FERNANDO JOSÉ MUCHALSKI

ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM AMBIENTES DE COMPUTAÇÃO EM NUVEM CONSIDERANDO O COMPARTILHAMENTO DE MEMÓRIA

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Curitiba PR
Agosto de 2014

FERNANDO JOSÉ MUCHALSKI

ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM AMBIENTES DE COMPUTAÇÃO EM NUVEM CONSIDERANDO O COMPARTILHAMENTO DE MEMÓRIA

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Área de concentração: *Engenharia de Sistemas Computacionais*

Orientador: Carlos Alberto Maziero

Curitiba PR
Agosto de 2014

Dados Internacionais de Catalogação na Publicação

M942a 2014

Muchalski, Fernando José
Alocação de máquinas virtuais em ambientes de computação em nuvem considerando o compartilhamento de memória / Fernando José Muchalski.-- 2014.
86 f.: il.; 30 cm

Texto em português, com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Computação Aplicada, Curitiba, 2014.
Bibliografia: f. 61-65.

1. Sistema de computação virtual. 2. Computação em nuvem. 3. Algoritmos. 4. Gerenciamento de memória (Computação). 5. Cliente/servidor (Computação). 6. Qualidade de serviço (Redes de computação). 7. Simulação (Computadores). 8. Computação - Dissertações. I. Maziero, Carlos Alberto, orient. II. Universidade Tecnológica Federal do Paraná - Programa de Pós-graduação em Computação Aplicada. III. Título.

CDD 22 -- 621.39

Biblioteca Central da UTFPR, Câmpus Curitiba

ATA DA DEFESA DE DISSERTAÇÃO DE MESTRADO 24

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE EM **COMPUTAÇÃO APLICADA**

No dia 29 de agosto de 2014, às 13 h30min, reuniu-se na Sala B-204 - bloco B - 2º andar do Câmpus Curitiba, a banca examinadora composta pelos professores doutores:

Prof. Carlos Alberto Maziero, Dr. (Presidente)	UTFPR - CT
Prof. Alcides Calsavara, Dr.	PUC-PR
Prof. Ana Cristina Barreiras Kochem Vendramin, Drª.	UTFPR - CT
Prof. Luiz Nacamura Júnior, Dr.	UTFPR - CT

sob Presidência de **Murilo Vicente Gonçalves da Silva** para examinar a dissertação do candidato **Fernando José Muchalski**, intitulada**:** "Alocação de Máquinas Virtuais em Ambientes de Computação em Nuvem considerando o compartilhamento de Memória"**.** Após a apresentação, o candidato foi arguido pelos examinadores e foi dada a palavra aos presentes para formularem perguntas ao candidato. Os examinadores reunidos deliberaram pela _____ da dissertação.

O candidato foi informado que a concessão do referido grau, na área de concentração Engenharia de Sistemas Computacionais, está condicionada à (i) satisfação dos requisitos solicitados pela Banca Examinadora e lavrados na documentação entregue ao Candidato; (ii) entrega da dissertação em conformidade com as normas exigidas pela UTFPR; e (iii) entrega da documentação necessária para elaboração do Diploma. A Banca Examinadora determina um **prazo de _____ dias** para o cumprimento dos requisitos (desconsiderar esse parágrafo caso a dissertação seja reprovada).

_____

_____

_____

Nada mais havendo a tratar, a sessão foi encerrada às _____, dela sendo lavrada a presente ata que segue assinada pela Banca Examinadora e pelo Candidato.

_____
Prof. **Carlos Alberto Maziero, Dr.**
Presidente - (UTFPR - CT)

_____
Prof. Alcides Calsavara, Dr.
(PUC-PR)

_____
Profª. Ana Cristina Barreiras Kochem Vendramin, Drª.
(UTFPR)

_____
Prof. Luiz Nacamura Júnior, Dr.
(UTFPR - CT)

Candidato: _____

DECLARAÇÃO PARA A OBTENÇÃO DO GRAU DE MESTRE

A coordenação do Programa declara que foram cumpridos todos os requisitos exigidos pelo Programa de Pós-Graduação para a obtenção do grau de mestre.

Curitiba, ____de _____de 20____.

"A Ata de Defesa original está arquivada na Secretaria do PPGCA".

Agradecimentos

Inicialmente gostaria de agradecer a minha esposa e filho, pela compreensão e paciência durante esse período de pesquisa e muito trabalho e sobretudo pelo apoio e incentivo oferecido nos momentos mais difíceis.

Aos meus pais, pela dedicação e esforço para me proporcionar uma sólida formação, que hoje me permite concluir esse mestrado.

A minha irmã, pela hospitalidade ao me receber em sua casa e pelas várias caronas até Curitiba.

Aos meus colegas de empresa, pela compreensão e colaboração durante os dias que estive ausente.

Ao professor Dr. Carlos Alberto Maziero, pelas orientações e empenho que proporcionaram a realização desse trabalho, para mim foi um período de aprendizado gratificante.

Por fim, meus agradecimentos aos demais professores e colaboradores da UTFPR, aos colegas de mestrado e às pessoas que de alguma forma participaram dessa fase da minha vida.

Resumo

A virtualização é uma tecnologia chave para a computação em nuvem que permite fornecer recursos computacionais, em forma de máquinas virtuais, para o consumo de serviços de computação. Nos ambientes de computação em nuvem, é importante manter sob controle a alocação de máquinas virtuais nos servidores físicos. Uma alocação adequada implica na redução de custos com hardware, energia e refrigeração, além da melhora da qualidade de serviço. Hipervisores recentes implementam mecanismos para reduzir o consumo de memória RAM através do compartilhamento de páginas idênticas entre máquinas virtuais. Esta dissertação apresenta um novo algoritmo de alocação de máquinas virtuais que busca o equilíbrio no uso dos recursos de CPU, memória, disco e rede e, sobretudo, considera o potencial de compartilhamento de memória entre máquinas virtuais. Através de simulações em cenários distintos, verificou-se que o algoritmo é superior à abordagem padrão na questão do uso equilibrado de recursos e que, considerando o compartilhamento de memória, houve um ganho significativo na disponibilidade deste recurso ao final das alocações.

