação virtualizada (ver Figura 4). A utilização de emuladores não é interessante do ponto de vista do desempenho, pois necessita simular as instruções do processador e as características de cada *hardware*, o que a torna complexa e conseqüentemente lenta em comparação com as outras opções de virtualização. Embora ela também apresente algumas vantagens, como a não necessidade de alterações do Sistema Operacional hospedeiro ou do *hardware*.

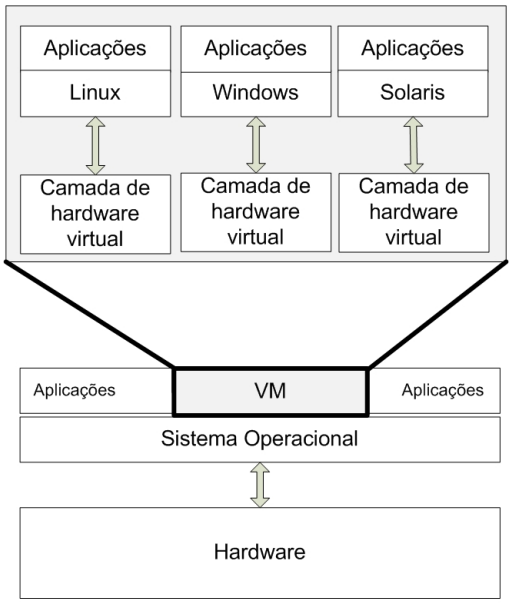

Figura 4 – Estrutura da Emulação

Outra técnica é a virtualização, que é implementada no nível de Sistema Operacional, em que a estrutura de *hardware* é completamente virtualizada, não sendo assim necessário executar modificações no Sistema Operacional convidado (ver Figura 5). Existem processadores,

AMD Pacifica [34] e Intel VT [35] com suporte a virtualização assistida, sem necessidade de modificações do Sistema Operacional.

Para contornar algumas características negativas da virtualização, como o baixo desempenho, surgiu como alternativa a paravirtualização. Ela difere da virtualização principalmente pelo fato do sistema convidado necessitar ser modificado, para que o acesso ao *hardware* seja simplificado (Figura 6). Esta medida melhora o desempenho, pois o *Hypervisor* não necessita executar outras operações além das delimitações de acesso à memória e ao armazenamento em disco pelas VMs.

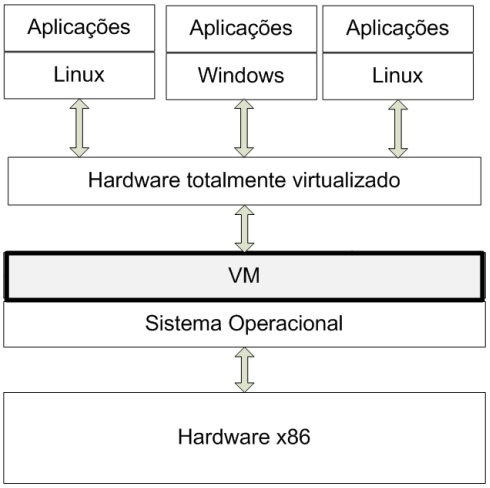

Figura 5 – Estrutura da Virtualização

O ponto negativo da paravirtualização é que a necessidade de alteração do sistema convidado limita a portabilidade do sistema (ver Figura 6).

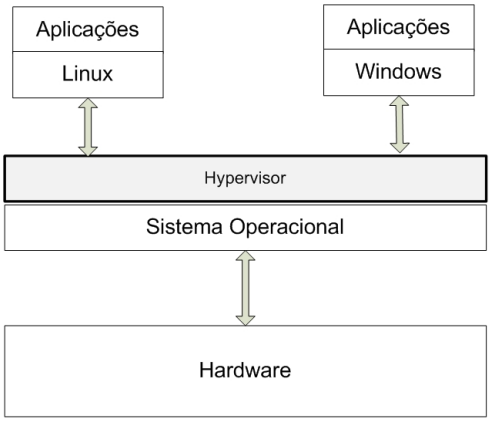

Figura 6 – Estrutura da Paravirtualização

2.6 Considerações Finais

Diversos fatores influenciaram o ressurgimento da virtualização dentro dos *data centers* das empresas. Atentas a estes fatores, diversas empresas de *hardware* e *software* desenvolveram soluções de virtualização que se utilizam de diversas técnicas para prover ambientes virtualizados. No entanto, quando se busca virtualizar um ambiente físico, é importante que se conheçam quais as características e necessidades das aplicações que são hospedadas nesse ambiente. Apenas desta forma pode-se conhecer qual a técnica de virtualização é ideal para suprir as necessidades das aplicações hospedadas no ambiente. Este capítulo apresentou diversas soluções de virtualização e suas respectivas técnicas. No entanto, por diversos fatores, como custo elevado da licença, baixo desempenho, necessidade de uma API (*Application Programming Interface*) para gerenciamento, entre outros, o Xen foi definido como o monitor de máquinas virtuais a ser utilizado neste trabalho.

3 Xen

Xen é um Monitor de Máquinas Virtuais (*Virtual Machine Monitor-VMM*) que foi desenvolvido pelo *System Research Group* da Universidade de Cambridge, e que se utiliza da paravirtualização para prover às VMs um desempenho próximo daquele obtido se elas estivessem diretamente sobre o *hardware* físico [36]. Ele é VMM para uma vasta gama de arquiteturas, como x86, x86-64 e PowerPC, que virtualiza uma estrutura física para diversas VMs (domU[1]) que podem hospedar uma variedade de Sistemas Operacionais, como Linux, FreeBSD, Solaris e Windows. Este capítulo apresenta o monitor de máquina virtual Xen, descrevendo sua estrutura interna e seus componentes.

3.1 Estrutura do Xen

A estrutura do Xen é composta por três componentes principais, o *Hypervisor*, domU e dom0[2] (ver Figura 7).

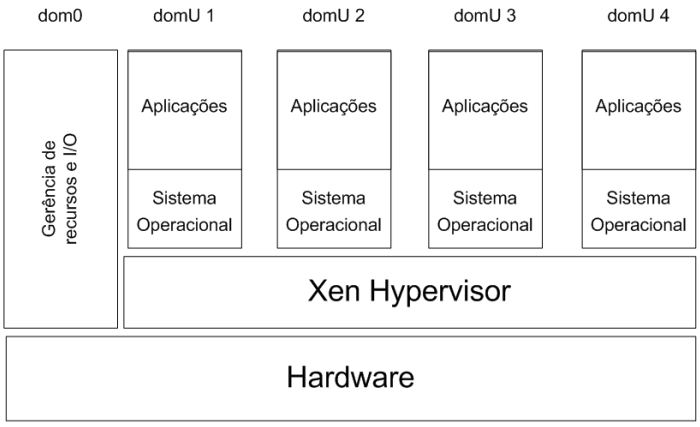

Figura 7 – Estrutura do Xen

O *Hypervisor* é uma camada inserida acima do *hardware*, e é responsável por implementar os recursos virtuais, como por exemplo, a memória virtualizada (Vmemória) e a CPU (*Central*

[1]Nomenclatura utilizada para se referenciar um sistema hóspede no Xen.

[2]Nomenclatura utilizada para se referenciar uma VM especial do Xen, que é utilizada para gerenciar as demais VMs.

Processing Unit) virtualizada (VCPU), a serem utilizados por um domU. Além disso, o *Hypervisor* deve ser capaz de interpretar algumas instruções dos Sistemas Operacionais hospedados nas VMs e repassar estas para o *hardware*.

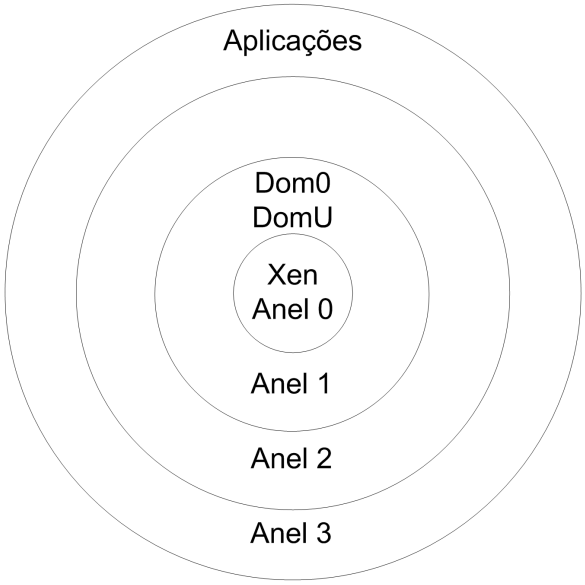

Figura 8 – Nível de privilégio do Xen [1]

Esses diversos componentes precisam executar em distintos níveis de privilégio, sendo que por ser um virtualizador da arquitetura x86, o Xen se utiliza do mesmo modelo de proteção, baseado em anéis. Os níveis de proteção da arquitetura x86 são baseados em quatro anéis, sendo que, quanto menor o nível, maior é o privilégio de acesso ao *hardware*. Na arquitetura x86/32 o anel 0 é o nível de privilégio onde se executa o Sistema Operacional (*Kernel*), os anéis 1 e 2 são utilizados para executar os serviços do Sistema Operacional (*drives* de dispositivos, por exemplo) e no anel 3 são executados os aplicativos dos usuários [37].

Pelo fato de que apenas o *Hypervisor* do Xen executa em nível 0, e os Sistemas Operacionais convidados executam em nível 1, qualquer instrução privilegiada, que o *Kernel* desses sistemas ou mesmo uma ferramenta de gerência necessite executar, deverá ser implementada através do *Hypervisor*. Para isso, o Xen disponibiliza um *daemon* que possibilita o acesso dessas ferramentas às funções do *Hypervisor*.

3.1.1 Xen API

Normalmente, a utilização do termo API (*Application Programming Interface*) do Xen costuma ser confuso, já que o Xen possui duas, a *Hypercall API* que é utilizada pelas VMs e a Xen *Management API* que é utilizada pelas ferramentas de gerência e monitoração. A Xen *Management* API é usada pelas ferramentas do Xen, tais como a ferramenta de linha de co-

mando *xm* para gerenciar o sistema. O *daemon* Xend aguarda por conexões dessas ferramentas através de XML-RPC, e em seguida, executa as funções administrativas solicitadas, que inclui, por exemplo, todo o gerenciamento do ciclo de vida da VM. O Xend é um *daemon* escrito em Python para prover uma interface para algumas funções do *Hypervisor* do Xen. O Xend controla os recursos virtualizados e provê diversas funções que são utilizadas pelas ferramentas de gerência e monitoração, como o Xen Master (*xm*) e o Xen Monitor (XenMon) [3]. Outra funcionalidade importante que o Xend provê, é o acesso aos consoles dos domínios através de um servidor HTTP (*Hypertext Transfer Protocol*). Mais detalhes sobre os argumentos que podem ser utilizados no Xend podem ser encontrados em [38].

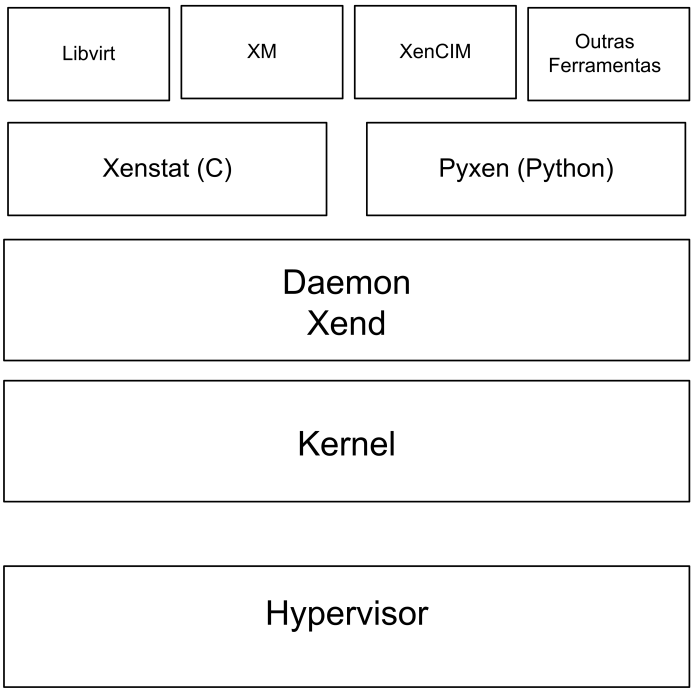

Figura 9 – Hierarquia da API do Xen

Quando uma ferramenta solicita a execução de um comando, esta solicitação deve ser em um primeiro momento traduzida para XML (*eXtensible Markup Language*), através da utilização de uma biblioteca, como a libxen para C ou a pyxen para Python. Uma função da biblioteca envia o XML através de RPC (*Remote Procedure Call*) para o Xend, que então processa o pedido e o transfere para o *Kernel* que o repassa para o *Hypervisor* através da *Hypercall* API. Deste modo, as ferramentas de monitoração e gerência, que estejam sendo executadas nos domU, podem realizar operações que estariam restritas ao dom0.

3.1.2 Ferramentas de Monitoração e Gerência no Xen

O Xen pode se utilizar de diversas ferramentas, próprias ou de terceiros, para gerenciar e monitorar os domínios, sendo o Xen Master (*xm*) a principal ferramenta disponibilizada para executar a gerência dos domU. Ele é totalmente escrito em Python e se comunica com o Xend através de um protocolo próprio, sendo que até a versão 3.1 do Xen era a única ferramenta de gerência do Xen.

Por agregar em uma única ferramenta diversas funcionalidades, o *xm* possui uma grande flexibilidade, podendo ser utilizado tanto como ferramenta de gerência, como de monitoração de ambientes virtualizados. Para gerência, ele disponibiliza uma grande variedade de comandos que possibilitam, por exemplo, criar, pausar, destruir e migrar domínios. Ele também pode ser utilizado para definir qual o escalonador a ser utilizado pelo Xen, bem como seus atributos, como percentual do processador (CAP) e peso. Apesar de não ser uma ferramenta de monitoração com muitos recursos, o *xm* disponibiliza a monitoração de algumas métricas, como o estado de um domínio (executando, pausa, bloqueado) e a quantidade de CPU e memória consumida por determinado domínio. Entretanto, caso seja necessário monitorar outras métricas, além dessas disponibilizadas pelo *xm*, pode-se recorrer a ferramentas de monitoração desenvolvidas por terceiros, como o XenMon [3] e a LibVirt [39].

O XenMon é uma ferramenta de monitoração desenvolvida pelo HP Labs para prover informações detalhadas do uso do processador pelos domínios no Xen. Sua estrutura é composta por três componentes (Ver figura 10), que são:

- Xentrace: é um componente, utilizado para a captura de eventos, nativo do Xen. Através de sua utilização é possível capturar determinados eventos no MMV.
- Xenbaked: os eventos capturados pelo XenTrace não são informações relevantes, caso não sejam tratadas. O Xenbaked é um processo que trata os eventos gerados pelo Xentrace e os processa para serem posteriormente utilizadas. É possível customizar as métricas monitoradas, o intervalo entre cada monitoração e por quanto tempo devem ser mantidas essas informações no histórico da aplicação.
- XenMon: é o *front-end* utilizado para exibir as informações tratadas pelo Xenbaked.

Dentre as diversas métricas que podem ser monitoradas pelo XenMon, podemos citar, por exemplo:

- Taxa de utilização do processador: é a porcentagem de uma unidade de tempo (por padrão 1 segundo), no qual um domínio fez uso efetivo do processador.
- Tempo de bloqueio: é a porcentagem de tempo gasto por um domínio aguardando por um evento de I/O (*Input/Output*).

- Tempo de espera: tempo gasto por um domínio aguardando na fila de execução do processador.

Existem ainda, diversas outras ferramentas de monitoração para o Xen, como a LibVirt e Xentop [40]. A LibVirt é uma biblioteca que provê diversas informações, bastante similar as fornecidas pelo *xm*, sobre os domínios que executam sobre o Xen. As principais vantagens de se utilizar a LibVirt, é que ela é portável para outros VMMs, como o QEMU e o fato de dispor de uma interface gráfica. O Xentop é uma ferramenta baseada no comando *top* do Linux, sendo que o Xentop, de forma análoga ao *top*, exibe uma lista dos processos em execução e a respectiva porcentagem de memória e processador consumido. No entanto, o Xentop exibe a lista de processos por domU e o *top* exibe a lista de processo de um único SO.