Palavras-chave: virtualização, computação em nuvem, alocação de máquinas virtuais.

Abstract

Virtualization is a key technology for cloud computing, it provides computational resources as virtual machines for consumption of computing services. In cloud computing environments it is important to keep under control the allocation of virtual machines in physical servers. A good allocation brings benefits such as reduction costs in hardware, power, and cooling, also improving the quality of service. Recent hypervisors implement mechanisms to reduce RAM consumption by sharing identical pages between virtual machines. This dissertation presents a new algorithm for virtual machines allocation that seeks the balanced use of CPU, memory, disk, and network. In addition, it considers the potential for sharing memory among virtual machines. Simulations on three distinct scenarios demonstrate that it is superior to the standard approach when considering the balanced use of resources. Considering shared memory, there was an appreciable gain in availability of resources.

Keywords: virtualization, cloud computing, virtual machine allocation.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

α Fator de compartilhamento
γ Limitador mínimo de recursos
δ Limitador máximo de recursos

Lista de Abreviações

AE	*Application Environment*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CBPS	*Content Based Page Sharing*
CEO	*Chief Executive Officer*
CLI	*Command Line Interface*
CPU	*Central Process Unit*
EBS	*Elastic Block Store*
EC2	*Elastic Computing Cloud*
GDM	*Global Decision Module*
GUI	*Graphical User Interface*
IaaS	*Infrastructure as a Service*
IAM	*Identity and Access Management*
KP	*Knapsack Problem*
KSM	*Kernel Samepage Merging*
KVM	*Kernel-based Virtual Machine*
LDAP	*Lightweight Directory Access Protocol*
LDM	*Local Decision Module*
MDKP	*Multidimensional Knapsack Problem*
MIPS	*Millions of Instructions Per Second*
MKP	*Multiple Knapsack Problem*
NIST	*National Institute of Standards and Technology*
OCCI	*Open Cloud Computing Interface*
OGF	*Open Grid Forum*
OVF	*Open Virtualization Format*
PaaS	*Platform as a Service*
QoS	*Quality of Service*
S3	*Simple Storage Service*
SLA	*Service Level Agreement*
SaaS	*Software as a Service*
TI	Tecnologia da Informação
VM	*Virtual Machine*
VMM	*Virtual Machine Monitor*

Capítulo 1

Introdução

A *Computação em Nuvem* é um recente paradigma computacional que vem recebendo grande atenção nos últimos anos. Ela possibilita o fornecimento de recursos computacionais, em forma de serviço, utilizando a internet como meio de acesso. Para o cliente, representa a possibilidade de redução de custos com a sua infraestrutura computacional, aliados à flexibilidade de acesso e escalabilidade. A variedade dos serviços oferecidos neste ambiente vai desde softwares prontos para o uso até uma completa infraestrutura virtualizada de TI, envolvendo servidores, rede e armazenamento (AlZain et al., 2012).

Para dar suporte a essa demanda, os provedores de serviços mantêm modernos centros de dados formados por milhares de servidores físicos que processam e tratam diferentes tipos de aplicações. Esses servidores, através da tecnologia de virtualização, disponibilizam seus recursos em forma de máquinas virtuais (VM, do inglês *Virtual Machines*).

Para garantir uma melhor utilização dos recursos físicos disponíveis é importante manter sob controle a alocação dessas máquinas virtuais. Uma alocação adequada traz como benefícios uma melhora na qualidade do serviço além da redução de custos com equipamentos, energia e refrigeração.

A tecnologia de máquinas virtuais não é nova, ela surgiu no final de década de 1960 para fazer melhor uso dos recursos dos computadores da época (Creasy, 1981). Com a disseminação dos computadores pessoais perdeu importância, mas recentemente emergiu como uma das tecnologias chaves para a computação em nuvem ao permitir a um centro de dados servir múltiplos usuários de uma forma segura, flexível e eficiente, principalmente devido a sua capacidade de isolamento, consolidação e migração de cargas de trabalho (Voorsluys et al., 2009).

Essas características das máquinas virtuais possibilitam a criação de políticas para o gerenciamento da infraestrutura computacional, visando a garantia da qualidade de serviço (QoS, do inglês *Quality of Service*) e economia de recursos através da melhor utilização da capacidade computacional. O desafio envolve a configuração e instanciação de imagens de sistemas operacionais em forma de máquinas virtuais, a ligação destas máquinas virtuais com os servidores físicos e o gerenciamento e controle de suas interações com o ambiente (Grit et al., 2006).

Uma forma de garantir o uso eficiente dos recursos de *hardware* de um centro de dados é através da alocação correta das máquinas virtuais nos servidores físicos. Esse mapeamento "correto" requer o conhecimento da capacidade dos servidores e dos recursos requeridos pela máquina virtual em questão, bem como das políticas do centro de dados para resolver possíveis

conflitos quanto à escolha do servidor físico adequado para se alocar uma máquina virtual (Hyser et al., 2007).

1.1 Motivação

Acompanhando a evolução da computação em nuvem, vários algoritmos e técnicas tem surgido em busca da melhor alocação de VMs nos centros de dados. Esses algoritmos tratam o problema de alocação a partir de enfoques distintos como: alocação inicial de VMs, empacotamento de um conjunto prévio de VMs, consolidação de servidores, balanceamento de carga, respeito aos contratos de serviço, custos operacionais, fatores ambientais, etc. (Mills et al., 2011).

Em uma análise desses algoritmos, foi percebido que boa parte deles buscam a otimização dos recursos, seja minimizando a quantidade de servidores utilizados ou minimizando um custo de alocação. Muitos deles utilizam apenas uma dimensão de recurso para análise, geralmente a capacidade de CPU, enquanto outros tratam de analisar um custo da alocação, como o gasto energético.

Poucos algoritmos tratam de uma importante característica presente em hipervisores modernos, que é a possibilidade de compartilhamento de páginas de memória entre máquinas virtuais alocadas em um mesmo servidor físico. E quando tratada, não levam em consideração as demais dimensões de recursos.