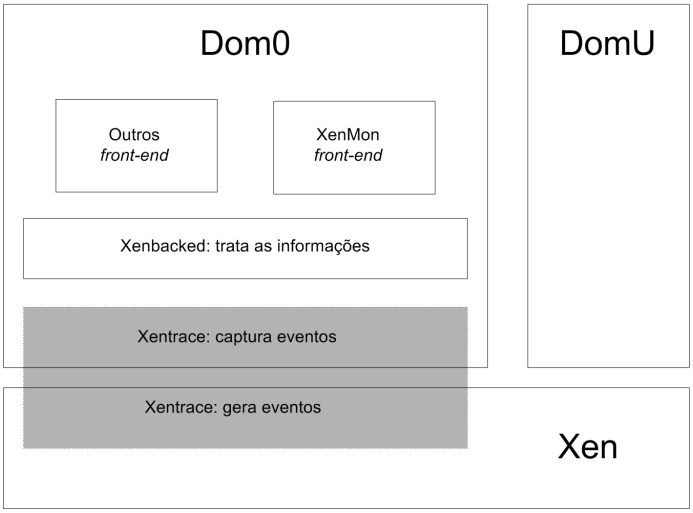

Figura 10 – Arquitetura XenMon

Como apresentado, o Xen possui diversas ferramentas de monitoração, como o Xentop e XenMon, que são utilizadas para fornecerem informações relevantes sobre o funcionamento do sistema. No entanto, caso seja necessário se utilizar das métricas de desempenho dos domínios, como por exemplo, para o desenvolvimento de um sistema, é importante que estas informações sejam obtidas diretamente da biblioteca de programação *xenstat*, já que esta opção elimina a necessidade de se utilizar uma ferramenta de monitoração, melhorando o desempenho da aplicação. A *xenstat* possibilita a um desenvolvedor obter diversas informações sobre a utilização dos recursos pelos domínios, como utilização da rede, do processador e da memória, sem utilizar ferramentas de monitoração de alto nível, como as já apresentadas. Essas informações, obtidas através da *xenstat*, podem ser utilizadas por exemplo, para se definir a quantidade de recursos que deve ser alocada para um domínio no Xen.

Apesar das diversas ferramentas descritas nesta seção fornecerem informações relevantes sobre o desempenho dos domU, apenas o *xm* disponibiliza algumas funcionalidades, como configurar em tempo de execução, em determinado escalonador do Xen, o percentual da memória

e do processador que estarão disponível para um determinado domínio.

3.2 Escalonadores do Xen

O Xen se difere dos demais VMM, pelo fato de permitir ao administrador da estrutura virtualizada definir qual o escalonador irá ser utilizado. Em suas primeiras versões o Xen oferecia uma grande variedade de escalonadores, como por exemplo, o Round Robin, o Borrowed Virtual Time (BVT) e o Atropos [41].

O Round Robin foi um dos escalonadores presentes no Xen até a versão 2.0, mas diferentes dos demais, ele era utilizado apenas para teste e para demonstrar como o Xen interagia através do Xend com o escalonador, não sendo recomendada a sua utilização em um ambiente virtualizado que hospede uma aplicação comercial.

O BVT também esteve presente até a versão 2.0 do Xen, mas diferente do Round Roubin ele era o escalonador padrão desta versão. O BVT trabalha utilizando o conceito de fatia de tempo virtual, que garante uma fatia mínima de tempo no processador para cada domínio. No entanto, como os domínios hospedados pelo Xen possuem diferentes necessidades de processamento, é interessante, do ponto de vista do desempenho, distribuir as fatias de tempo conforme a demanda das VMs. Por este motivo, o BVT implementa um mecanismo chamado de *warp* que possibilita que cada domU altere, dentro de um limite informado pelo administrador, a fatia de tempo que lhe foi atribuída [42]. Este mecanismo possibilita uma melhora do desempenho das aplicações hospedadas em um domU, quando o mesmo está realizando operações que necessitam de uma grande capacidade de processamento.

Outro escalonador que era disponibilizado pelo Xen, é o Atropos. O Atropos estava presente da versão 2.0 até a 3.0 do Xen, sendo que ele pode ser classificado como um escalonador *soft realtime* [43]. Seu algoritmo garante que cada domU irá executar por *n* milissegundos a cada *m* milissegundos do tempo real. Esta abordagem é excelente para VMs que hospedam aplicações sensíveis à latência, mas acaba penalizando VMs que hospedam aplicações que fazem uso intenso do processador.

No entanto, a partir da versão 3.0 do Xen, todos estes escalonadores anteriormente descritos, deixaram de ser suportados, sendo disponibilizados apenas dois: o SEDF (*Simple Earliest Deadline First*) e o Credit.

O SEDF é um escalonador desenvolvido para ser utilizado por ambientes virtualizados que hospedam aplicações que necessitam trabalhar com uma latência bastante reduzida. Assim como o Atropos, o SEDF precisa que dois parâmetros sejam configurados: o tamanho de uma fatia de tempo em milissegundos que cada domínio terá direito de utilizar o processador e qual o tamanho do intervalo em milissegundos entre cada utilização por um mesmo domínio. Para exemplificar o funcionamento do algoritmo do SEDF, pode-se utilizar, como exemplo, uma estrutura virtualizada que hospeda três domínios com as seguintes configurações:

- **Domínio 1**: 15ms de utilização do processador a cada 80ms;
- **Domínio 2**: 2ms de utilização do processador a cada 10ms;
- **Domínio 3**: 5ms de utilização do processador a cada 10ms.

Inicialmente, o algoritmo realiza uma varredura e verifica qual domínio possui o menor *deadline*. Neste caso, os domínios 2 e 3 possuem um intervalo menor para terem acesso a sua fatia de tempo, porque ambos necessitam escalonar dentro de 10ms. Desta forma, o escalonador libera o domínio 3 para iniciar a utilização de sua fatia de tempo, enquanto o domínio 2 espera para executar (ele pode aguardar 8ms). Após ter executado o domínio 3, o algoritmo irá executar o próximo domínio com menor *deadline*, neste caso o domínio 2, enquanto o domínio 1 e 3 aguardam. Após a execução do domínio 2, o escalonador realiza uma nova busca pelo menor *deadline*, que é novamente o domínio 3. Estes dois domínios serão escalonados durante aproximadamente 65ms, até que o domínio 1 tenha que ser executado. Quando o domínio 1 ganhar o direito de uso do escalonador, ele poderá executar pelos próximos 15ms. No entanto, caso isso venha a acontecer, os domínios 2 e 3 não conseguirão executar antes do *deadline*. Para estes casos, o SEDF possui um mecanismo que detecta estas situações e diminui a fatia de tempo do domínio 1 para que os demais domínios executem antes do *deadline*.

Apesar de ser um escalonador que possui mecanismos que o tornam eficiente para aplicações de tempo real, o fato de não executar em máquina SMP (*Symmetric Multi-Processor*) tornaram o SEDF pouco utilizado, sendo que já não é prevista a sua inclusão na próxima versão do Xen. Esta decisão possibilita que o Credit se consolide ainda mais como escalonador padrão do Xen.

O escalonador Credit é o atual escalonador padrão do Xen, sendo que ele é otimizado para uso em máquinas SMP, pelo fato de que caso exista uma CPU física em *idle*, o Credit possibilita a realocação automática de VCPUs das filas de execução de outras CPUs físicas para a CPU ociosa. Desta forma, ele garante que enquanto existir uma CPU física com uma fila de VCPUs para execução, outras CPUs físicas que se encontrem em *idle*, poderão realocar estas VCPUs para suas filas de execução.

Assim como outros escalonadores do Xen, o Credit possibilita que duas propriedades possam ser configuradas para cada domínio, o *Weight* (peso) e o CAP (percentual da CPU atribuído a determinado domínio). O peso de um domínio está relacionado à prioridade que este domínio terá para utilizar o processador. Por exemplo, em determinado ambiente virtualizado onde haja concorrência, um domínio de peso 512 terá duas vezes mais prioridade para utilizar o processador que um domínio de peso 256. Mas caso todos os domínios possuam o mesmo valor de peso, não importa se este peso é 256 ou 512, a preferência de acesso a CPU é igual para todos.

Diferente do *weight*, o CAP é um valor absoluto, sendo que este valor representa a porcentagem máxima de uma CPU que pode ser atribuída ao domínio. Por exemplo, para se configurar a totalidade de uma CPU para um domínio, com o CAP, define-se o valor de CAP como 100, para se configurar a metade de uma CPU para um domínio, define-se seu valor de CAP como

50 e para configurar 4 processadores para um domínio, configura-se o CAP com o valor 400. No entanto, caso o valor atribuído ao CAP seja “0”, este domínio não irá possuir um limite de processamento, e caso necessário, poderá receber uma ou mais CPUs. Desta forma irá garantir uma melhor utilização do processador e que cada domínio receba o tempo de processamento que foi concedido. Para isso, no entanto, é necessário que exista pelo menos uma CPU sem ser utilizada entre as CPUs disponíveis.

No Credit, cada processador possui uma fila de VCPU, sendo que cada VCPU pode ter um nível de prioridade. Quando uma VCPU é atribuída a uma fila de execução de um processador, sua ordem de inserção é definida pela sua prioridade. Desta forma, uma VCPU será inserida em uma determinada posição de uma fila, de modo que fique atrás das demais VCPUs que possuem uma prioridade igual ou maior. Cada VCPU consome créditos do domínio enquanto é executada. A cada 30ms[3] o escalonador recalcula os créditos dos domínios para saber quanto cada um gastou ou ganhou. Caso um domínio inicie com uma prioridade alta, conforme ele for consumindo seus créditos, irá perder prioridade, até ter uma prioridade baixa.

Apesar do Credit ser um escalonador amplamente difundido, ele possui algumas limitações, como por exemplo, o fato de permitir apenas a realocação de recursos através da migração de VCPUs entre CPUs físicas. No entanto, para isso ser possível, é necessário que sempre exista uma CPU física ociosa. Só que, esta situação não é comum, nem desejável em um ambiente empresarial. Desta forma, esta limitação impede que o escalonador garanta determinado nível de recursos para um domínio, o que acaba impossibilitando que seja garantida a qualidade do serviço de uma aplicação através da utilização de SLAs (*Service Level Agreement*) [44].

3.3 Alocação de Recursos no Xen

Atualmente, como já foi apresentado, o Xen possui a funcionalidade de poder se utilizar de um dos dois escalonadores disponíveis: o Credit ou SEDF. Comparando-se esses dois escalonadores, apenas o Credit possui a funcionalidade de realocar recursos físicos entre as VMs em tempo de execução ou de definir um limite máximo de recursos físicos que cada domínio pode utilizar.

Atualmente o escalonador Credit, possibilita a configuração em um de dois distintos modos: estático e dinâmico. Configurado no modo dinâmico, o Credit realiza a realocação dos recursos entre os domínios conforme a necessidade momentânea dos mesmos por recursos. No entanto, a falta de um limite máximo de processamento para cada domínio implica em algumas limitações ao uso deste modo.

Uma das principais limitações, ao uso do modo dinâmico, é a sua utilização em ambientes que necessitem de certo grau de isolamento de desempenho entre os domínios, já que caso

[3]O Credit utiliza como padrão fatias de tempo de uso do processador de 30ms para cada domínio

um domínio apresente um defeito e venha a consumir uma grande porcentagem de recursos do sistema, os demais domínios da mesma estrutura física irão apresentar uma queda significativa em seu desempenho.

Outra limitação ao uso do modo dinâmico é a sua utilização em ambientes virtuais que necessitam garantir uma porcentagem mínima de recursos físicos para determinado domínio, por exemplo, em ambientes virtualizados que se utilizam de um SLA (ver Capítulo 4). Nesses ambientes, é essencial que se garanta o limite de recursos acordado no SLA. Desta forma, é importante que se utilize um escalonador que possua esta funcionalidade, como o escalonador Credit, configurado em modo estático.

O Credit, quando configurado em modo estático, possibilita a funcionalidade de se definir uma porcentagem de recursos (CPU e memória) para cada domínio da estrutura. Para isso, no entanto, é necessário que a configuração dos mesmos seja executada no momento da criação dos domínios, sendo que a soma de todos os percentuais não pode ser superior ao percentual total disponibilizado pela estrutura. Após a inicialização dos domínios, é possivel alterar os percentuais definidos para cada domínio, de forma manual, através do comando *xm*. No entanto, para se definir um percentual de recurso para um domínio, é necessário que se conheça o funcionamento de uma ambiente virtualizado do Xen, quando o mesmo estiver configurado com o escalonador Credit em modo estático.

3.4 Considerações Finais

Como foi apresentado nesse capítulo, o Xen possui uma variedade de escalonadores que podem ser configurados e utilizados, conforme as necessidades das aplicações hospedadas nas VMs. Contudo, apesar da existência desses diversos escalonadores, nenhum deles possibilita ao Xen realocar recursos não utilizados e ao mesmo tempo garantir certo nível de recursos para cada VM da estrutura. Casos essas duas funcionalidades fossem providos pelo escalonador, ou por um sistema externo ao mesmo, possibilitaria ao Xen garantir certo nível de recurso para uma aplicação e desta forma possibilitar a utilização de SLAs em ambientes virtualizados.

Apesar do escalonador Credit em modo estático oferecer, com limitações, a possibilidade de se utilizar SLAs no Xen, ele possui algumas restrições, como por exemplo, a impossibilidade de alterar o nível de recurso atribuído a uma VM caso o SLA seja alterado, e a incapacidade de realocar recursos não utilizados pelas demais VMs, para garantir o SLA de uma aplicação que sofreu uma sobrecarga no número de usuários.

4 *Service Level Agreement- SLA*

Aplicações de empresas são normalmente compostas de um grande número de componentes que interagem entre si de uma maneira complexa, sendo necessário um grande número de políticas de monitoração. Um SLA pode especificar várias necessidades de serviço, que incluem métricas como disponibilidade, tempo de resposta, processamento, garantia e assim por diante [44]. Estes valores de níveis de serviços, quando expressos em um SLA, tomam a forma de *Service Level Objectives (SLOs)*, e são aplicados nas métricas de desempenho, conhecidas como *Service Level Indicators (SLIs)*. "Disponibilidade" e "desempenho" são exemplos de SLOs utilizados em um SLA, enquanto que os valores 95% de disponibilidade ou 1,3 segundos de tempo de resposta são exemplos de valores de um SLIs [45]. Apesar de SLO e SLI serem os mais importantes parâmetros, um SLA pode ainda possuir diversos outros que podem ser definidos, tais como penalidades a serem pagas pelo provedor do serviço quando este falhar em cumprir o nível de serviço prometido e recompensas que serão pagas pelo cliente do serviço, quando o provedor do serviço superar as expectativas do nível de serviço acordado [46].

Os *Service Level Indicators (SLIs)* podem ser definidos como os principais componentes de qualquer SLA. SLIs também podem ser definidos como as medidas que são de fato realizadas para representar os SLOs, e sendo que eles podem ser medidas e quantificadas [47]. Eles permitem que o SLA seja mensurado e assim prover a avaliação de que uma obrigação está sendo ou não cumprida. Cada aspecto ou objetivo (SLO) do SLA, como disponibilidade, terá um nível designado para ser alcançado. Mas um SLA pode incluir múltiplas medidas ou indicadores (SLIs) para cada SLO. A Figura 11, mostra um exemplo de como um SLO pode ser decomposto em dois SLIs. Na figura, o SLA é composto de dois SLOs: Desempenho e Disponibilidade; enquanto que o SLO Desempenho é decomposto nos SLIs Tempo de Resposta e Carga de Trabalho. Os SLOs definem o nível de serviço que será provido, em acordo com o que foi definido pelas partes envolvidas. Eles são articulados dentro do contexto de metas empresariais e podem conter um ou mais *Service Level Indicators(SLIs)*. Disponibilidade, desempenho e tempo médio de reparo são alguns exemplos de medidas a serem monitoradas pelos SLOs e que são encontrados na grande maioria dos SLAs em Computação. A definição dos SLOs é diretamente influenciada pelo ambiente onde aplicamos o SLA.

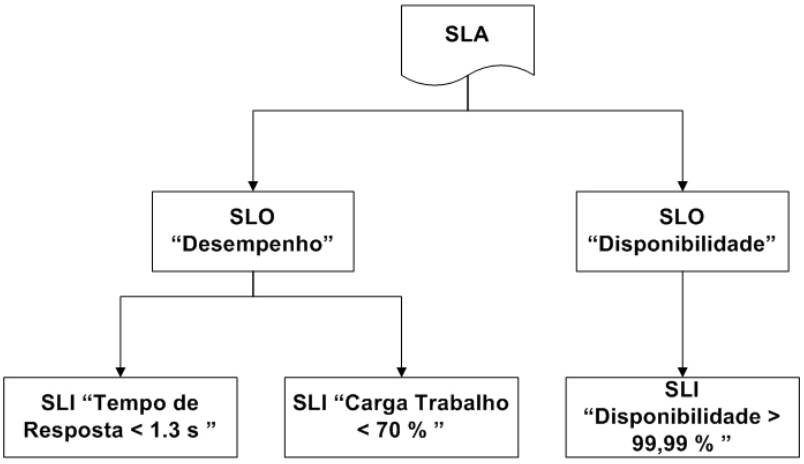

Figura 11 – Estrutura de um SLA

4.1 Aplicações de SLA

Possuindo como característica um grande campo de uso, atualmente pode-se encontrar a utilização de SLAs em um grande número de áreas, nas mais diversas atividades. Nesta seção descreve-se a utilização de SLAs, com o objetivo de demonstrar o seu uso, em diversas áreas da computação, por exemplo, em provedores de acesso à *Internet*, em *clusters*, em *grids* e na virtualização.

4.1.1 *Service Level Agreement* aplicado em servidores ISP

Um Provedor de Serviços de *Internet (ISP-Internet Service Provider)* é responsável principalmente por oferecer acesso à *internet*, além de agregar outros serviços como *e-mail, blogs* e hospedagem de *home page*. Para que os usuários possam utilizar esses serviços, é necessário formalizar um contrato com o provedor, onde, neste contrato, ficam estabelecidas as responsabilidades de cada parte. Sendo o provedor responsável por disponibilizar o serviço, e o usuário a fazer uso do mesmo de acordo com as normas do provedor e, na maioria das vezes, a pagar uma tarifa pelo serviço prestado. O modelo de contrato descrito, na maioria dos casos, cobre as necessidades do usuário doméstico, mas não de um usuário empresarial, como um banco ou uma grande empresa. Isto se deve ao fato de que, diferente dos usuários domésticos, essas empresas possuem necessidades especiais, como disponibilidade, largura de banda, tempo de recuperação entre falhas, etc. O SLA também será utilizado para especificar o que o provedor pode esperar de seus clientes em termos de tráfego de rede e utilização dos recursos da rede [48].

Um SLA completo para servidores ISP é um acordo formal, que contém as partes envolvidas, os termos do acordo específicos para cada contrato e as suas responsabilidades, do provedor

e do cliente, tais como serviço de suporte, penalidades em caso de não cumprimento do acordo, tarifas e relatórios. Vale ressaltar que outras métricas podem ser definidas, conforme a necessidade do cliente ou do provedor. Com base nestas necessidades e possuindo como perspectiva a *Internet* 2, que possui como meta fornecer uma infraestrutura com uma ampla largura de banda e ainda disponibilizar novas tecnologias capazes de suportar aplicações interativas, serviços integrados e serviços diferenciados, os provedores ISPs começam a buscar formas eficientes de definir SLAs com o objetivo de suprir as novas necessidades. Em [49] é abordada a utilização de SLAs como forma de garantir largura de banda para aplicações que necessitem de grande largura de banda para proverem serviços com qualidade. Outro trabalho descreve estudos de caso de distribuição de vídeo sob demanda em uma infra-estrutura composta por implementações que abrangem parte dos Modelos de Serviço Integrado e Serviço Diferenciado [50]. O estudo de caso proposto é o de um cliente que deseja receber um serviço diferenciado do seu provedor de serviços *internet*, neste caso ele deve ter um SLA com seu ISP. Este SLA basicamente especifica as classes de serviços suportadas e a quantidade de tráfego permitida em cada classe. Os resultados alcançados validarão a importância de realizar reservas de recursos através de SLAs na infra-estrutura de rede e que tais reservas são viáveis quando realizadas no nível de rede da *internet*.

Outra proposta de utilização de SLAs em servidores ISP foi utilizar DiffServ[1] em conjunto com métodos destinados a melhorar as condições de segurança do modelo DiffServ [51], pois ataques como *man-in-the-middle* e *Denial of QoS (DQoS)* permanecem por resolver no modelo [52]. Com o modelo proposto, o cliente que pretenda originar um fluxo terá de se autenticar antes em um servidor de autenticação que lhe disponibilizará permissões de acessar o servidor de QoS (*Quality of Service*). Somente após a autenticação, o cliente terá acesso aos recursos de rede, e a partir deste momento, o seu fluxo será de acordo com o SLA pré-estabelecido pelo cliente com o provedor. Esta proposta apresenta-se como uma possível solução para problemas de segurança em redes com grande largura de banda e demonstra a potencialidade da negociação de QoS com autenticação dos clientes juntamente com SLAs [53].

4.1.2 SLA aplicado em redes de computadores

As redes de computadores têm apresentado avanços tecnológicos notáveis e se tornam cada vez mais complexas. Tanto os programas como os equipamentos evoluem na mesma velocidade, para acompanhar esse ritmo e se manterem atualizadas, as empresas destinam investimentos cada vez maiores nesse tipo de tecnologia, que nem sempre é o foco do seu negócio. Com o objetivo de contornar essa situação, muitas empresas estão optando por terceirizar essas áreas, contratando prestadores de serviços especializados.

Com o objetivo de garantir um nível de serviço adequado às necessidades da empresa, um

[1] Método utilizado na tentativa de conseguir qualidade de serviço em redes

SLA é negociado entre as partes com o objetivo de especificar os parâmetros da qualidade do serviço (QoS) e garantir que estes parâmetros sejam obedecidos na vigência do contrato. A negociação de um SLA em redes é o processo de selecionar parâmetros de QoS aplicáveis no SLA, sendo também negociadas as penalidades, no caso de violação do SLA (ver Figura 12). Em um SLA de redes, vários parâmetros podem ser negociados, pois, dependendo das características da rede um parâmetro pode ser ou não importante. Os parâmetros mais comuns de serem monitorados são disponibilidade, latência, largura de banda, tempo médio entre defeitos *(MTBF- Mean Time Between Failures)* e tempo médio para restaurar o serviço após um defeito *(MTTR - Mean Time To Recover)*. Após a definição do SLA, é preciso monitorar os parâmetros de QoS com o objetivo de se identificar degradação ou violação do contrato. Quando uma violação de um parâmetro de QoS é detectado, a ferramenta de gerência do SLA analisa a razão por que a degradação ocorreu e qual o parâmetro de QoS foi ultrapassado, em seguida notifica o cliente da violação do SLA [54].

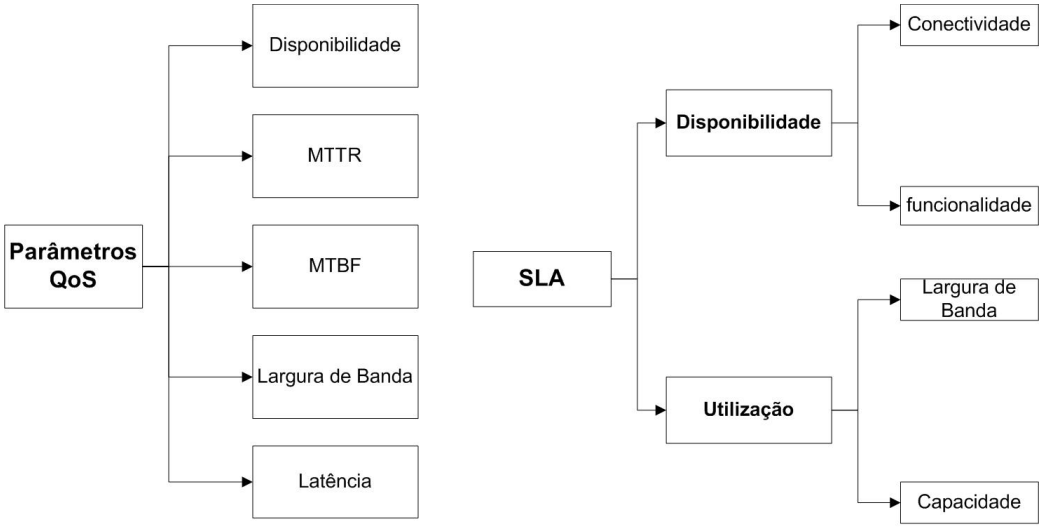

Figura 12 – Definição SLA com base QoS

4.1.3 SLA aplicado em *Cluster*

Ambientes de pesquisa e até mesmo grandes corporações vêm necessitando de alto poder computacional a baixo custo, para as mais variadas tarefas, como renderizar gráficos em alta velocidade, prever o clima e outras tarefas que exigem alto desempenho [55]. Os *cluster* de computadores, são utilizados para suprir uma grande gama de aplicações de alto desempenho e alta disponibilidade. Trabalhos submetidos em um *cluster* têm exigências variadas, dependendo de necessidades específicas e das expectativas do usuário. Com base no conceito de *cluster*, voltado a serviços, um número cada vez maior de empresas, tais como IBM, HP e SUN, estão oferecendo de maneira agressiva no mercado serviços que fornecem entrega dinâmica de pro-

cessamento, ou seja, os usuários pagam somente os recursos computacionais que utilizam na execução de sua aplicação. A partir desta nova abordagem, surgiu a necessidade de se garantir que o nível da qualidade destes serviços seja alcançado. A utilização de SLAs, é proposta, para que o *cluster* cumpra as obrigações contratuais negociadas e concordadas entre a empresa e os usuários do serviço, ou seja, o SLA tem que assegurar-se de que os níveis contratados de desempenho e do serviço sejam cumpridos [56].

4.1.4 SLA aplicado em *Grids* de computadores

Algumas áreas de conhecimento como a astronomia, meteorologia e genética necessitam de muitos recursos computacionais, sendo aplicações que necessitam de um alto desempenho para realizar cálculos complexos e repetitivos. As grades de computação surgiram com uma nova proposta para o processamento de alto desempenho, que é a criação de um ambiente de colaboração, com o uso de recursos geograficamente distantes e heterogêneos para processamento de alto desempenho [52]. Este ambiente de computação, no entanto, não possui um controle efetivo sobre o processamento, onde ele ocorre, o tempo de processamento de determinada tarefa, e nem mesmo garante a entrega do resultado do processamento. Sendo assim, a utilização de *Grids* com propósitos comerciais, certamente necessita de um controle mais rígido e deve ser baseado em SLAs, para que os objetivos do cliente sejam atendidos. Com base nestas necessidades, foi proposta uma linguagem para especificar e monitorar estes contratos de maneira clara e sem ambigüidades, como uma *engine*, para avaliar e monitorar estes SLAs e um protocolo para obter e se agregar medições de SLAs em vários *sites* [57].

4.1.5 SLA orientado à segurança

A segurança de sistemas está entre as áreas da computação com maior importância, devido especialmente ao impacto de uma falha que poderia expor dados sensíveis. O SLA orientado a segurança define os cuidados ou deveres relacionados à segurança que o provedor do serviço deve disponibilizar. Normalmente um SLA define aspectos como disponibilidade do canal de comunicação, a vazão média, taxa de erros e picos de congestionamento. O modelo de um SLA de redes difere de um SLA de segurança basicamente no enfoque. Um SLA de segurança estaria focado em métricas relacionadas à criptografia, à integridade da informação que trafega na rede e aos registros de transporte de mensagens no canal. A análise da política de segurança é importante para delimitar as métricas de segurança do SLA. Verificações da segurança do ambiente onde o contrato será implantado, têm por objetivo definir mecanismos de segurança para que um nível de segurança seja alcançado e avaliar se o SLA cobre as expectativas descritas pelo cliente e pelo fornecedor. Um fato importante a ressaltar é que o SLA não substitui ferramentas

de segurança, sendo seu objetivo definir níveis aceitáveis de segurança da informação para a organização [58].

4.1.6 SLA em máquinas virtuais

Como já apresentado no Capítulo 2, a utilização da virtualização para a consolidação de servidores resultou em um grande número de benefícios, tais como redução de custo de *hardware* e *software*, do consumo de energia e da facilidade no gerenciamento do ambiente. A consolidação no entanto não acrescentou melhorias significativas na qualidade do serviço oferecido pelos ambientes virtualizados. Com o objetivo de implementar garantias de qualidade do serviço em ambientes virtuais buscou-se implementar SLAs nesses ambientes. Um SLA em um ambiente virtualizado pode ser utilizado para garantir níveis de QoS para uma aplicação executando em um servidor virtual ou entre dois ou mais servidores virtuais.

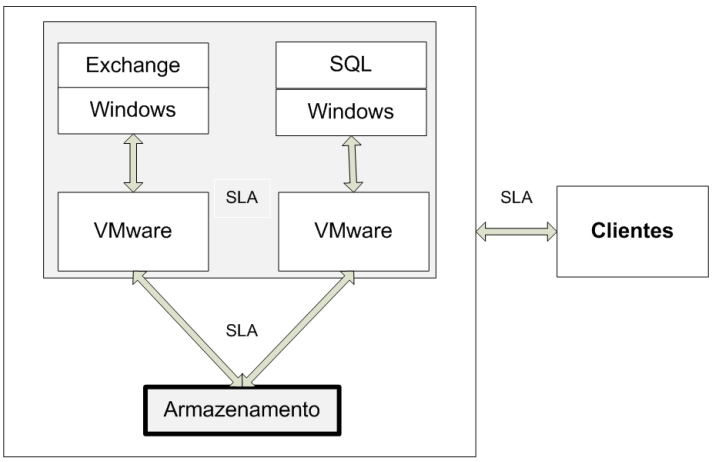

Figura 13 – Estrutura de um SLA no VMware

Um exemplo do uso de SLAs em ambientes virtuais foi proposto inicialmente pelo VMware, onde foi utilizado para estipular o nível de recursos e prioridades de um conjunto de contas de usuário em um servidor de *e-mail*. O ambiente de infra-estrutura que hospeda este tipo de serviço é o de uma empresa que disponibiliza serviços a seus clientes, e que virtualiza estes serviços em um servidor de *e-mail Exchange* e em um servidor de banco de dados SQL server. Estes servidores compartilham uma mesma estrutura, com capacidade de armazenamento compartilhada (ver Figura 13). Os clientes foram agrupados de acordo com o contrato de nível de serviço firmado, em quatro tipos de contas, platina, bronze, prata e ouro. Esta classificação permite uma simplificação das políticas e da gerência do SLA, pois com base na classificação pode-se definir por exemplo o tamanho da caixa de mensagem, das mensagens e também as políticas de *backup* e de recuperação de desastres. Por exemplo, caso um servidor seja afetado por um desastre, o grupo de contas ouro teria um *restore* mais elevado do que uma conta bronze devido às políticas de SLA negociadas. Este tipo de solução é viável, onde uma solução física

completa não seria possível devido a restrições financeiras, de espaço físico, ou na exigência de manutenção urgente sem que os serviços sejam parados. A solução virtual apresentada fornece um nível de desempenho aceitável para prover um serviço de acordo com as necessidades dos usuários [59].

Outra abordagem do uso de SLAs em ambientes virtualizados, foi proposta pela Microsoft para o Virtual Server. Como já apresentado no Capítulo 2, o Virtual Server é capaz de realizar a migração de VMs entre diferentes *host* fisícos. Desta forma, caso uma VM venha a sofrer degradação do desempenho, uma solução é transferir esta VM para um outro *host* físico. Por exemplo (ver Figura 14), o *host* físico ABC começou a sofrer degradação de desempenho, quando a utilização do processador alcançou 90%, sendo que *host* XYZ está utilizando 50% do processador. Para que um possível SLA não seja quebrado, transfere-se uma VM do *host* ABC para o *host* XYZ. Esta ação minimiza o problema de desempenho no *host* ABC ao balancear a carga para que os dois *hosts* possuam uma carga em torno de 70% do processador.

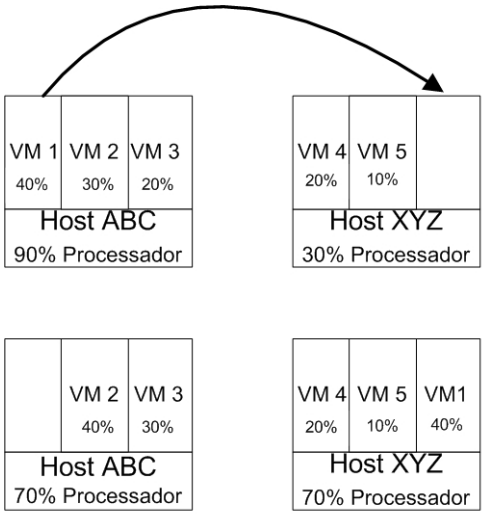

Figura 14 – Migração de VM com Virtual Server

Com base nestes resultados, a Microsoft pesquisa como definir métricas de SLA para construir um modelo que defina as necessidades da estrutura, com base nas necessidades dos clientes e que possibilite a utilização de um número maior de VMs por *host*.