Diante disso, o presente trabalho foi motivado pelo fato de não ser encontrado um algoritmo que permita selecionar o melhor servidor físico para alocar uma nova VM a ser instanciada, que leve em consideração as várias dimensões de recursos como memória, CPU, disco e rede (mas não limitados a esses) e sobretudo leve em consideração a possibilidade de compartilhamento de páginas de memória entre VMs.

Nesse contexto, o presente trabalho apresenta um novo algoritmo para alocação de máquinas virtuais, onde serão avaliados os recursos de CPU, memória, disco, rede e o potencial de compartilhamento de páginas de memória da VM a ser alocada.

1.2 Objetivos

O objetivo geral da pesquisa é desenvolver um algoritmo para alocação dinâmica de máquinas virtuais, que leve em consideração as múltiplas dimensões de recursos, como CPU, memória, disco e rede e que considere o potencial de compartilhamento de páginas de memória entre VMs.

Os objetivos específicos são:

- Desenvolver o algoritmo de alocação de máquinas virtuais;
- Validar o algoritmo em um simulador de uma infraestrutura em nuvem;
- Projetar cenários distintos para realização de experimentos;
- Comparar os resultados do algoritmo com um algoritmo que utilize uma abordagem padrão de alocação;

- Comparar os resultados do algoritmo considerando o fator de compartilhamento de memória e sem considerá-lo;
- Elaborar uma análise estatística dos resultados obtidos.

1.3 Estrutura do documento

Este documento está estruturado da seguinte forma: o capítulo 2 apresenta a fundamentação teórica com os principais conceitos relacionados à computação em nuvem e à virtualização; o capítulo 3 apresenta trabalhos relacionados às estratégias de alocação de máquinas virtuais e alguns algoritmos propostos na literatura; o capítulo 4 apresenta a descrição e modelagem do algoritmo proposto; o capítulo 5 apresenta o ambiente de simulação utilizado na validação do algoritmo, os cenários testados e os resultados obtidos; por fim, o capítulo 6 apresenta as considerações finais do trabalho e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

A computação em nuvem é um novo paradigma computacional que envolve o consumo remoto de serviços de computação. Esses serviços são fornecidos por provedores que oferecem os recursos físicos de sua infraestrutura encapsulados em forma de múltiplas máquinas virtuais. Este capítulo apresenta a fundamentação teórica necessária à compreensão dos conceitos de computação em nuvem, sistema operacional de nuvem e virtualização.

2.1 Introdução

A presente pesquisa está inserida no ambiente de computação em nuvem, especificamente no gerenciamento da infraestrutura virtualizada dos centros de dados. Para melhor compreensão desse ambiente, neste capítulo será apresentada a fundamentação teórica para a construção da pequisa.

Desse modo, o capítulo está organizado da seguinte maneira: a seção 2.2 apresenta os conceitos relacionados à computação em nuvem, sua definição e características; a seção 2.3 apresenta a noção de sistema operacional de nuvem e algumas plataformas disponíveis para implantação de infraestruturas na nuvem; a seção 2.4 traz o conceito de computação em nuvem verde; por fim, a seção 2.5 trata sobre a virtualização e as técnicas que permitem o desenvolvimento de métodos para um melhor gerenciamento dos centros de dados.

2.2 Computação em nuvem

A Computação em Nuvem (do inglês, *Cloud Computing*) está relacionada a um novo paradigma de provisionamento e consumo de recursos de computação que deixa de ser realizado localmente e passa a ser entregue como um serviço disponibilizado via internet (Vaquero et al., 2009). A Nuvem é uma metáfora para descrever a utilização da internet como meio de acesso, sendo o símbolo da nuvem comum em diagramas e fluxogramas para representar a internet (Buyya et al., 2009). O termo ganhou popularidade após Eric Schmidt, CEO da Google, em 2006 ter usado a palavra para descrever o modelo de negócio para provimento de serviços através da internet. Desde então o termo "nuvem"tem sido usado como um termo de *marketing* numa variedade de contextos e para representar diferentes ideias (Zhang et al., 2010).

A variedade de serviços disponibilizados na nuvem é grande, incluindo o acesso remoto a *software*, armazenamento de arquivos, plataformas de desenvolvimento de *software* e

infraestruturas completas de TI envolvendo processamento, armazenamento e rede. Esses serviços são hospedados em modernos centros de dados mantidos por grandes empresas como Amazon, Google, Yahoo, Microsoft, Sun, etc (Abadi, 2009).

Em geral a mudança da computação tradicional para a computação em nuvem busca a redução dos custos associados a manutenção e ao gerenciamento dos recursos de *hardware* e *software* (Hayes, 2008), que ao serem terceirizados possibilitam o acesso sob demanda e com o pagamento baseado no uso, a exemplo de vários serviços de utilidade pública como água, eletricidade, telefonia e gás.

2.2.1 Definição

O termo Computação em Nuvem ainda tem causado confusão quanto ao que exatamente ela é e quando ela é útil, de modo que muitos trabalhos tentam encontrar a melhor definição ou esclarecer os conceitos que envolvem a computação em Nuvem. Como exemplo, (Vaquero et al., 2009) comparou mais de 20 diferentes definições de diversas fontes para confirmar uma definição padrão. Também (Armbrust et al., 2010) busca responder uma série de questões envolvendo o que é a computação em nuvem, o que a diferencia de outros paradigmas existentes e quais as oportunidades e obstáculos para o seu sucesso.

Atualmente, a definição mais aceita na comunidade acadêmica e que cobre todos os aspectos essenciais da computação em nuvem é dada pelo *National Institute of Standards and Technology* (NIST), instituto integrante do Departamento de Comércio dos Estados Unidos da América (Mell and Grance, 2011).

> ***Definição do NIST para Computação em Nuvem****: A Computação em Nuvem é um modelo que permite acesso conveniente, sob-demanda, via rede, a um pool compartilhado de recursos computacionais configuráveis (redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor de serviços.*

Este modelo de nuvem é composto ainda de cinco características essenciais, três modelos de serviço e quatro modelos de implantação que buscam caracterizar importantes aspectos da computação em nuvem.