Apesar dos casos de uso apresentados possibilitarem a utilização de SLAs em ambientes virtualizados, os mesmos não buscam se utilizar de uma das principais funcionalidades da virtualização: a realocação de recursos entre domínios. Esta funcionalidade provê a capacidade de se utilizar SLA, com o objetivo de se garantir determinado nível de recursos físicos (porcentagem de processador e memória) para uma aplicação, e até mesmo em alguns casos, manter os SLIs de um contrato, quando o cliente quebra o acordo.

No entanto, antes de se portar determinada aplicação para um ambiente virtualizado é necessário que se conheça o nível de recursos físicos que ela necessita para fornecer um serviço com qualidade. Em [5] é apresentada uma abordagem utilizando o Xen como VMM (Ver capítulo

3), que busca definir de maneira precisa qual a porcentagem de recursos (processador e memória) é necessária para que uma aplicação *web* mantenha um SLA em um ambiente virtualizado. Para se conhecer qual a porcentagem de recursos necessários pela aplicação foi proposta uma técnica chamada de Decomposição de SLA. A Decomposição de SLA baseia-se em submeter a aplicação a um conjunto de testes de desempenho, sendo que em cada teste varia-se a quantidade de usuários simultâneos conectados à aplicação. Desta forma, através da monitoração dos percentuais de recursos físicos consumidos, do tempo de resposta e da quantidade de usuários em cada teste, pode-se mapear qual a percentagem de recursos é necessária para que uma aplicação sobre determinada carga de trabalho cumpra um SLA.

Apesar da Decomposição de SLA possibilitar a um provedor garantir, por exemplo, um determinado tempo de resposta para uma aplicação, ele não pode garantir que manterá o mesmo tempo de resposta, caso a aplicação sofra uma sobrecarga no número de usuários conectados. Em [60] é proposto um algoritmo que busca possibilitar a realocação de recurso, em tempo de execução, com o objetivo de melhorar o desempenho das aplicações hospedadas nas VMs. No entanto, o algoritmo realiza apenas a realocacão de percentagem do processador, não possibilitando desta forma a realocação de memória entre as VMs. Esta limitação acaba impactando no desempenho das VMs e na subutilização da memória.

4.2 Considerações Finais

Conforme foi apresentado neste capítulo, a utilização de SLAs, ao longo do tempo, deixou de ser uma ferramenta utilizada apenas por provedores ISP e passou a ser utilizada nas mais diversas atividades, tais como redes, segurança e máquinas virtuais.

Especificamente na virtualização, a utilização de SLAs tem sido apresentada em alguns trabalhos [59] [5] , sendo que em uma pesquisa recente [60], é proposta a sua utilização como ferramenta que visa definir de maneira precisa o nível de recurso necessário para uma aplicação alcançar um determinado nível de QoS, e também para prover uma melhora no desempenho das VMs através da realocação de porcentagem da CPU.

No entanto, estes trabalhos não possibilitam ao provedor do serviço garantir o SLA, caso a aplicação sofra uma carga de trabalho maior do que a contratada. Outra limitação é quanto à utilização dos recursos livres para atender às VMs. Normalmente, em ambientes virtuais, como o Xen, é necessário que seja atribuída uma porcentagem de processador e de memória para o dom0. Como ele é responsável por executar as operações de I/O dos domU, dependendo do tipo das aplicações hospedadas pelos domU, o dom0 necessita de diferentes percentuais de recursos físicos. Como o tipo das aplicações hospedadas nos domU de uma mesma estrutura tende a ser diferente entre si, e como também são diferentes as necessidades por recursos e os horários que estas aplicações são submetidas a uma carga de trabalho maior, o dom0 utiliza apenas uma parte dos recursos que tem a sua disposição. Assim sendo, a outra parte dos recursos fica livre

no sistema e não pode ser utiliza pelos domU.

Explorar essas limitações e melhorar a utilização dos recursos físicos por parte dos domínios pode resultar em um ganho significativo de desempenho para as aplicações hospedadas nos domU. Com este objetivo, é proposto no próximo capítulo um processo que se utiliza da decomposição de SLAs e da realocação de recursos para distribuir de forma apropriada os recursos entre os domínios e também para garantir um SLA em um ambiente virtualizado.

5 Realocação de Recursos: Proposta e Implementação do Subsistema de Realocação

Este capítulo descreve um processo que busca realocar recursos entre domínios no Xen, conforme as políticas definidas em um SLA, com o objetivo de proporcionar uma melhora em alguns atributos de QoS das aplicações hospedadas nos domínios, por exemplo, o tempo de resposta e também um melhor aproveitamento do *hardware*. Também são apresentados neste capítulo as características de um ambiente virtualizado provido pelo Xen e a implementação de um subsistema para prover a realocação de recursos nesse ambiente.

5.1 Ambiente Virtual Xen com Credit Estático

O funcionamento de um ambiente virtualizado do Xen, configurado com o escalonador Credit em modo estático, pode ser observado na Figura 15, onde se tem quatro VMs (1, 2, 3, 4), sendo que cada uma dessas VMs tem necessidades diferentes de consumo de recursos (CPU e memória), que é representado pela altura de cada VM. A linha tracejada representa a porcentagem de recursos que foi atribuído para cada VM. A área hachurada equivale aos recursos disponíveis, mas não utilizados, no ambiente.

Como se pode observar (Figura 15) a distribuição dos recursos é realizada de modo empírico, não se preocupando em definir precisamente os recursos, conforme as necessidade de cada VM. Por este motivo, pode uma VM necessitar de recursos e não possuir, ou possuir e não os utilizar. Essas duas situações são exemplificadas na Figura 15, onde a VM 2 está subutilizando os recursos a ela disponíveis, enquanto que a necessidade de consumo da VM 3 é maior do que o limite à sua disposição, não sendo possível atender a toda sua demanda.

Para melhorar o desempenho global do Xen e garantir determinado nível de QoS às aplicações hospedadas nesses ambientes, espera-se que os recursos dos mesmos possam ser adequadamente distribuídos, conforme políticas de SLA definidas.

Este modelo ideal está representado na Figura 16, onde é possível notar, pela linha tracejada, que os recursos computacionais foram distribuídos proporcionalmente à demanda de cada VM, sem excesso ou escassez, respeitando a disponibilidade global do ambiente.

A utilização das políticas do SLA e do Credit em modo estático, para se definir e manter o nível de recurso que é necessário para um ambiente virtualizado, é suficiente para se manter um SLA em uma situação normal de processamento. Entretanto, a utilização dessa técnica não

possibilita uma ótima utilização do *hardware*, já que os recursos físicos não utilizados, não podem ser realocados entre os domínios. Outra limitação imposta, é que caso um domínio sofra uma carga de trabalho maior do que a prevista no SLA, o que irá gerar uma necessidade por mais recursos, e existam recursos não utilizados na estrutura, o escalonador não irá realocar esses recursos para atender às necessidades do domínio sobrecarregado.

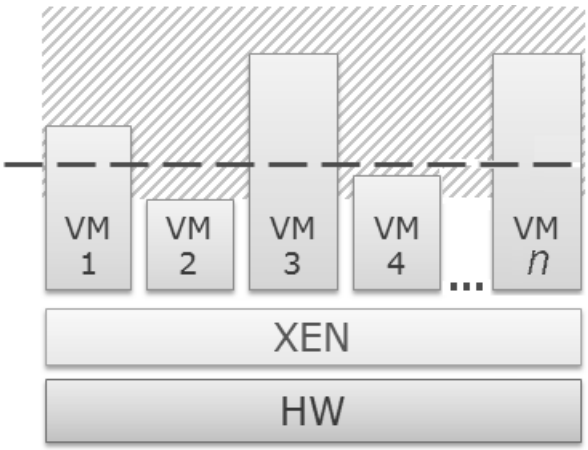

Figura 15 – Alocação de recursos no Xen

Com o objetivo de superar estas limitações, é proposto um processo que busca definir de maneira adequada o nível de recurso necessário para um domínio, com base em seu SLA e também possibilitar ao provedor do serviço manter a qualidade do serviço em algumas situações de quebra do SLA por parte do cliente.

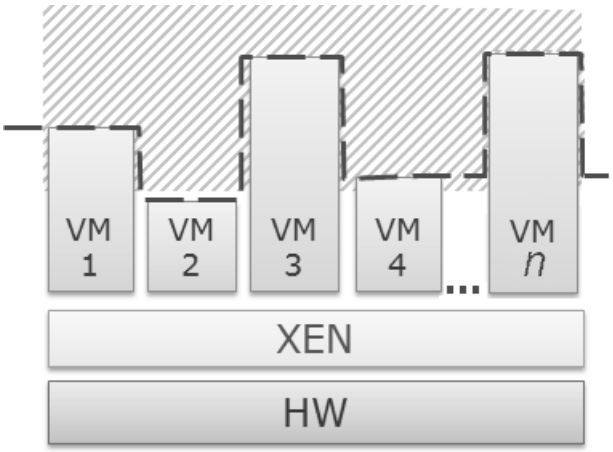

Figura 16 – Realocação de recursos no Xen

5.2 Processo Proposto

Para prover a realocação de recursos no Xen é proposto um processo que é composto por 5 passos, dividido em duas etapas (Figura 17). A primeira etapa consiste na decomposição do SLA através da execução de testes de desempenho sobre a estrutura virtualizada. No primeiro passo dessa etapa, realiza-se a execução de *benchmark* sobre a aplicação hospedada no ambiente virtualizado. Os resultados obtidos nos testes são utilizados para se mapear as métricas de baixo nível (processador e memória), sob diferentes cargas de trabalho, provendo dados de configuração e de desempenho desse ambiente. Na segunda etapa, estes dados são utilizados em conjunto com os SLAs das VMs para que, dada uma configuração vigente de uma máquina Xen e sendo seguidas as políticas de SLA, o subsistema identifique qual a melhor maneira de reconfigurar os seus recursos.

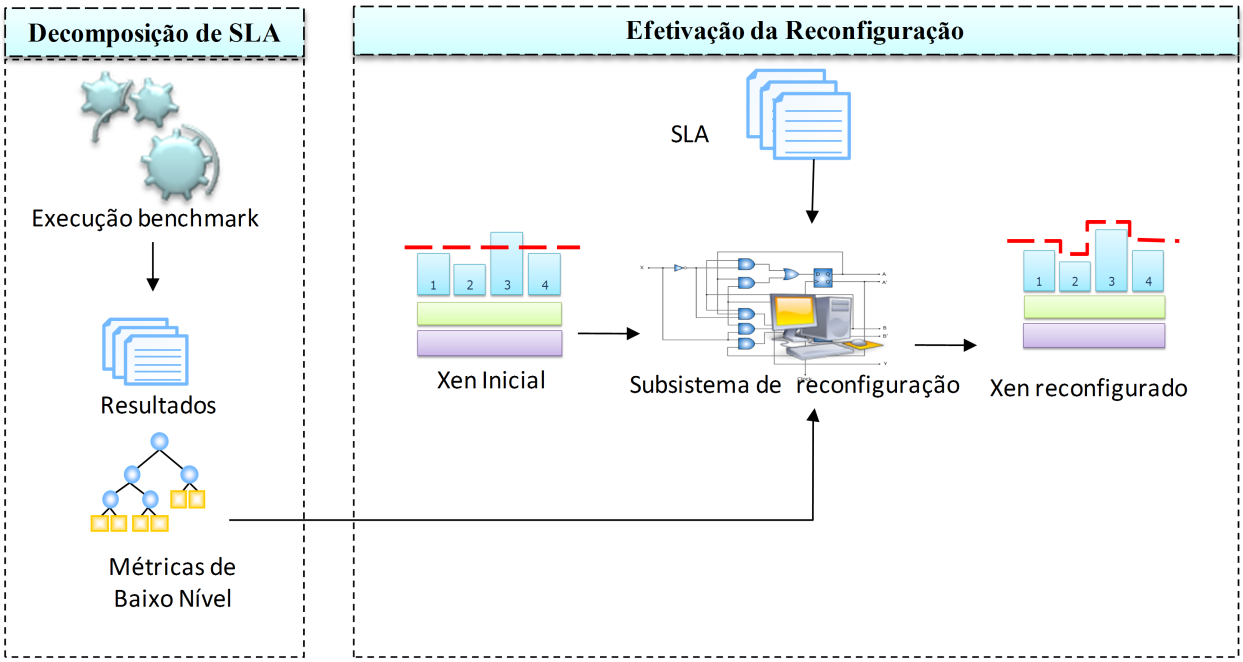

Figura 17 – Processo proposto para a realocação de recursos

5.2.1 Decomposição de SLA

Para uma definição precisa dos níveis de recursos que uma aplicação necessita para atender às especificações de um SLA, é necessário decompor as métricas de alto nível em métricas de baixo nível. Ou seja, uma métrica como tempo de resposta de uma aplicação é decomposta em

métricas de baixo nível, com uso dos seguintes atributos: utilização do processador (up) e memória disponível (md). Para cada um desses são definidos valores mínimos a serem atingidos, onde $vmup$ e $vmmd$ representam, respectivamente, tais valores para os atributos citados.

Para atingirmos um indicador de nível de serviço (SLI) é necessário que, por exemplo,$up > vmup$ e $md > vmmd$. Para conhecer estes níveis, precisa-se submeter à aplicação a um conjunto de testes através da execução de *benchmarks* executando sob diversas configurações, com o objetivo de definir a quantidade de recursos do sistema que cada componente monitorado pelas métricas de baixo nível consome. Deste modo, é possível determinar o volume de recursos consumidos por cada componente do sistema para suprir de maneira precisa cada SLI de um SLA.

Apesar de essa técnica possibilitar ao administrador de uma estrutura virtualizada conhecer o nível de recursos que é necessário para que um determinado SLI seja alcançado, no entanto, o processo de execução dos *benchmarks* é um processo lento, pois é necessário que um grande número de casos de testes seja executado, para cobrir todos os possíveis SLAs da estrutura.

Com o objetivo de automatizar e tornar viável, em relação ao tempo de duração, o processo de criação e execução dos testes, foi desenvolvida uma ferramenta de geração de casos de teste baseado em modelos UML (*Unified Modeling Language*) [61]. O desenvolvimento dessa ferramenta é resultado de esforços conjuntos, no âmbito do centro de pesquisa HP/PUCRS, com outros dois trabalhos de mestrado [62] [63], que buscam desenvolver uma linha de produto de teste de *software* baseado em modelos. Assim sendo, é reutilizado, no desenvolvimento da ferramenta de teste de desempenho baseado em modelos UML, a parte do código comum as demais ferramentas de teste baseado em modelos.

O primeiro passo nessa abordagem é se utilizar dos diagramas UML da aplicação para extrair informações sobre o comportamento da aplicação sobre teste. Neste caso, as informações necessárias serão extraídas especificamente dos diagramas de casos de uso e dos diagramas de atividades, sendo que, cada caso de uso terá seu diagrama de atividades correspondente. No entanto, algumas informações necessárias para geração automatizada dos casos de teste de desempenho não são encontradas nos diagrama UML da aplicação, e devem ser inseridas manualmente. Estas informações, relevantes para se conhecer o comportamento do sistema, são inseridas através da utilização de *tags*.

Nos diagramas de casos de uso é necessário que sejam inseridas duas informações, através do uso das *tags*. A primeira, que se chama “PApopulation”, corresponde ao número de usuários que irão acessar o sistema durante o teste, e a segunda *tag*, chamada de “PAprob”, corresponde à probabilidade de execução de cada caso de uso. A inserção dessas informações é necessária, para que se conheça o número de usuários que serão simulados em cada teste, e a probabilidade de determinado caminho ser executado em cada caso de teste. A utilização da *tag* PAprob é especialmente importante, quando testamos aplicações como uma página de comércio eletrônico, pois determinados *links* do site são mais acessados que outros, sendo necessário simular este comportamento durante o teste de desempenho da aplicação.