2.2.2 Características

Segundo o NIST, algumas características são essenciais para que um serviço possa ser considerado com sendo um serviço na nuvem, são elas (Mell and Grance, 2011):

Autoatendimento: o consumidor pode, de forma unilateral, provisionar recursos computacionais como tempo de processamento e armazenamento em rede conforme sua necessidade sem que seja preciso uma interação humana com os provedores de serviço. Dessa forma todos os recursos são configurados, provisionados e liberados pelo próprio usuário;

Amplo acesso à rede: os recursos são disponibilizados via rede (internet) e são acessados por mecanismos padronizados que promovem o uso de plataformas heterogêneas como celulares, *smartphones*, *tablets*, *notebooks*, estações de trabalho e qualquer outra forma de acesso;

***Pool* de recursos:** Os recursos computacionais, físicos e virtuais, são agrupados na nuvem e disponibilizados de acordo com a demanda de consumo. A localização desses recursos é transparente ao usuário. Exemplo de recursos seriam armazenamento, processamento, memória e largura de banda de rede;

Elasticidade: Os recursos podem ser rapidamente provisionados e liberados, em alguns casos automaticamente, de acordo com a demanda de consumo. O consumidor tem a visão de que os recursos são infinitos e podem ser conseguidos a qualquer momento e em qualquer quantidade;

Serviços mensuráveis: O uso de recursos e serviços disponibilizados na nuvem são automaticamente controlados e otimizados através de métricas que promovem transparência ao provedor e ao consumidor sobre a utilização dos recursos. Essas métricas permitem a mensuração de acordo com o tipo de serviço fornecido (volume de armazenamento, tempo de processamento, largura de banda, atividades nas contas de usuário) e permitem a utilização do modelo de serviço *pay-per-use*, onde o pagamento pelos serviços é realizado de acordo com sua utilização (Buyya et al., 2009).

2.2.3 Modelos de serviço

Os serviços disponibilizados na nuvem podem ser visualizados como uma pilha de três camadas, a Figura 2.1 apresenta três categorias distintas de serviço na Computação em Nuvem: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS) (Zhang et al., 2010).

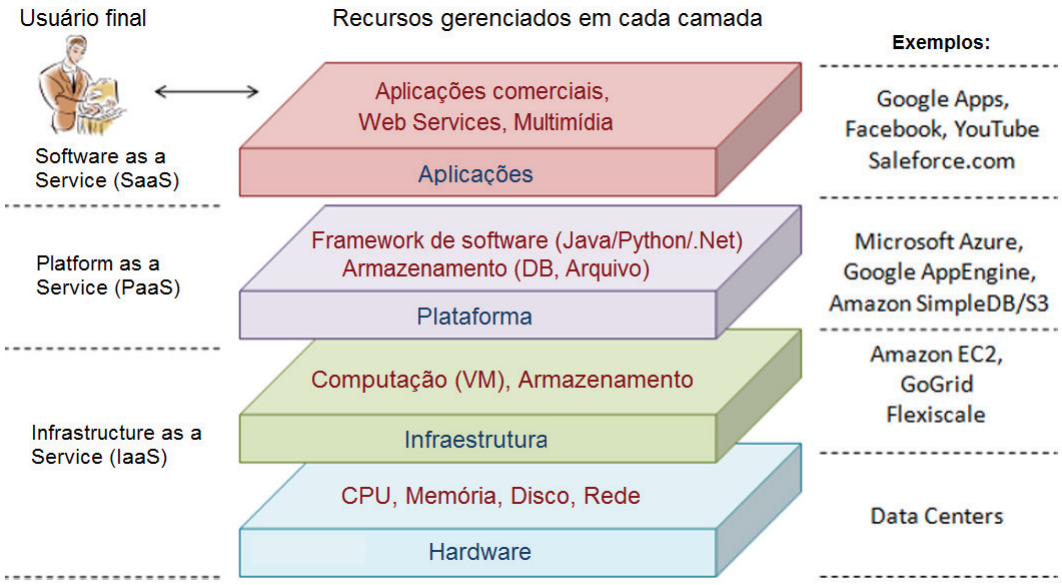


Figura 2.1: Pilha de Serviços da computação em nuvem
Fonte: Adaptado de (Zhang et al., 2010)

Esses três modelos de serviço estão relacionados e cada um oferece uma abstração para um conjunto de funcionalidades:

***Software as a Service* (SaaS):** no topo da pilha estão as aplicações que são desenvolvidas para o usuário final, sendo entregues via web e disponíveis através de qualquer equipamento que tenha acesso a internet. O usuário não precisa conhecer nenhum detalhe sobre a aplicação e seu papel é de consumidor do software;

***Platform as a Service* (PaaS):** na camada intermediária está a plataforma, que é formada por um conjunto de ferramentas e serviços disponibilizados para a construção e manutenção das aplicações SaaS. Nesse modelo, o usuário fica responsável pela utilização da plataforma para criação e manutenção de seu próprio software;

***Infrastructure as a Service* (IaaS):** na base da pilha estão o hardware e o software sobre os quais tudo é executado, os provedores de serviço disponibilizam um conjunto de recursos computacionais virtualizados (banda de rede, capacidade de armazenamento, memória, capacidade de processamento). É responsabilidade do usuário executar e manter o sistema operacional e as aplicações nesses recursos virtuais. IaaS utiliza a tecnologia de virtualização para converter os recursos físicos em recursos lógicos que podem ser dinamicamente provisionados e liberados pelos consumidores de acordo com a necessidade.

A Tabela 2.1 apresenta uma forma de identificar as responsabilidades do usuário e do provedor de serviços (CSP, do inglês *Cloud Service Provider*) de acordo com o modelo de serviço. No modelo *SaaS*, o usuário tem o papel apenas de consumidor, já que todos os recursos já foram disponibilizados pelo provedor de serviços. No *PaaS*, o usuário fica responsável pelo Software, neste modelo o provedor de serviços disponibiliza uma plataforma configurada restando ao usuário desenvolver suas aplicações. Por fim, no modelo *IaaS* o provedor de serviços fornece apenas a infraestrutura na forma de máquinas virtuais, ficando sob a responsabilidade do usuário manter o sistema operacional e demais softwares necessários conforme seus interesses.