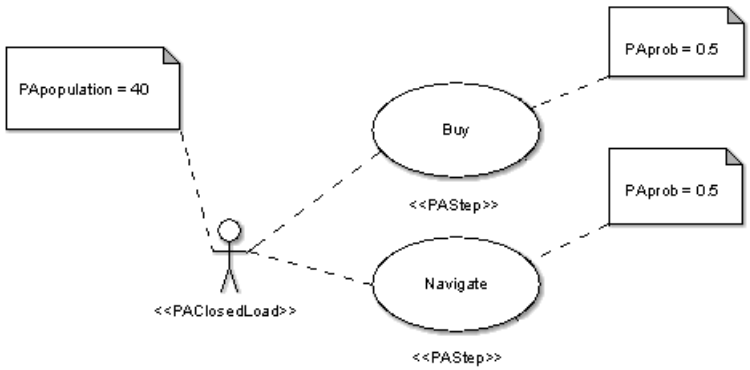

Figura 18 – Diagrama de casos de uso da aplicação *web* do TPC-W [2]

Nos diagramas de atividades é necessário que sejam inseridas *tags* em cada atividade do diagrama, com o objetivo de simular o tempo que cada usuário gasta para executar a atividade.

Neste caso é inserida uma *tag* com um valor aleatório, que corresponde ao *think time* de cada atividade.

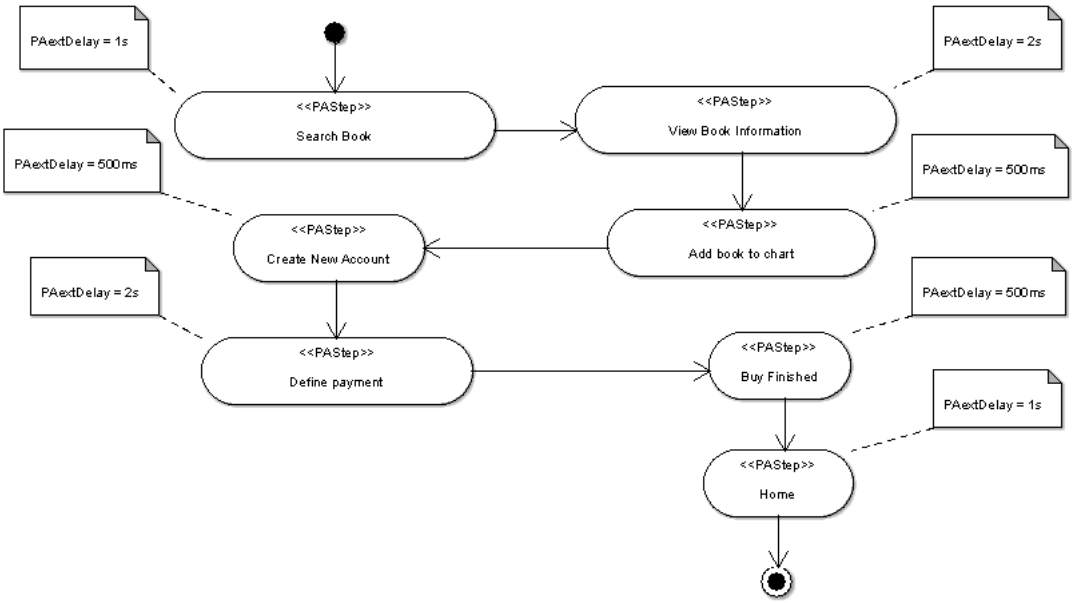

Figura 19 – Diagrama de atividades da aplicação *web* do TPC-W [2]

Após a modelagem é necessário extrair dos modelos, as informações necessárias para a geração dos *scripts*. Para extrair essas informações e padronizar o formato de entrada dos dados, foi utilizado um *parser* XML, que extrai e padroniza o formato de entrada, para a ferramenta de geração dos *scripts*, com as demais ferramentas da linha.

Com os dados formatados no novo padrão, os mesmos são inseridos em uma ferramenta que gera os possíveis caminhos e as interações do usuário dentro do sistema, gerando desta forma os casos de teste do sistema. A ferramenta de geração dos casos de teste é utilizada por todas as demais ferramentas de geração de casos/scripts da linha de produto, sendo seu desenvolvimento, tema de outro trabalho de mestrado [62].

Os casos de teste gerados pela ferramenta são salvos em um arquivo XML, e desta forma, não podem ser executados diretamente por uma ferramenta de teste de desempenho. Por este motivo foi desenvolvido um *parser* que captura as informações dos arquivos XML e as converte em *scripts* de teste da ferramenta de teste de desempenho Jmeter [64]. Após a geração dos *scripts*, os testes são executados pela ferramenta sobre a aplicação alvo, sendo os resultados utilizados para realizar a decomposição de SLA.

Conseqüentemente, com a utilização de teste baseado em modelos para automatizar o processo de decomposição de SLA, o processo descrito na Figura 17 foi alterado para representar os passos adicionados (conforme Figura 20). Entretanto, a utilização de teste baseado em modelos não altera o processo de decomposição de SLA, mas apenas o automatiza, permitindo assim resultados mais precisos em menos tempo.

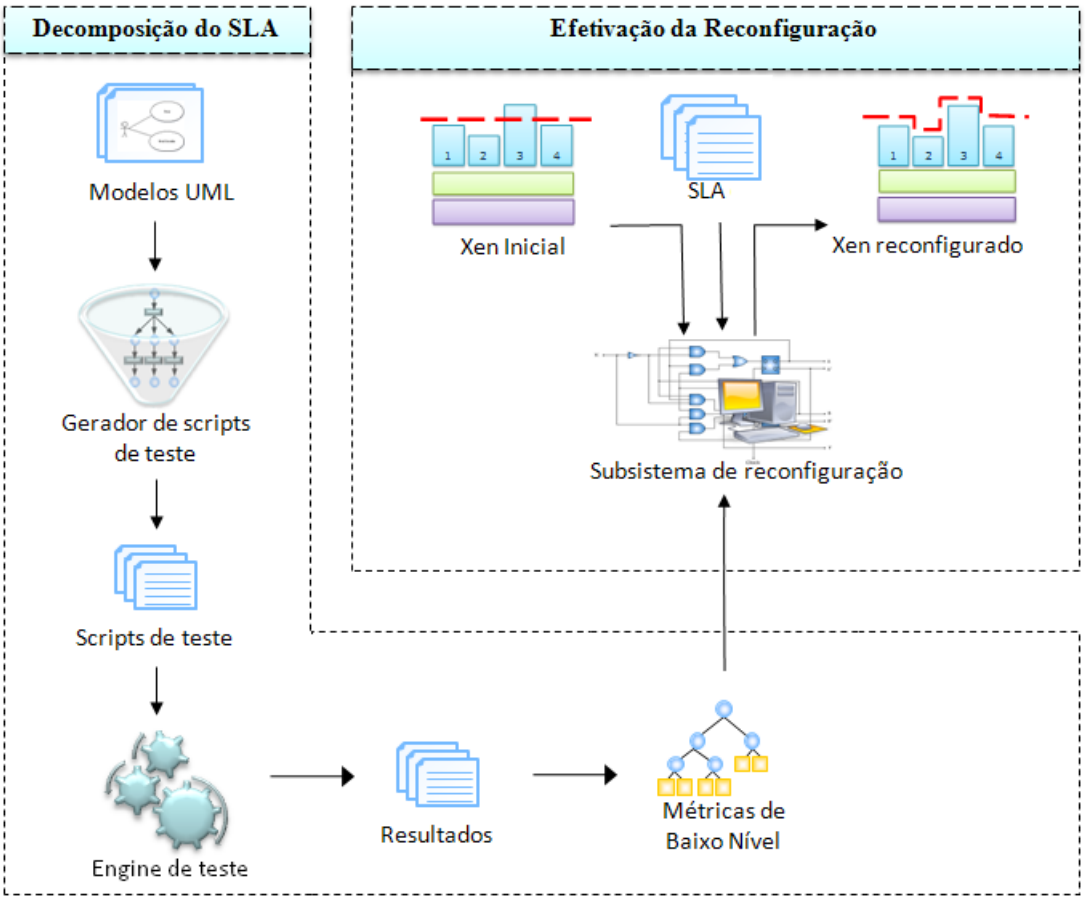

Figura 20 – Processo proposto para a realocação de recursos

A ferramenta de geração de casos de teste de desempenho baseado em modelos UML se utiliza dos modelos UML da aplicação, que será submetida aos testes, para gerar de forma automatizada os casos de teste e os *scripts* de teste, que são inseridos em uma ferramenta de teste de desempenho. Em seguida, a ferramenta realiza os testes no ambiente virtualizado, sendo os resultados desses utilizados para se mapear as métricas de baixo nível (processador e memória) sob diferentes cargas de trabalho, provendo dados de configuração e de desempenho

desse ambiente. Na segunda etapa, de forma análoga ao processo anterior, estes dados são utilizados em conjunto com os SLAs das VMs para que, dada uma configuração vigente de uma máquina Xen e sendo seguidas as políticas de SLA, o subsistema identifique qual a melhor maneira de reconfigurar os seus recursos.

A utilização da decomposição permite apenas a definição estática dos recursos necessários para que uma aplicação cumpra um SLA. Caso uma aplicação necessite de mais recursos do que aqueles definidos em seu SLA, mesmo existindo recursos não utilizados por outras VMs, a aplicação poderá apresentar uma degradação na QoS.

Com o objetivo de superar esta limitação, é proposto um processo de reconfiguração, que se utiliza da decomposição de SLA e da realocação de recursos, para prover a utilização de SLAs em ambientes virtualizados (VMs) que compartilhem uma mesma estrutura física.

5.2.2 Realocação de Recursos

A realocação consiste em mover os recursos que não foram atribuidos às VMs (linhas transversais da Figura 21), e que ficam alocados para o dom0, para a VM que esteja utilizando todos os recursos atribuídos a ela, (simbolizado pelas barras verticais na Figura 21) através da decomposição de SLA e, que, ainda assim, necessite de mais recursos (área escura da VM 3 da Figura 21). No entanto a quantidade de recursos que o subsistema pode realocar para uma VM nunca pode provocar a quebra do SLA das demais VMs que compartilham a mesma estrutura. Assim sendo, mesmo que uma VM possua recursos não utilizados, dentro dos limites definidos pela decomposição SLA, estes recursos não devem ser realocados (área quadriculada da VM 1, Figura 21).

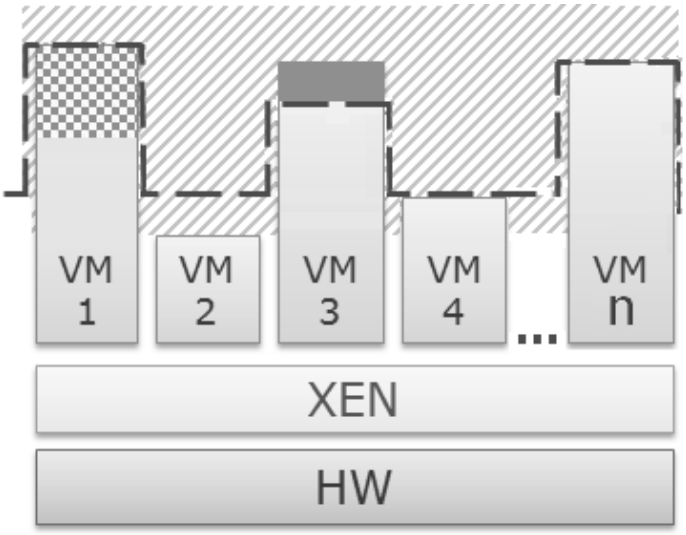

Figura 21 – Alocação dos recursos no Xen com Credit

Ou seja, o subsistema executa a realocação apenas dos recursos que não foram atribuídos às

VMs, e que não estão sendo utilizados pelo dom0 (área hachurada, Figura 22).

Desta forma, o subsistema garante que o percentual de recursos utilizados na realocação não causará uma queda de desempenho nas aplicações hospedadas nas VMs, pois este percentual não é "visto" pelos domU. Outro fato relevante na realocação é que o percentual de recursos livres, que pode ser utilizados para realocação, não é fixo, sendo que ele depende da quantidade de recursos que a estrutura física dispõe, do tipo de aplicação, da carga de trabalho, entre outros diversos fatores. Desta forma, o provedor do serviço pode garantir que cumpre o SLA, mas não pode garantir que sempre atenderá o contrato durante uma sobrecarga, por menor que ela seja.

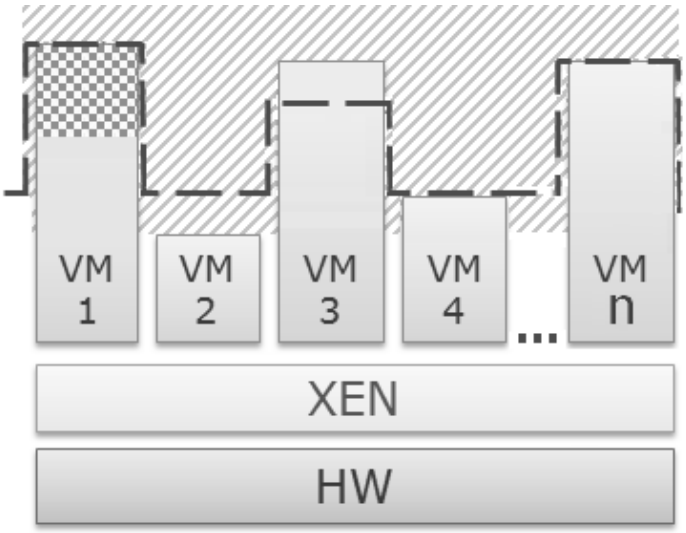

Figura 22 – Realocação de recursos com o subsistema

Com o objetivo de permitir a realocação de recursos (CAP e memória) entre os domínios de um ambiente virtualizado provido pelo Xen, foi implementado um subsistema que possibilita ao escalonador Credit, quando em modo estático, realizar a realocação de recursos.

Para realizar o processo de realocação descrito, o subsistema de realocação se utiliza, num primeiro momento, dos resultados do processo de decomposição de SLA. A partir dos resultados obtidos com a decomposição, o subsistema de realocação de recursos monta uma estrutura de dados onde armazena os resultados da decomposição. Esses dados, em conjunto com o SLA, serão utilizados pelo subsistema para configurar automaticamente o nível de recurso que será atribuído a cada domínio, no momento da sua inicialização, ou para se definir um novo nível de recurso para um domínio, caso o seu respectivo SLA seja alterado após a sua inicialização.

Após a inicialização dos domínios e da configuração do nível de recursos, através da decomposição, o subsistema inicia a monitoração dos domínios, buscando identificar uma necessidade maior por recursos adicionais por parte dos mesmos (Figura 23, linha 15). Ele monitora e analisa continuamente, a cada 1 segundo, o nível de recursos utilizados por cada um dos domínios que compartilham a estrutura. Caso seja detectado, que um domínio necessita de um nível de recurso maior do que o limite que lhe foi atribuído (Figura 23,linha 7), e exista a disponibilidade de recursos (Figura 23, linha 8), o subsistema irá realocar esses recursos para atender às

necessidades do domínio (Figura 23, linha 9). Após os recursos serem alocados para o domínio, o algoritmo altera o valor da *flag* "recursos livres", para sinalizar que os recursos estão sendo utilizados e impedir que outra VM tente realocar os mesmos recursos. Em seguida, ele atualiza a posição da VM em uma lista circular, onde estão ordenadas as VM por ordem de utilização dos recursos, inserindo a mesma na última posição.

```
1  VMs[TOT_VMS]
2  recursosLivres = falso
3  IND_VM_USANDO_RECURSO
4
5  Principal()
6      Para cada VMs[i]
7          Se (UtilizacaoRecursos( VMs[i] ) > SLA)
8              Se (recursosLivres = falso)
9                  RealocaRecursosLivresPara(VMs[i])
10                 recursosLivres = verdadeiro
11                 MoveParaFimDaLista(VMs[i])
12                 IND_VM_USANDO_RECURSO = i
13             fimSe
14         fimSe
15         Monitora(IND_VM_USANDO_RECURSO)
16         Desaloca(IND_VM_USANDO_RECURSO)
17     fimPara
18 fimPrincipal
```

Figura 23 – Algoritmo do subsistema de realocação

Na seqüência , é chamado o procedimento Monitora, cuja função é realizar continuamente a monitoração dos domínios e imprimir as métricas monitoradas (Figura 24, linha 3).

```
1 Procedimento Monitora(x)
2     Se (UtilizacaoRecursos( VMs[x] ) > 95% )
3         imprimir "Imprimir metricas"
4 fimProcedimento
```

Figura 24 – Algoritmo do subsistema de realocação - Módulo Monitora

Após a monitoração, é chamado o procedimento Desaloca, cuja função é detectar se o nível de utilização dos recursos retornou a aquele definido através da decomposição de SLA (Figura 25, linha 2). Quando esta situação ocorre, os recursos adicionais alocados serão retirados (Figura 25, linha 2) e a *flag* "recursos livres" é alterada, tornando os recursos novamente disponíveis para serem realocados para os demais domínios. No entanto, o tempo de utilização dos recursos pela VM é garantido pelo escalonador, que não será inferior a 15 segundos. Esta medida busca evitar situações em que o custo de realocar/desalocar recursos é maior que o benefício apresentado pelos recursos adicionais [10].

```
1 Procedimento Desaloca(x)
2       Se (UtilizacaoRecursos( VMs[x] ) <= SLA)
3           DesalocaRecursosLivres(VMs[x])
4           recursosLivres = falso
5  fimProcedimento
```

Figura 25 – Algoritmo do subsistema de realocação - Módulo Desaloca

5.3 Considerações Finais

Cada vez mais as empresas estão se utilizando da virtualização para consolidar diversos servidores físicos em uma estrutura virtualizada, em busca de uma redução nos custo de aquisição, manutenção e gerenciamento do seu parque de máquinas.

Entretanto, a migração das aplicações hospedadas em servidores físicos para um ambiente virtualizado apresenta alguns desafios. Um desses desafios é estimar precisamente a quantidade de recursos que é necessária para essa aplicação manter o mesmo nível de serviço no ambiente virtualizado, já que um valor estimado inferior ao ideal causaria uma degradação da QoS da aplicação e uma estimativa superestimada provocaria uma subutilização do *hardware*. Para prover um indicador que permita ao administrador do ambiente virtualizado migrar uma aplicação, que possui um SLA, de um ambiente físico para um ambiente virtualizado sem comprometer o comportamento da aplicação e a quebra do SLA, utilizamos a decomposição de SLA.

Outro desafio é otimizar a utilização do *hardware*, pois, como as VMs que compartilham uma mesma estrutura física possuem diferentes níveis de recurso, a soma desses não constitui 100% dos recursos que a estrutura disponibiliza. Assim sendo, em ambientes Xen configurados com o escalonador Credit em modo estático, na maioria dos casos existirão recursos que não foram atribuídos a nenhuma VM. Neste caso, os recursos que não estão sendo utilizados ficam a disposição do dom0, que pode utilizá-los ou não. Normalmente, o dom0 não utiliza esses recursos, já que ele possui a sua disposição os seus próprios recursos, ficam os demais ociosos. A implementação do subsistema busca otimizar a utilização do *hardware*, utilizando estes recursos para atender um domínio que possua uma sobrecarga e necessite de mais recursos.

No próximo capítulo, será testado o desempenho do processo de realocação descrito, e comparado com os escalonadores disponíveis no Xen. Como estudo de uso, será utilizado um servidor de comércio eletrônico em cada VM da estrutura, que sofrem uma carga de trabalho variada.

6 Resultados

Este capítulo descreve os testes realizados para avaliar o desempenho do subsistema de realocação proposto neste trabalho. A avaliação de desempenho foi realizada através da utilização de *benchmarks*, executando uma grande variedade de casos de teste. Os resultados obtidos com esses testes são utilizados para se comparar o desempenho de uma aplicação, quando se utiliza dos escalonadores nativos do Xen, como quando se utilizam do conjunto, escalonador Credit e o subsistema de realocação de recurso.

6.1 Ambiente de Teste

Para validar o ganho de desempenho proporcionado pela proposta descrita neste trabalho, utilizamos um conjunto de três aplicações de *software* livre, executando em um ambiente virtualizado. Esse ambiente é composto de quatro máquinas virtuais idênticas, que buscam simular um *data center* virtualizado. Cada VM hospeda um servidor *web* Apache [65], um servidor Tomcat 5.5 [66] e um servidor de banco de dados MySQL 5.0 [67]. Para realizar os *workloads* sobre as aplicações hospedadas no ambiente virtualizado, utilizamos duas ferramentas de teste de desempenho, o Apache Jmeter e o TPC-W (*Transactions Performance Council-Web*) [2].

O Jmeter é uma ferramenta para teste de desempenho e de *stress* de aplicações *web*, desenvolvida dentro do projeto Apache. Entre outras funcionalidades, ele possibilita enviar requisições HTTP para uma aplicação, com o objetivo de simular o acesso simultâneo de múltiplos usuários ao sistema.

Neste trabalho o Jmeter é utilizado, em um primeiro momento, como ferramenta de teste de desempenho, para executar os testes necessários para a decomposição de SLA. Desta forma, o Jmeter se utiliza dos *scripts* gerados pela ferramenta de geração de testes baseados em modelos, e esses *scripts* são carregados na ferramenta, que os executa sobre a aplicação, sendo os resultados obtidos utilizados para mapear os percentuais de recursos físicos (processador e memória) que são necessários para que uma VM mantenha determinado SLA.

Também foram realizados diversos testes com o Jmeter, com o objetivo de se comparar o desempenho do subsistema proposto com os demais escalonadores do Xen, já que a execução dos mesmos, sem utilizar um processo de geração e execução automatizado, é inviável em relação ao tempo de execução.

Para simular de maneira realista o acesso simultâneo de diversos usuários ao ambiente virtu-

alizado, e desta forma avaliar o desempenho do subsistema desenvolvido frente aos escalonadores do Xen, também foram realizados diversos testes com a ferramenta de teste de desempenho TPC-W. O TPC-W é uma ferramenta de *benchmark* para ambientes *web*, que busca simular de maneira realista o comportamento de diversos usuários simultâneos no sistema sob teste.

O TPC-W é um *benchmark* padrão da indústria para aplicações de comércio eletrônico, que simula o comportamento de um *site* de comércio eletrônico, similar a Amazon [68], sendo que ele permite que sejam realizadas algumas séries de configurações, com o objetivo de aproximar o máximo possível, o comportamento do *benchmark* ao ambiente a ser utilizado.

Para isso, são realizadas múltiplas interações para simular as atividades de uma loja *on-line*, onde cada interação é sujeita a um determinado tempo de resposta. O tamanho da loja a ser emulada é definido com base na quantidade de produtos que será utilizado no banco de dados, que, neste caso, são livros, podendo o tamanho do banco de dados variar de 1.000 a 10.000.000 obras. Estas obras são mantidas em um catálogo de livros, que podem ser localizados pelos usuários, sendo associada a cada um destes livros, uma imagem de 5 KB de tamanho [69]. No TPC-W, um usuário é emulado através de um emulador remoto de *browser* (*RBE-Remote Browser Emulator*), que simula o mesmo tráfego da rede do HTTP que seria gerado por um cliente real ao usar um *browser* para navegar, realizar buscas e compras de livros no *site*.

A principal métrica monitorada pelo TPC-W é a taxa de interações com o servidor *web* por segundo *(WIPS - Web Interactions Per Second)*. O WIPS é o número de interações com o servidor *web* por segundo, sendo que seu número define qual o limite de interações que o sistema sob teste suporta *(SUT - System Under Test)*. O SUT pode ser um servidor, ou um conjunto de servidores que provêem a solução de comércio eletrônico do TPC-W. O TPC-W possui ainda dois tipos de interações que são medidas para fornecer duas métricas secundárias que são o WIPSo e WIPSb. O WIPSb (*Web Interactions Per Second - browsing*) é uma simulação de um *web site* onde a maioria das solicitações ao servidor são de páginas *web*, e poucas de acesso a banco de dados. O WIPSo (*Web Interactions Per Second - ordering*) é uma simulação onde a maioria das interações é de acesso a banco de dados. Neste trabalho foi monitorado apenas o número de interações por segundo (WIPS), pois ele é utilizado para se obter o tempo de resposta da aplicação. Isso porque esta métrica não é monitorada diretamente pelo TPC-W, mas sim obtida através de um fórmula algébrica, definida pela Lei do Tempo de Resposta [70], onde o tempo de resposta (**R**) é igual ao número de *Emulate Browser* (**M**) dividido pelo número de interações com o servidor *web* (**WIPS**) subtraído do *think time* (**Z**) [71].

$$R = \frac{M}{WIPS} - Z$$

No desenvolvimento desse trabalho, o TPC-W foi configurado com o banco de dados no tamanho de 10.000 itens e 200.000 usuários, sendo que o intervalo que cada cliente emulado aguarda entre o final da última interação executada, e antes de iniciar a interação seguinte (*think time*) foi configurado como um valor randômico entre 1 e 7 segundos. Para permitir um número

de conexões simultâneas maiores que o valor padrão aceito pelo Tomcat e pelo Mysql, foi definido 100 usuários como o número máximo de conexões simultâneas nos mesmos.

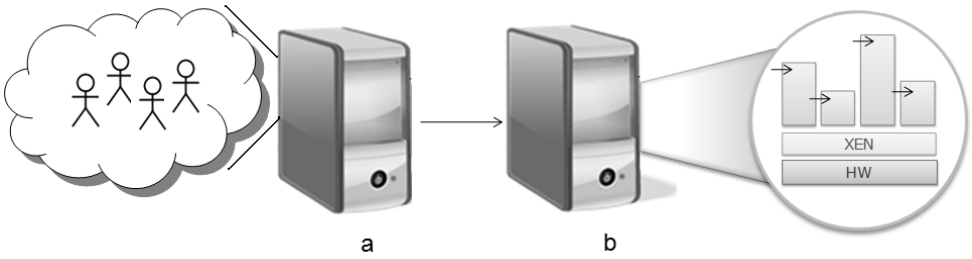

Figura 26 – Estrutura utilizada

O ambiente de teste busca simular um *data center* virtualizado, com múltiplas aplicações compartilhando um conjunto comum de recursos, sendo composto de duas máquinas físicas. A primeira máquina é utilizada como hospedeiro das VMs com os serviços, sendo que esta estrutura hospeda quatro Máquinas Virtuais Xen (Figura 26 (b)). A outra máquina (Figura 26 (a)) é utilizada como gerador de *workload* dos clientes (*RBE - Remote Browsers Emulator*) e pelo Jmeter. A ligação entre os clientes (*RBEs - Remote Browser Emulate*) e o servidor é realizada através de uma rede rápida Gigabit Ethernet. A estrutura descrita é hospedada em um HP dx5150 MT com um processador Athlon(tm) 64 X2 Dual Core, 3.8 GHz com 2.0 GB de memória RAM e com um disco rígido IDE com capacidade de 160 GB. O Sistema Operacional utilizado foi o Ubuntu 7.04 server i386, *Kernel* 2.6.19-4-server.

6.2 Casos de Teste e Resultados

Com base no ambiente de teste descrito na Seção 6.1, foram implementados diversos casos de teste que buscam retratar diferentes cenários. Os cenários exemplificam situações onde um número variável de usuários acessa diversos conjuntos de aplicações que são hospedados em distintas VMs do mesmo ambiente virtualizado.

No entanto, antes de se realizar os testes para avaliar o desempenho do subsistema, foram definidos e implementados os casos de testes baseados em modelos a serem utilizados para a decomposição de SLA. Só que para realizar a geração automática dos casos de teste, é necessário que se conheça o número máximo de usuários que cada VM da estrutura suporta. Caso não sejam encontradas e testadas todas as possíveis cargas de usuários que a estrutura suporta, pode ocorrer que o SLA de uma aplicação seja alterado para um determinado número de usuários, e não existam as informações da decomposição para o mesmo, já que não foram realizados os testes para fornecer esses dados. Com o objetivo de conhecer o limite máximo de usuários que uma VM da estrutura pode atender, e seus respectivos mapeamentos de recursos, foram configurados todos os recursos disponibilizados por essa estrutura (CAP e memória) para uma VM

específica. Em seguida são realizados testes de desempenho, com um número de usuários crescente sobre o sistema em teste (SUT), até que o mesmo "quebre". Desse modo, esta abordagem garante que todos os possíveis SLAs que essa estrutura possa vir a cumprir serão cobertos pela execução dos testes, e desta forma, possuirá os dados necessários para a sua decomposição.

Após o subsistema ser alimentado com os dados da decomposição, foram realizados alguns casos de teste, se utilizando do Jmeter e do TPC-W como ferramentas de teste de desempenho.

6.2.1 Casos de teste e resultados com o TPC-W

O TPC-W foi utilizado para simular o cenário de um *data center* que hospeda diversas VMs em uma mesma estrutura física, sendo que uma ou mais dessas VMs sofrem uma sobrecarga no seu número de usuários. Para se comparar o desempenho do subsistema, com três diferentes níveis de recursos, com o escalonador Credit em modo estático, foram realizadas 120 execuções do *benchmark*, totalizando 240 horas de teste.

Para simular a situação de sobrecarga, foi definido em um primeiro momento um SLA que foi aplicado a todas as VMs da estrutura. Esse SLA possui um SLO que define que com uma carga de até 25 usuários simultâneos no sistema o tempo de resposta da aplicação deve ser inferior a 1 segundo. A partir deste ponto, foi realizada a decomposição e configuração dos recursos que são disponibilizados de maneira uniforme para todas as VMs da estrutura. Através da decomposição, o subsistema definiu os limites de recursos alocados para cada VM, sendo 20% do processador (CAP) e 400 MB de memória RAM. Entretanto, a VM 1 recebeu uma carga de 35 usuários, sendo a mesma superior a aquela utilizada para realizar a decomposição. Esta adição de usuários busca simular uma situação onde uma VM da estrutura recebe uma carga de usuários maior do que a definida no SLA.

Como se pode observar na Figura 27, quando o Xen está utilizando o escalonador Credit, o tempo de resposta das VMs 2, 3 e 4 se encontram dentro dos limites definidos em seus SLAs, no entanto a VM 1 apresenta um tempo de resposta que ultrapassou por larga margem o valor estipulado no SLA. Este resultado se deve ao número de usuários adicionais aos definidos no SLA, pois embora a estrutura que hospeda as VMs possua 20% dos seus recursos livres, o escalonador do Xen (Credit estático) não é capaz de realizar a realocação dos recursos não utilizados em ambientes com SLAs. Com a utilização do subsistema de realocação, esses recursos tornam-se disponíveis para qualquer uma das VMs da estrutura, possibilitando suprir uma demanda maior por recursos pelas mesmas. Comparando o tempo de resposta da aplicação, quando executada apenas com o escalonador do Credit, com o tempo de resposta de quando se utiliza o subsistema, nota-se uma acentuada redução do mesmo. Porém quando o mesmo teste foi realizado, só que se alterando a quantidade de recursos livres para 10%, os resultados mostraram que, quando comparado com o escalonador Credit estático, apesar do tempo de resposta da VM com sobrecarga apresentar redução, essa redução foi inferior a obtida no teste com

20% de recurso livre, mas foi suficiente para manter o SLA. Outro resultado relevante obtido, é que a utilização do subsistema, com 20% ou 10% de recursos livres, possibilitou um ganho de desempenho global para as VMs da estrutura, sendo que esse ganho é proporcionado pelo fato do subsistema realizar a realocação dos recursos para a VM que necessitar primeiro, neste caso a VM 1.

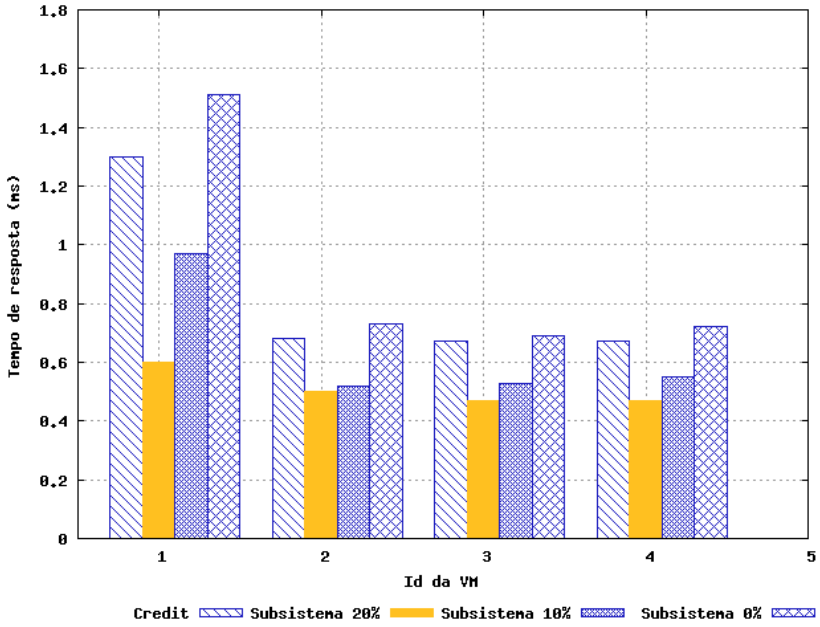

Figura 27 – Sobrecarga em 1 VM

No entanto, se o processamento da VM 1 retornar aos níveis descritos em seu SLA, o subsistema retira os recursos adicionais e os realoca para próxima VM que apresentar um pico de processamento. Deste modo, os recursos não são monopolizados pela VM com sobrecarga, e tornam-se disponíveis para todas as VMs da estrutura. Contudo, apesar da utilização do subsistema ter se mostrado benéfica para o desempenho das aplicações hospedadas nas VMs quando o mesmo possuía recursos livres para realocar, este bom desempenho não se repete quando não existem recursos livres para serem realocados, sendo o desempenho do subsistema inferior ao escalonador Credit estático. Este fato ocorre, devido ao custo computacional provocado pela execução do subsistema, que realiza chamadas constantes aos métodos da biblioteca Xenstat para monitorar o nível de recursos livres, e também por tentar em algumas situações utilizar a ferramenta *xm* para realocar os recursos.