Tabela 2.1: Atribuição de recursos nos modelos de serviço em nuvem

Modelo	Software	Sistema Operacional	Hardware
SaaS	CSP	CSP	CSP
PaaS	Cliente	CSP	CSP
IaaS5	Cliente	Cliente	CSP

2.2.4 Modelos de implantação

Quanto ao modo de acesso e disponibilidade dos ambientes de computação em nuvem têm-se quatro modelos de implantação. O modelo escolhido leva em consideração o processo do negócio e o tipo de informação acessada, pois em determinados casos deseja-se manter certos recursos restritos onde somente alguns usuários podem utilizar os serviços providos. Os modelos de implantação da computação em nuvem podem ser divididos em quatro categorias: pública, privada, comunitária e híbrida (Mell and Grance, 2009).

***Nuvem privada*:** a infraestrutura de nuvem tem seu uso restrito a uma única organização, ela emula alguns recursos da computação em nuvem, como a virtualização, mas o faz em uma rede privada (Kepes, 2011). Este modelo oferece um nível maior de segurança e controle, mas exigem que a empresa ainda adquira e mantenha toda a infraestrutura e software necessários (el Khameesy and Rahman, 2012);

***Nuvem pública*:** a infraestrutura de nuvens é disponibilizada para o público em geral, sendo acessado através da internet por qualquer usuário que conheça a localização do serviço. Estas nuvens oferecem um alto nível de eficiência nos recursos compartilhados, no entanto são mais vulneráveis que as nuvens privadas (el Khameesy and Rahman, 2012);

***Nuvem comunitária*:** neste modelo ocorre o compartilhamento da nuvem entre diversas organizações, sendo essa suportada por uma comunidade específica que partilhou seus interesses. Essa nuvem pode existir localmente ou remotamente;

***Nuvem híbrida*:** ocorre uma composição de dois ou mais modelos de nuvens, que podem ser privadas, públicas ou comunitárias. Nesse modelo há a necessidade de uma tecnologia que permita a portabilidade de dados e aplicações entre as nuvens.

2.3 Sistema Operacional de Nuvem

Uma funcionalidade fundamental da computação em nuvem é o fornecimento de infraestrutura computacional em forma de serviço, através da oferta de máquinas virtuais aos usuários. Para que o centro de dados garanta uma operação segura, eficiente e escalável dessa infraestrutura virtualizada é preciso uma plataforma que possibilite o gerenciamento da infraestrutura virtual. O principal papel desse componente é possibilitar a entrega de infraestrutura como um serviço (*IaaS*), transformando o centro de dados tradicional em uma arquitetura em nuvem (Buyya et al., 2009).

O gerenciador da infraestrutura virtual, também chamado de *Sistema Operacional de Nuvem* (do inglês, *cloud OS*) organiza a implantação de recursos virtuais e gere as infraestruturas físicas e virtuais, permitindo também o complemento da infraestrutura local com recursos remotos de outros centros de dados federados ou nuvens comerciais.

Essa visão de nuvem do centro de dados traz várias vantagens, incluindo:

- Consolidação de servidores, para redução dos requisitos de *hardware* e de energia;
- Redimensionamento dinâmico da infraestrutura física;
- Balanceamento da carga de trabalho entre os recursos físicos para melhor utilização dos recursos e eficiência;
- Replicação de servidores, para suporte as capacidades de tolerância a falhas e alta disponibilidade.
- Particionamento dinâmico da infraestrutura física para isolamento e execução de diferentes serviços e cargas de trabalho.

Nesse sentido, o Sistema Operacional de nuvem se torna um componente chave, gerenciando a infraestrutura física e virtual e controlando o provisionamento dos recursos virtuais de acordo com as necessidades dos serviços de usuários. Seu papel é gerenciar de uma forma eficiente os recursos do centro de dados de modo a oferecer um ambiente de execução multi-inquilinos flexível, seguro e isolado para os serviços do usuário, que abstraia a infraestrurura física subjacente e ofereça diferentes interfaces e APIs para interação com a nuvem (Moreno-Vozmediano et al., 2012).

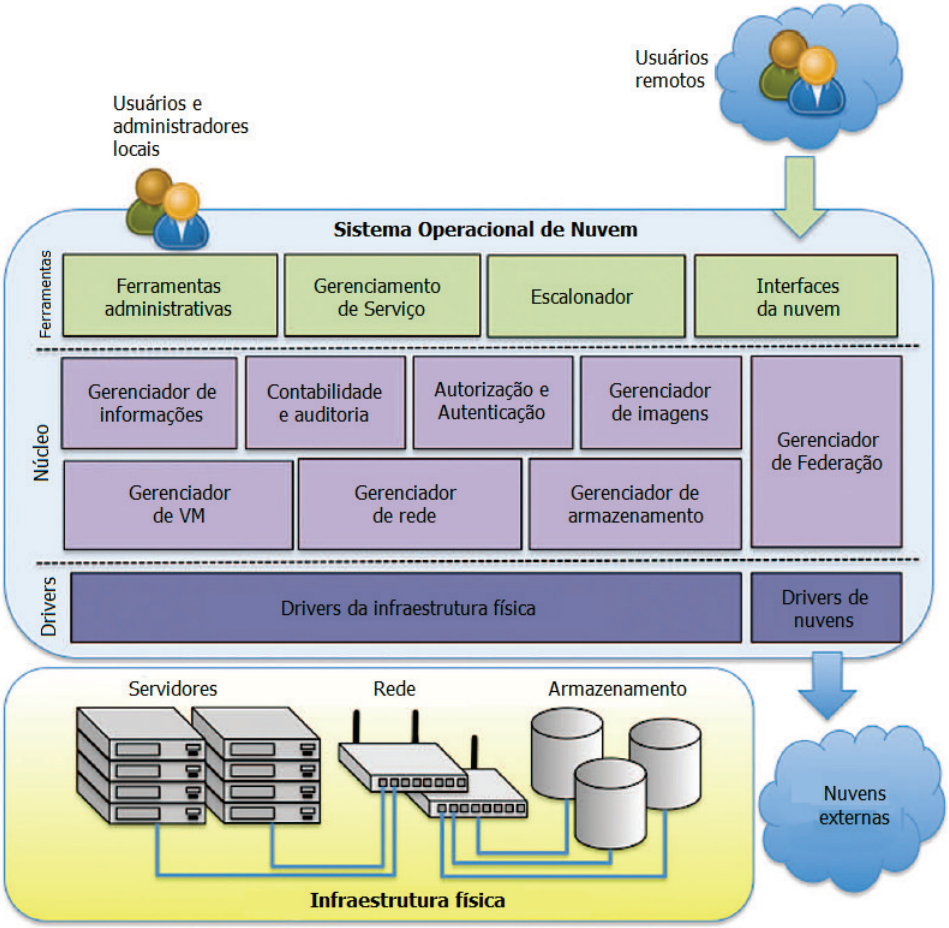


Figura 2.2: Sistema Operacional de Nuvem – *Cloud OS*
Fonte: Adaptado de (Moreno-Vozmediano et al., 2012)