No segundo cenário de teste, de modo análogo ao anterior, as aplicações hospedadas na VM 3 e na VM 4 foram submetidas a uma carga que simula o acesso simultâneo de 25 usuários, e as VMs 01 e 02 tiveram o número de usuários simultâneos alterado para 35. O acréscimo de usuários busca retratar um cenário onde o SLA das duas VMs é quebrado temporariamente por parte do cliente e as VMs disputam os recursos livres disponibilizados pelo subsistema.

Como resultado (Figura 28), podemos observar que as VM 1 e 2 apresentaram um tempo de resposta maior do que o valor definido nos SLOs dos respectivos SLAs, sendo que quando

foram realizados os mesmos testes, mas utilizando o subsistema de realocação com 20% e 10% de recursos livros, foi obtida uma significativa redução do tempo de resposta das VM 1 e 2, quando comparado com o Credit estático. Entretanto nesse cenário houve uma redução do ganho nas demais VMs (VM 3, Figura 28) ou até mesmo não apresentando ganho (VM 4, Figura 28). Este resultado é devido às necessidades de processamento das VM 1 e 2 serem maiores. Deste modo, elas se utilizaram por um intervalo de tempo maior, dos recursos realocados, em detrimento das outras VMs que possuíam uma carga menor. Apesar do bom desempenho do escalonador com 20% e 10% de recursos livres, o desempenho do mesmo sem recursos para realocar (0%), assim como no teste anterior, foi ligeiramente inferior ao do escalonador Credit estático.

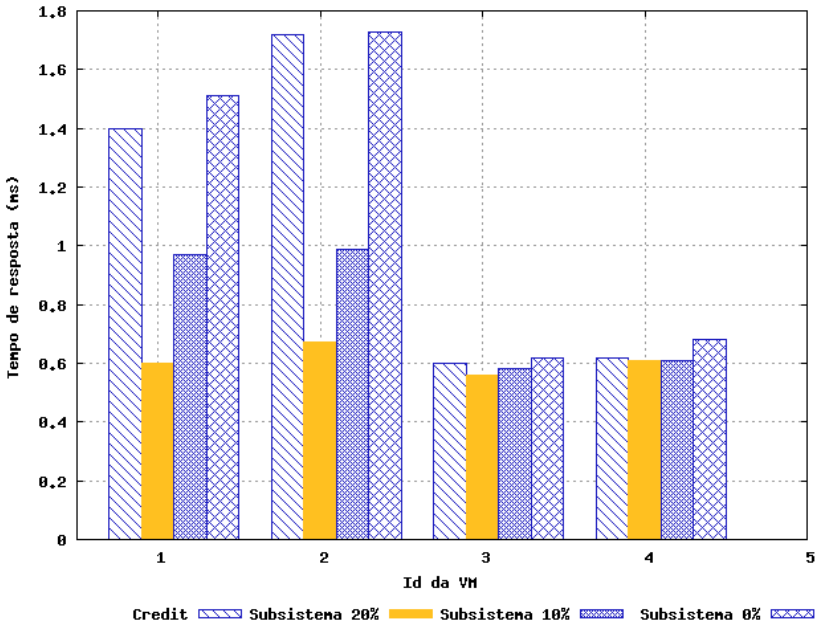

Figura 28 – Sobrecarga em 2 VMs

No terceiro cenário de teste, busca-se retratar uma situação onde três VMs sofrem uma carga de usuários, maior do que aquela definida no SLA. Neste caso, as VMs 1, 2 e 3 sofrem uma carga de 35 usuários e a VM 4 uma carga de 25 usuários (carga prevista no SLA). Como se pode observar na Figura 29 a utilização do subsistema proporcionou uma melhora no desempenho das VMs com sobrecarga quando comparado com o escalonador Credit configurado em modo estático, no entanto, devido ao fato de existirem três VMs disputando os recursos livres, o subsistema acabou não beneficiando a VM sem sobrecarga.

Conforme os resultados apresentados nesta seção, a utilização do processo de realocação de recursos e do subsistema, proporciona uma melhora significativa no desempenho das aplicações hospedadas nos ambientes virtualizados (VMs), quando comparado com o escalonador Credit configurado em modo estático, principalmente em situações onde apenas poucas VMs apresentem sobrecarga. Comparando-se os gráficos das Figuras 27, 28 e 29, pode-se observar que a eficiência do subsistema tende a diminuir quando o número de VMs com sobrecarga aumenta.

Esta situação é motivada pelo fato do subsistema compartilhar um conjunto comum de recursos entre as diversas VMs, sendo que quanto maior o número de VMs, menor o tempo de utilização dos recursos. Outro fato mandatório para esta redução de desempenho, é que quando existem diversas VMs com sobrecarga o subsistema necessita realizar um número maior de operações de realocar/desalocar recursos, o que gera certo custo computacional e conseqüentemente uma degradação no desempenho das aplicações. Os resultados também demonstram que a utilização do subsistema, em situações onde não existam recursos livres para serem realocados, causa degradação no desempenho, motivado pelo custo computacional da execução do subsistema e, principalmente, da utilização do *xm* para realocar recursos.

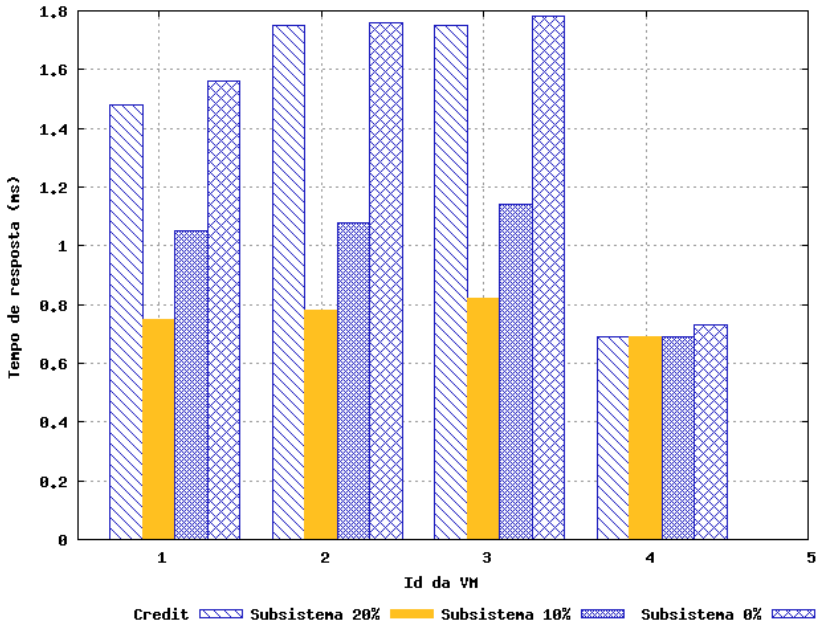

Figura 29 – Sobrecarga em 3 VMs

Os resultados apresentados nesta seção foram obtidos comparando-se apenas o desempenho do escalonador Credit em modo estático, com o subsistema de realocação. Para demonstrar o desempenho do subsistema frente aos demais escalonadores disponíveis no Xen e configurados em seus distintos modos de operação, se faz necessário um amplo cenário de teste. Por este motivo, foi utilizado o processo de geração de casos de teste baseado em modelos UML e a ferramenta de teste de desempenho Jmeter.

6.2.2 Casos de testes e resultados com o Jmeter

Os testes realizados com a ferramenta Jmeter buscam gerar resultados que possibilitem comparar o desempenho do subsistema, quando configurado com diferentes percentagens de recursos livres, com os demais escalonadores do Xen, em situações onde todas as VMs da estrutura recebem a mesma carga de usuários. Esta situação busca avaliar o desempenho do subsistema

em uma situação que simule o pior caso possível de uso do mesmo, que é aquele em que todas as VMs da estrutura disputam os recursos livres alocados pelo subsistema.

Para gerar os resultados foram realizadas diversas cargas com a ferramenta de teste de desempenho Jmeter sobre o ambiente, descrito na seção 6.1, variando-se de 5 a 75 a quantidade de usuários simultâneos, sendo que cada caso de teste foi realizado repetidamente com o Xen utilizando-se dos seguintes escalonadores em diferentes configurações:

- SEDF;
- Credit em modo dinâmico;
- Credit em modo estático;
- Credit em modo estático com o subsistema de realocação realocando 20% dos recursos da infra-estrutura;
- Credit em modo estático com o subsistema de realocação realocando 10% dos recursos da infra-estrutura;
- Credit em modo estático com o subsistema de realocação realocando 0% dos recursos da infra-estrutura (sem recurso livre);

Com o objetivo de se obter os dados necessários para se comparar o desempenho do subsistema, com diferentes níveis de recursos, com os escalonadores do Xen configurados em seus distintos modos, foram realizados 840 execuções de *benchmark*, totalizando 210 horas de execução dos testes. Apesar do número de casos de teste ser superior ao do TPC-W, o tempo de execução foi reduzido através da realização de casos de teste com menor tempo de duração e a utilização de teste baseado em modelos.

Os resultados obtidos (ver Figura 30) com os testes executados sobre a aplicação demonstram que a utilização do escalonador Credit em conjunto com o subsistema, quando existe 20% de recurso livre, proporciona uma melhora significativa no desempenho das aplicações hospedadas nas VMs, quando comparado com o Credit estático. Nota-se também neste caso que a diferença de desempenho entre o escalonador Credit e o subsistema, tende a ser menor quando poucos usuários estão acessando o servidor ou quando à carga de usuários é muito superior a capacidade do servidor. A primeira situação é motivada pelo fato de que em situações onde o número de usuários simultâneos na aplicação é baixo, o processamento nas VMs não aumenta a ponto do subsistema atuar efetivamente, e nas poucas situações em que aumenta, a quantidade adicional de recurso não se traduz em uma considerável diferença no tempo de resposta, dada a pouca carga na aplicação. Quanto à diferença no desempenho ser menor quando muitos usuários acessam a estrutura, isso se deve ao fato de que quando a capacidade do servidor é ultrapassada, o subsistema não consegue atender às necessidades por mais recursos por parte de todas as VMs, sendo que em muitos casos, determinada VM recebe os recursos e os utiliza durante todo o período do teste.

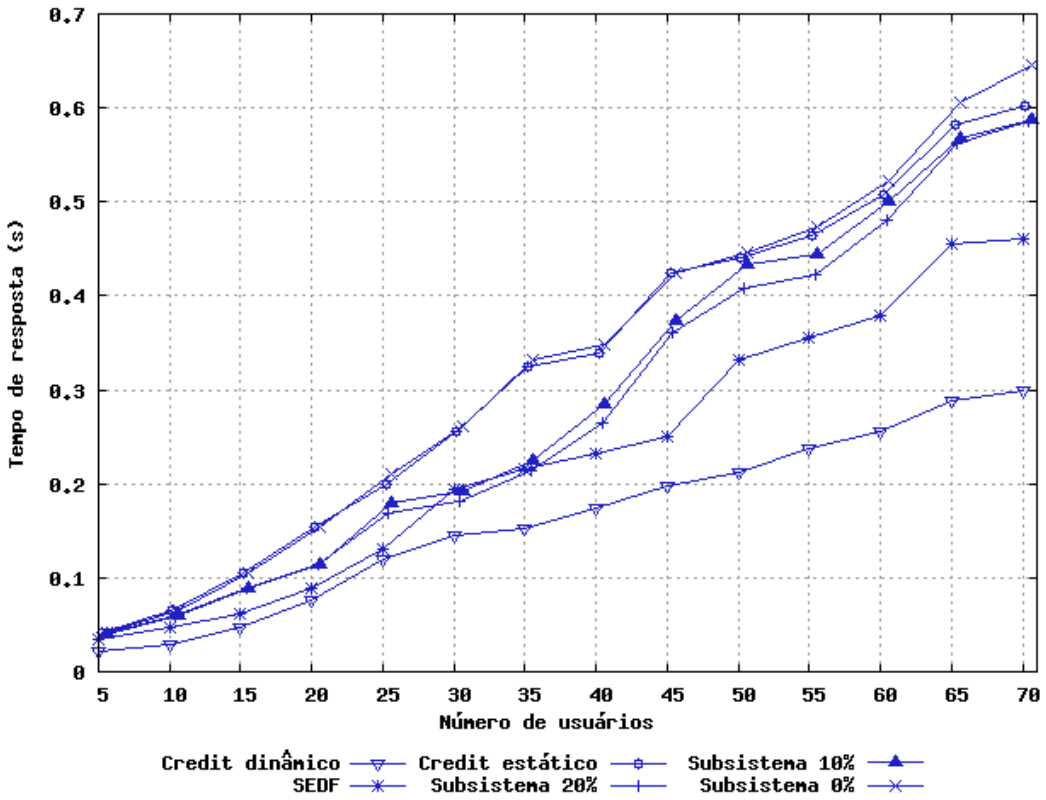

Figura 30 – Comparação do desempenho do subsistema com os escalonadores do Xen

Com o objetivo de se conhecer qual é o impacto no desempenho das aplicações quando o subsistema possui uma menor quantidade de recursos livres para realocar, foram realizados os mesmos testes descritos anteriormente, só que com 10% de recursos livres. Os resultados obtidos com esses testes demonstram que a queda no desempenho da aplicação, quando comparado com o teste onde o subsistema possuía 20% de recursos para realocar, foi inexistente em situações onde poucos usuários acessavam a aplicação. No entanto, conforme a carga de usuários sofreu um incremento, a diferença no desempenho tende a tornar-se maior. Outra situação análoga ao teste com 20% de recursos ocorre quando a quantidade de usuários está próxima do limite do que a VM suporta. Nessa situação, o desempenho do subsistema, em seus distintos níveis de recursos, tende a tornar-se bastante próximo. Comparando-se o desempenho do subsistema quando com o escalonador Credit em modo estático, a utilização do subsistema com 20% e 10% de recursos livres proporciona uma melhora no desempenho das aplicações em todos os casos de teste. Sendo que, a utilização do subsistema, apenas em uma configuração, com 0% de recursos para realocar, apresentou um desempenho inferior ao obtido com o escalonador Credit em modo estático. Este fato é motivado pelo próprio custo computacional da execução e monitoração do subsistema, já que apesar de não tentar realocar recursos livres, quando os mesmos não estão disponíveis, o subsistema continua a monitorar as VMs em busca de situações que quebrem o SLA, e a quantidade de recursos livres no sistema. Esta monitoração é necessária para que o subsistema identifique quando algum recurso livre torna-se disponível no

ambiente virtualizado, sendo esta situação possível, por exemplo, caso uma VM seja realocada para outra estrutura, ou removida pelo administrador.

Apesar de apenas o escalonador Credit configurado em modo estático possibilitar a utilização de SLAs em ambientes virtualizados e a realocação de recursos através do subsistema, também foram realizados os testes com os escalonadores SEDF e Credit em modo dinâmico.

Os resultados obtidos com esses escalonadores foram os que apresentaram os melhores resultados, uma vez que o bom desempenho do Credit dinâmico já era esperado, pois o próprio escalonador já possui internamente as funções necessárias para a realocação, não necessitando executar chamadas a bibliotecas de monitoração e a ferramentas de alocação de recursos. Um resultado não esperado foi o desempenho do escalonador SEDF, que apesar de ser inferior ao Credit dinâmico, foi superior aos demais escalonadores e ao subsistema. Esse resultado é relevante para a continuidade no desenvolvimento do mesmo, pois devido ao seu suposto fraco desempenho, não é prevista a inclusão desse escalonador na próxima versão do Xen.