A Figura 2.2 apresenta o Sistema Operacional de Nuvem organizado em três camadas distintas: *drivers*, núcleo e ferramentas de alto nível.

Na primeira camada encontram-se os *drivers* da infraestrutura física. Através deles é possível interagir com uma variedade de tecnologias de virtualização, incluindo hipervisores, rede e armazenamento, fornecendo uma abstração da tecnologia subjacente. Adicionalmente podem ser incluídos diferentes *drivers* de nuvens que permitem o acesso a provedores remotos.

A segunda camada é formada pelos componentes do núcleo do SO de nuvem, contendo os serviços de gerenciamento dos diversos elementos da infraestrutura física e virtual:

Gerenciador de máquinas virtuais: é o responsável pelo gerenciamento de todo o ciclo de vida e pela execução de diferentes ações sobre a VM, tais como implantar, migrar, suspender, retomar, encerrar – de acordo com comandos do usuário ou estratégias de escalonamento. Também é o responsável por preservar os acordos de nível de serviço contratados pelo usuário, expressos em termos de disponibilidade da VM, incluindo mecanismos para detecção de erros e reinicialização automática em caso de falha.

Gerenciador de rede: a implantação de serviços na nuvem, além da provisão de VMs, envolve também a instanciação de redes de comunicação para interligar os diferentes componentes

de serviço e para tornar o serviço acessível para os usuários externos, se necessário. É de sua responsabilidade gerir as redes privadas para interconectar os componentes internos com os endereços IP públicos e conectar à internet os serviços de *front-end*.

Gerenciador de armazenamento: sua função principal é prover serviços de armazenamento e sistemas de armazenamento virtual para o usuário final. Deve ser escalável, permitindo crescer dinamicamente de acordo com as necessidades do serviço; altamente disponível e confiável, para evitar problemas no acesso aos dados em caso de falha; alta performance para suportar altas demandas; fácil de gerenciar, abstraindo os usuários da complexidade física do hardware subjacente.

Gerenciador de imagens: deve manipular grandes quantidades de imagens de máquinas virtuais[1] pertencentes a diferentes usuários, com diferentes sistemas operacionais e configurações de *software*. Assim, deve possuir ferramentas apropriadas para gerenciamento destas imagens de forma eficiente e segura bem como deve ter funcionalidades adicionais para a administração do repositório de imagens. Um conjunto de atributos definem uma imagem de VM, incluindo seu nome, descrição de conteúdo, tipo da imagem – pública, privada, compartilhada – o proprietário e a localização da imagem no repositório. As funcionalidades básicas são: criar uma nova imagem no repositório, excluir uma imagem, clonar uma imagem a partir de uma já existente, adicionar ou modificar atributos, compartilhar uma imagem com outros usuários, publicar uma imagem para uso público ou listar as imagens disponíveis no repositório.

Gerenciador de informações: é responsável pelo monitoramento e coleta de informações sobre o estado das VMs, servidores físicos e outros componentes da infraestrutura física e virtual como dispositivos de rede e sistemas de armazenamento. Esse monitoramento é essencial para garantir que todos os componentes estejam executando de forma otimizada.

Autorização e autenticação: as nuvens devem incorporar mecanismos para autenticação de usuários e administradores, provendo-os com acesso apenas aos recursos autorizados. A autenticação de usuário verifica e confirma a identidade do usuário que está tentando acessar recursos na nuvem, sendo implementada através de diferentes métodos: mecanismos de verificação de senhas via LDAP, mecanismos de autenticação de confiança baseados em chave pública, certificados X.509 ou *Kerberos*[2]. As políticas de autorização controlam e gerenciam os privilégios e permissões do usuário para acessar diferentes recursos da nuvem, como VMs, redes ou sistemas de armazenamento. Adicionalmente, podem ser usados mecanismos de cota para limitar a quantidade de recursos – CPU, memória, banda de rede ou disco – que um usuário específico pode acessar.

Contabilidade e auditoria: o objetivo da contabilidade é obter e registrar as informações sobre o uso de recursos dos serviços implantados, sendo essencial para a implementação dos mecanismos que produzem as informações de faturamento. A auditoria fornece informações sobre as atividades nos recursos da nuvem, indicando quem acessou, quando o

[1] Uma imagem de máquina virtual é um arquivo único que contém um disco virtual com um sistema operacional auto-inicializável instalado.

[2] *Kerberos* é o nome de um protocolo de rede que permite comunicações individuais seguras e identificadas, em uma rede insegura, fornece autenticação em aplicações usuários/servidor, funcionando como a terceira parte neste processo.

acesso foi conseguido e quais operações foram executadas. Essas informações são úteis para melhorar a segurança da nuvem e protegê-la de ameaças como acesso não autorizado, uso abusivo de recursos e outras formas de intrusão.

Gerenciador de federação: permite o acesso a infraestruturas de nuvem remotas, que podem ser tanto infraestruturas parceiras ou provedores públicos de nuvem. Deve fornecer mecanismos básicos para implantação, gerenciamento em tempo de execução e término de recursos virtuais em nuvens remotas e também o monitoramento de recursos, autenticação de usuários, gerenciamento de controle de acesso e permissões de recursos e ferramentas para a construção de imagens em diferentes nuvens com diferentes formatos de imagem.