6.3 Considerações Finais

Com o objetivo de se comparar os diversos diferentes desempenhos do subsistema proposto neste trabalho, foi implementado um ambiente de teste que busca simular de maneira realista um *site* de comércio eletrônico, sendo realizado sobre este ambiente uma carga de usuários variável. Também foram realizados testes com o objetivo de se conhecer o desempenho proporcionado pelo uso desse subsistema em um ambiente virtualizado onde todas as VMs recebem de maneira crescente uma mesma carga de usuários.

Conforme os resultados obtidos com estes testes, pode-se observar que a utilização do subsistema, além de possibilitar a utilização de SLAs em ambientes virtualizados e a realocação de recursos, proporciona uma significativa melhora no desempenho das aplicações hospedadas nesses ambientes quando comparado com o escalonador Credit em modo estático.

Os testes também demonstraram algumas limitações, como a necessidade de se implementar um mecanismo que regule o tempo que cada VM pode se utilizar dos recursos livres, já que em situações de sobrecarga uma determinada VM pode monopolizar a utilização dos recursos, e o impacto no desempenho quando o subsistema não possui recursos livres. No próximo capítulo serão apresentadas as conclusões sobre a proposta descrita nesse trabalho, bem como algumas sugestões de trabalhos futuros.

7 Conclusão

Conforme foi apresentado neste trabalho, a utilização de SLAs em um ambiente virtual oferece uma garantia formal de que um nível de serviço será atingido, ou quais as penalidades caso isso não aconteça. No entanto, uma definição precisa dos indicadores de níveis de serviço (SLI) é uma tarefa complexa e que necessita de constantes ajustes.

Este trabalho descreveu uma proposta, implementação e avaliação de um processo para definição e realocação de recursos em ambientes virtualizados baseados em SLAs, que é dividido em duas etapas: a definição de SLAs em ambientes virtualizados é realizada através da decomposição de métricas de alto nível em métricas de baixo nível, e a realocação, com base na decomposição, dos recursos físicos (processador e memória) entre as VMs. Para se realizar a decomposição de SLA, foram utilizadas ferramentas de *benchmark* para realizar exaustivos testes na aplicação hospedada na estrutura virtualizada, de modo que tornar-se possível conhecer de forma precisa os níveis de recursos utilizados por cada VM sobre uma determinada carga de trabalho. A realocação de recursos busca melhorar a utilização do *hardware* e garantir o SLA em algumas situações de sobrecarga das VMs.

Para prover essas funcionalidades, esta proposta se baseou em duas principais estratégias. A primeira está em utilizar testes baseados em modelos para automatizar a geração e execução dos testes na aplicação, sendo que os resultados desses testes são utilizados para realizar a decomposição de SLA, que é utilizada para indicar qual o nível de recurso deve ser configurado para que determinada aplicação hospedada em uma VM mantenha o SLA contratado. A segunda estratégia se baseia na realocação de recursos, com base nos SLA e na decomposição de SLA. Para efetuar a reconfiguração foi implementado um subsistema que tem por objetivo otimizar a utilização do *hardware* e efetuar a utilização dos SLAs definidos. Para realocar os recursos, esse subsistema faz uso dos dados da decomposição de SLA obtidos através de testes de desempenho.

A avaliação do processo proposto foi realizada através da utilização da execução de *benchmarks* sobre as aplicações, sendo realizadas com diferentes cargas de usuários e simulando diversas situações. Os resultados apresentados demonstraram que com a utilização do subsistema implementado, o provedor consegue garantir o serviço com os níveis definidos no SLA, e em algumas situações de sobrecarga da VM. Deste modo, ele garante recompensas da parte do cliente, pelo fato do serviço superar as expectativas do nível de serviço acordado e melhora a utilização dos recursos da estrutura, diminuindo os custos. Os resultados também demonstraram que a utilização do subsistema proporciona uma melhora no desempenho das aplicações hospedadas nas VMs, estando às mesmas com sobrecarga ou não.

Para aprimorar o subsistema desenvolvido propomos como possíveis trabalhos futuros, automatizar e refinar o processo de decomposição de SLA durante a execução da aplicação, se utilizando dos testes baseados em modelos apenas para definir os níveis de recursos iniciais para cada aplicação, sendo os demais dados coletados durante a utilização da aplicação no ambiente. Esta abordagem proporcionaria uma maior redução do tempo de execução dos testes e tornaria os resultados da decomposição mais precisos, pois captura os dados necessários para a decomposição de SLA em tempo de execução, sendo que esses dados são mais precisos, pois são gerados por uma carga real de usuários na aplicação. Outro ponto a ser trabalhado, é a possibilidade de se realocar os recursos não utilizados pelas VMs, mas que se encontram dentro do nível de recurso definido no SLA. Entretanto, para ser possível a realocação destes recursos, se faz necessária a utilização de alguma técnica de predição que possibilite avaliar quais as necessidades futuras por recursos de uma VM, de modo a não quebrar o SLA quando esta aplicação volte a necessitar dos mesmos.

Referências

[1] Introduction to the Xen Virtual Machine. Acessado em 09/10/2008. Disponível em: <http://www.linuxjournal.com/article/8540>.

[2] Transactional Processing Performance Council. Acessado em 01/02/2008. Disponível em: <http://www.tpc.org/tpcw/>.

[3] GUPTA, D.; GARDNER, R.; CHERKASOVA, L. *XenMon: QoS Monitoring and Performance Profiling Tool*. Palo Alto, CA, USA, 2005.

[4] SMITH, J.; NAIR, R. *Virtual Machines: Versatile Platforms for Systems and Processes*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[5] CHEN, Y. et al. Sla decomposition: Translating service level objectives to system level thresholds. In: *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2007. p. 3.

[6] KELLER, E.; LUDWIG, H. Defining and monitoring service-level agreements for dynamic e-business. In: *LISA '02: Proceedings of the 16th USENIX conference on System administration*. Berkeley, CA, USA: USENIX Association, 2002. p. 189–204.

[7] SCHWEITZER, C. M. *Informação de Desempenho e Acordos de Nível de Serviço para Redes de Transporte PDH e SDH*. Dissertação (Mestrado) — Centro Federal de Educação Tecnológica do Paraná, PR, Brasil, 1999.

[8] WANG, Z. et al. Capacity and performance overhead in dynamic resource allocation to virtual containers. *Integrated Network Management, IM '07. 10th IFIP/IEEE International Symposium*, p. 149–158, 2007.

[9] GOIRI, I.; GUITART, J. Autonomic resource management for the xen hypervisor. In: *1st Workshop on Execution Environments for Distributed Computing*. [S.l.: s.n.], 2007. p. 17–24.

[10] ROSSI, F. D. *Alocação Dinâmica de Recursos no Xen*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, RS, Brasil, 2008.

[11] WINCK, A. T. *Um processo de KDD para auxílio à reconfiguração de ambientes virtualizados*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, RS, Brasil, 2008.

[12] RODRIGUES, E. M. et al. Uso de Modelos Preditivos e SLAs para Reconfiguração de Ambientes Virtualizados. In: *WSO – Workshop de Sistemas Operacionais*. Belém, PA: SBC, 2008. p. 147–158.

[13] CREASY, R. J. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, IBM Systems Journal, NY, USA, v. 25, n. 5, p. 483–490, 1981.

[14] ROSENBLUM, M. The reincarnation of virtual machines. *Queue*, ACM, New York, NY, USA, v. 2, n. 5, p. 34–40, 2004.

[15] SHANLEY, T.; ANDERSON, D. *ISA System Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[16] Bochs User Manual. Acessado em 25/09/2008. Disponível em: <http://bochs.sourceforge.net/doc/docbook>.

[17] BELLARD, F. Qemu, a fast and portable dynamic translator. In: *Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005. p. 41–51.

[18] LAWTON, K. P. Bochs: A portable pc emulator for unix/x. *Linux J.*, Specialized Systems Consultants, Inc., Seattle, WA, USA, p. 7.

[19] YU, Y. et al. A feather-weight virtual machine for windows applications. In: *Proceedings of the 2nd international conference on Virtual execution environments*. New York, NY, USA: ACM, 2006. p. 24–34.

[20] MENASCE, D. A.; BENNANI, M. N. Autonomic virtualized environments. In: *Proceedings of the International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2006. p. 28.

[21] VMWare: Virtualization via Hypervisor, Virtual Machine e Server Consolidation. Acessado em 25/09/2008. Disponível em: <http://www.vmware.com>.

[22] Virtual PC. Acessado em 25/09/2008. Disponível em: <http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx>.

[23] Lightweight virtual machines for distributed and networked systems. Acessado em 25/09/2008. Disponível em: <http://denali.cs.washington.edu/>.

[24] Xensource. Acessado em 25/09/2008. Disponível em: <http://www.xensource.com/Pages/default.aspx>.

[25] Virtualization Products,Virtual Infraestructure, Virtual Machine. Acessado em 25/09/2008. Disponível em: <http://www.vmware.com/products/>.

[26] CROSBY, S.; BROWN, D. The virtualization reality. *Queue*, ACM, New York, NY, USA, v. 4, n. 10, p. 34–41, 2007.

[27] The FreeBSD Diary–Jails under FreeBSD 6. Acessado em 25/09/2008. Disponível em: <http://www.freebsddiary.org/jail-6.php>.

[28] HOSKINS, M. E. User-mode linux. *Linux J.*, Specialized Systems Consultants, Inc., Seattle, USA, v. 2006, n. 145, p. 2, 2006.

[29] The FreeBSD Project. Acessado em 01/11/2008. Disponível em: <http://www.freebsd.org/>.

[30] KAMP, P. H.; WATSON, R. N. M. Jails: Confining the omnipotent root. In: *In Proceedings of the 2nd International SANE Conference*. Maastricht, Netherlands: [s.n.], 2000.

[31] Myung-Sup Kim, H. Mapping between QoS parameters and network performance metrics for SLA monitoring. *Consumer Communications and Networking Conference*, p. 1–12, 2005.

[32] Java Technology. Acessado em 25/09/2008. Disponível em: <http://www.sun.com/java/>.

[33] CZAJKOWSKI, G. Application isolation in the java virtual machine. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 35, n. 10, p. 354–366, 2000.

[34] AMD Virtualization. Acessado em 25/09/2008. Disponível em: <http://www.amd.com/virtualization>.

[35] Intel Virtualization Technology in computing. Acessado em 25/09/2008. Disponível em: <http://http://www.intel.com/technology/virtualization/>.

[36] BARHAM, P. et al. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 37, n. 5, p. 164–177, 2003.

[37] Intel 64 and IA-32 Architectures Software Developer Manual. Acessado em 09/10/2008. Disponível em: <http://download.intel.com/design/processor/manuals/253665.pdf>.

[38] Xen User Manual. Acessado em 09/10/2008. Disponível em: <http://www.xen.org/files/xen_user_manual.pdf>.

[39] The Virtualization API. Acessado em 20/09/2008. Disponível em: <http://www.libvirt.org/>.

[40] CHISNALL, D. *The Definitive Guide to the Xen Hypervisor*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[41] CHERKASOVA, L.; GUPTA, D.; VAHDAT, A. Comparison of the three CPU schedulers in Xen. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 35, n. 2, p. 42–51, 2007.

[42] DUDA, K. J.; CHERITON, D. R. Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 33, n. 5, p. 261–276, 1999.

[43] LIN, C. *Unified and effective soft real-time processing in integrated systems*. Tese (Doutorado) — University of California, Santa Cruz, CA, USA, 2006.

[44] UDUPI, Y.; SAHAI, A.; SINGHAL, S. *A Classification-Based Approach to Policy Refinement*. Palo Alto, CA, USA, 2007.

[45] AL, J. S. et. Optimal choice of service level objectives from a business perspective. *2nd Annual Workshop of HP openview university association*, p. 1–14, 2005.

[46] MARQUES, F. T. *Projeto de Infra-estrutura de TI pela Perspectiva de Negócio*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, Campina Grande, PB, Brasil, 2006.

[47] BARBOSA, A. C. B. *Avaliando Arquiteturas de Auditoria de Acordos de Nível de Serviço para Web Services e Grid Services*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, Campina Grande,PB, Brasil, 2005.

[48] GUEDES, M. V. A. *Um modelo de qualimetria para a gestão da qualidade nos serviços de Telecomunicações*. Dissertação (Mestrado) — Universidade Católica de Brasília, Brasília, DF, Brasil, 2002.

[49] VIANA, A. C. et al. Perspectivas sobre Qualidade de Serviço nos Protocolos da Internet - Estudo de Caso: Aplicações de Vídeo Sob Demanda. *II Workshop RNP2*, p. 1–12, 2000.

[50] GUARDIEIRO, P. R.; MAURíCIO, M. Avaliação de um ambiente de serviços diferenciados com tráfego de vídeo MPEG-4. *Anais do 19º Simpósio Brasileiro de Telecomunicações - SBrT*, p. 1–6, 2001.

[51] MELO, E. T. L. *Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de Medições*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, SC, Brasil, 2001.

[52] STRIEGEL, A. Security issues in a differentiated services internet. *Workshop Trusted Internet - HiPC, Bangalore, India*, p. 1–4, 2002.

[53] KOUCHERYAVY, Y.; MOLTCHANO, D.; HARJU, J. A top-down approach to VoD traffic transmission over DiffServ domain using AF PHB class. *IEEE International Conference on Communications*, v. 1, p. 243–249 vol.1, 2003.

[54] HYO-JIN, L.; MYUNG-SUP, K. Mapping between QoS Parameters and Network Performance Metrics for SLA monitoring. *Consumer Communications and Networking Conference*, p. 1–12, 2005.

[55] NETTO, M.; BARCELOS, A.; ROSE, C. A. D. Desenvolvimento de sistemas de gerência de agregados para plataforma GNU/Linux. *III Workshop de Software Livre (WSL 2002)*, p. 1–4, 2002.

[56] YEO, C. S.; BUYYA;, R. Service Level Agreement based allocation of cluster resources: Handling penalty to enhance utility. *7th IEEE International Conference on Cluster Computing*, p. 1–10, 2005.

[57] SAHAI, A. et al. Specifying and monitoring guarantees in commercial grids through sla. In: *Proceedings of the 3st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2003. p. 292.

[58] RIGHI, R.; PELLISSARI, F.; WESTPHALL, C. Sec-SLA: especificação e validação de métricas para acordos de níveis de serviço orientados à segurança. In: *IV Workshop em Segurança de Sistemas Computacionais (WSeg) - Simpósio Brasileiro de Redes e Computadores*. Gramado, RS, Brasil: SBC, 2004. p. 199–210.

[59] JANAKIRAMAN, J.; SANTOS, J. R.; TURNER, Y. Automated system design for availability. In: *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2004. p. 411.

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

[61] Object Management Group - OMG. Unified Modeling Language. Acessado em 20/09/2008. Disponível em: <http://www.omg.org/docs/formal/07-02-06.pdf>.

[62] OROZCO, A. *Linha de produto de teste baseado em modelos*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, RS, Brasil, 2009.

[63] PERALTA, K. P. *Uma Estratégia para Especificação e Geração de Casos de Teste de Segurança usando Modelos UML*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, RS, Brasil, 2008.

[64] Apache Jmeter. Acessado em 01/02/2008. Disponível em: <http://jakarta.apache.org/jmeter/>.

[65] Apache HTTP Server Project. Acessado em 01/02/2008. Disponível em: <http://httpd.apache.org/>.

[66] Apache Tomcat. Acessado em 01/02/2008. Disponível em: <http://tomcat.apache.org/>.

[67] MySQL. *The world's most popular open source database*. Acessado em 01/02/2008. Disponível em: <http://www.mysql.com/>.

[68] Amazon.com: Online Shopping for Electronics, Computers, Books, DVDs & more. Acessado em 05/02/2008. Disponível em: <http://www.amazon.com>.

[69] LIU, X.; HEO, J.; SHA, L. Modeling 3-tiered web applications. In: *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. Washington, DC, USA: IEEE Computer Society, 2005. p. 307–310.

[70] OSOGAMI, T. Relations in the central limit theorem version of the response time law. In: *Fourth International Conference on Quantitative Evaluation of Systems*. Washington, DC, USA: IEEE Computer Society, 2007. p. 69–78.

[71] SOPITKAMOL, M.; MENASCé, D. A. A method for evaluating the impact of software configuration parameters on e-commerce sites. In: *WOSP '05: Proceedings of the 5th international workshop on Software and performance*. New York, NY, USA: ACM, 2005. p. 53–64.