Tabela 2.2: Critérios de otimização para políticas de alocação/realocação

Critério	Objetivos	Política de alocação	Política de realocação
Consolidação de servidores	Redução do número de servidores em uso para minimizar o consumo de energia	VMs devem ser alocadas usando um número mínimo de servidores	VMs podem ser dinamicamente realocadas para reduzir o número de servidores em uso
Balanceamento de carga	Balanceamento da carga de trabalho dos servidores para evitar saturação e queda de desempenho	VMs devem ser distribuídas uniformemente entre os servidores disponíveis	Uma VM pode ser dinamicamente realocada para balancear a distribuição das VMs entre os servidores
Balanceamento de CPU	Balanceamento do uso das CPUs para evitar saturação e queda de desempenho	Uma nova VM deve ser alocada no servidor com a maior quantidade de CPU disponível	Em caso de saturação do servidor (sobrecarga de CPU), uma VM pode ser dinamicamente realocada em um servidor menos carregado
Balanceamento térmico	Balanceamento da temperatura de todos os servidores para evitar superaquecimento e reduzir a necessidade de resfriamento	Uma nova VM deve ser alocada no servidor que apresente a menor temperatura	Em caso de superaquecimento do servidor, uma VM pode ser dinamicamente realocada para um servidor mais frio

Na camada do topo encontram-se as ferramentas e interfaces (linha de comando ou GUI) que o sistemas operacional de nuvem deve fornecer aos usuários e administradores para executarem suas tarefas:

Escalonador: há dois níveis de escalonamento dentro de uma infraestrutura de nuvem: o nível do *host* físico, gerenciado pelo escalonador do hipervisor, o qual é responsável por decidir quando as VMs podem obter os recursos do sistema – como CPU e memória – e quais servidores físicos são designados a cada VM; e o nível da nuvem, gerenciado pelo escalonador do sistema operacional de nuvem, o qual é responsável por decidir o servidor físico específico em que cada VM é implantada.

A principal função do escalonador é decidir a localização inicial de cada VM seguindo critérios específicos. Também pode decidir implantar a VM em uma nuvem remota quando os recursos disponíveis na infraestrutura local são insuficientes. Além disso o escalonador pode fornecer a capacidade de otimização dinâmica, permitindo a realocação dinâmica (migração) de VMs de um recurso físico para outro para cumprir os critérios de otimização especificadas. A tabela 2.2 apresenta diferentes políticas de escalonamento, com base em diferentes critérios de otimização, para orientar tanto a colocação inicial e as ações de realocação dinâmica.

O usuário também pode especificar as restrições que afetarão as decisões do escalonador, como por exemplo:

- *Hardware*: quantidade de CPU, memória e assim por diante;

- Plataforma: tipo de hipervisor, Sistema Operacional, etc;
- Afinidade: duas ou mais VMs que precisam ser implantadas no mesmo servidor ou *cluster* físico;
- Localização: restrições geográficas;
- Acordo do nível de serviço – SLA (do inglês, *Service Level Agreement*): garantia de capacidade de CPU ou alta confiabilidade operacional.

Ferramentas administrativas: para o nível privilegiado de administração o sistema deve incluir tanto ferramentas para administração de usuários (criar, modificar ou excluir usuários e gerenciar as autorizações e políticas de controle de acesso) e de gerenciamento da infraestrutura física (ligar e desligar servidores físicos, monitorar a infraestrutura física, e assim por diante). Para usuários sem privilégios devem ser fornecidas ferramentas para o gerenciamento de sua própria infraestrutura: ferramentas de gerenciamento de VMs (implantar, desligar, suspender, restaurar ou monitorar uma VM), gerenciamento de rede virtual (criar ou excluir redes virtuais), gerenciamento de armazenamento virtual (criar, excluir ou anexar um disco virtual) e gerenciamento de imagens (criar, clonar ou excluir imagens).

Gerenciamento de serviço: o sistema operacional de nuvem deve ser capaz de gerenciar e suportar serviços virtualizados multicamadas. Um serviço multicamada pode incluir vários componentes/camadas com algumas dependências intrínsecas entre elas. Estes serviços podem ser implementados como um grupo de VMs interligadas na nuvem com dependências específicas de implantação e, opcionalmente, algum localização, afinidade e requisitos de elasticidade.

A função de controle de admissão implica decidir se aceita ou rejeita um serviço, em função das necessidades de serviço e disponibilidade de recursos na nuvem. Uma vez que o serviço é aceito, o gerenciamento de serviço é responsável pelo gerenciamento do seu ciclo de vida, que envolve várias ações, incluindo a implantação, suspensão, continuação ou cancelamento do serviço. Para implantar um novo serviço, o gerenciador de serviço interage com o escalonador para decidir a melhor colocação para as várias VMs que compõem o serviço, de acordo com o critério de otimização selecionado e restrições do serviço.

Outra função é o gerenciamento do serviço de elasticidade. O gerenciador de serviços pode incorporar diferentes mecanismos para o serviço de auto-dimensionamento baseado nas regras de elasticidade, que desencadeiam a implantação de novas instâncias (escala horizontal) ou pelo redimensionamento das instâncias existentes (escala vertical) quando métricas de serviço especificadas pelo usuário excedam certos limites.

Independente do gerenciador de serviços, os usuários estão sempre autorizados a empregar as interfaces fornecidas pelas ferramentas administrativas ou a interface da nuvem para implantar, redimensionar, migrar ou desligar suas VMs individuais.

Interfaces da nuvem: no atual *ecossistema* de nuvem, a maioria dos produtos e provedores de nuvem oferecem suas próprias APIs, tais como *Amazon EC2* ou VMware's vCloud. Embora algumas dessas APIs estejam ser tornando um padrão de fato, essa heterogeneidade torna difícil alcançar a interoperabilidade e portabilidade entre nuvens. Vários órgãos de

padronização estão abordando questões de interoperabilidade e portabilidade em torno das infraestruturas de nuvem. Por exemplo: OGF OCCI (*Open Grid Forum – Open Cloud Computing Interface*), OVF (*Open Virtualization Format*)

Muitos ferramentas que atuam como um sistema operacional de nuvem estão disponíveis para a criação de ambientes para o desenvolvimento e gerenciamento de uma infraestrutura em nuvem. *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack* são exemplos de implementações *open source* disponíveis ao público em geral, enquanto *VMWare vSphere* é um exemplo de um sistema proprietário.

2.3.1 Eucalyptus

Eucalyptus (*Elastic Utility Computing Architeture for Linking Your Programs to Useful Systems*) é um dos pacotes pioneiros e mais populares para construção de infraestruturas privadas e híbridas de computação em nuvem (Eucalyptus, 2014). O projeto teve início na Universidade Santa Bárbara na Califórnia e atualmente é mantido pela Eucalyptus Systems Inc.

Eucalyptus é uma implementação *open-source* da Amazon EC2 (*Elastic Computing Cloud*) que une a infraestrutura virtualizada existente para criar recursos de nuvem para computação, rede e armazenamento. É altamente escalável podendo dimensionar as necessidades de recursos de acordo com a carga de trabalho. Oferece compatibilidade com as APIs AWS (*Amazon Web Services*), incluindo a EC2 (*Elastic Computing Cloud*), S3 (*Simple Storage Service*), EBS (*Elastic Block Store*) e IAM (*Identity and Access Management*).

A Figura 2.3 apresenta sua arquitetura básica, que é composta por seis componentes distintos, agrupados em três níveis diferentes.

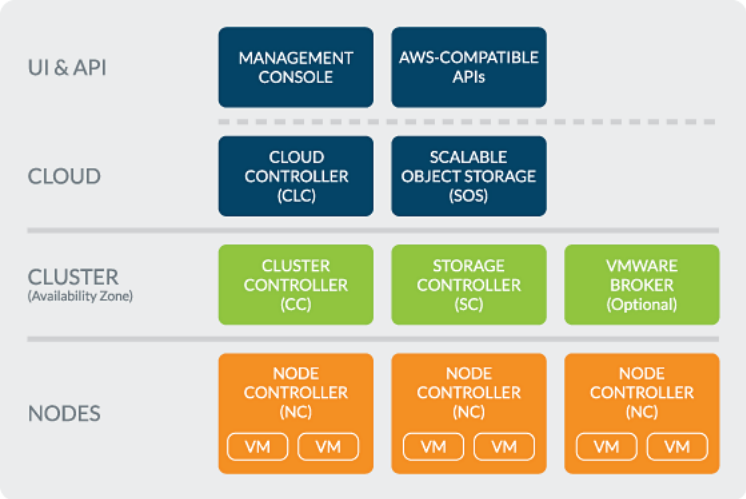


Figura 2.3: Arquitetura Eucalyptus
Fonte: (Eucalyptus, 2014)

O nível da nuvem (*Cloud level*) apresenta os componentes *Cloud-Controller (CLC)*, responsável pelas interfaces com o mundo exterior e o gerenciamento da nuvem (autenticação, contabilidade, relatórios, controle de quotas) e o *Scalable Object Storage (SOS)*, um serviço de armazenamento inter-conectável e flexível.

O *nível de cluster* é formado pelos componentes *Cluster Controller (CC)* que gerencia a execução de VMs e SLA por clusters, *Storage Controller (SC)* que gerencia volumes e *snapshots* para as instâncias de um cluster específico. Opcionalmente, o componente *VMware Broker* oferece interfaces ao ambiente do hipervisor *VMware*.

Finalmente, no nível de nós da arquitetura está o *Node Controller (NC)*, que hospeda as instâncias de VMs e gerencia os terminais de redes virtuais.

2.3.2 Nimbus

Nimbus é um conjunto de ferramentas *open-source* focada no fornecimento de recursos de infraestrutura (IaaS) para a comunidade científica, mas suporta também aplicações não-científicas (Nimbus, 2014). Tem como foco três objetivos:

- Permitir aos provedores de serviço construir nuvens IaaS privadas ou comunitárias, através do serviço *Nimbus Workspace Service* que permite aos usuários a locação de recursos através da implantação de máquinas virtuais. Complementarmente a ferramenta *Cumulus* fornece uma implementação de armazenamento em nuvem baseado em quotas.

- Permitir aos usuários utilizar nuvens IaaS de vários provedores, formando um ambiente *multi-cloud* que combina os recursos de nuvem pública e privada. A ferramenta *Nimbus Context Broker* cria um uma configuração comum e um contexto de segurança através de recursos compartilhados potencialmente em múltiplas nuvens. Devido à sua característica de interconectar múltiplas nuvens, essas ferramentas são conhecidas como ferramentas de *Sky Computing* (computação no céu).

- Permitir aos desenvolvedores estender, experimentar e personalizar nuvens IaaS, fornecendo uma implementação *open source* configurável. Um exemplo é o *Workspace Service*, que pode ser configurado para suportar diferentes implementações de virtualização (Xen ou KVM), interfaces (Amazon EC2), além de outras opções.

2.3.3 OpenNebula

OpenNebula é umas das principais tecnologias nas pesquisas de virtualização de infraestrutura e computação em nuvem na União Europeia (OpenNebula, 2014). É um projeto *open-source* que visa a construção de um padrão para a indústria com ferramentas para gerenciar a complexidade e heterogeneidade de infraestruturas grandes e distribuídas. Oferece recursos, meios flexíveis e melhor interoperabilidade para construir nuvens privadas, públicas ou híbridas.

É construído de forma a permitir uma gestão transparente de todos os recursos, tando locais como remotos. Coordena tecnologias de virtualização, armazenamento, redes, monitoração e segurança, permite a atribuição dinâmica de serviços multicamadas em infraestruturas distribuídas, podendo combinar recursos de centros de dados locais e nuvens remotas. A arquitetura do OpenNebula segue a estrutura apresentada na Figura 2.2.

A Figura 2.4 oferece uma visão geral das interfaces oferecidas pelo OpenNebula, através das quais é possível interagir com as funcionalidades oferecidas para o gerenciamento da infraestrutura física e virtual.

Para os consumidores da nuvem (*Cloud Consumers*) são oferecidas interfaces como OCCI, EC2 e EBS e uma visão da nuvem através da ferramenta *Sunstone*, que pode ser usado

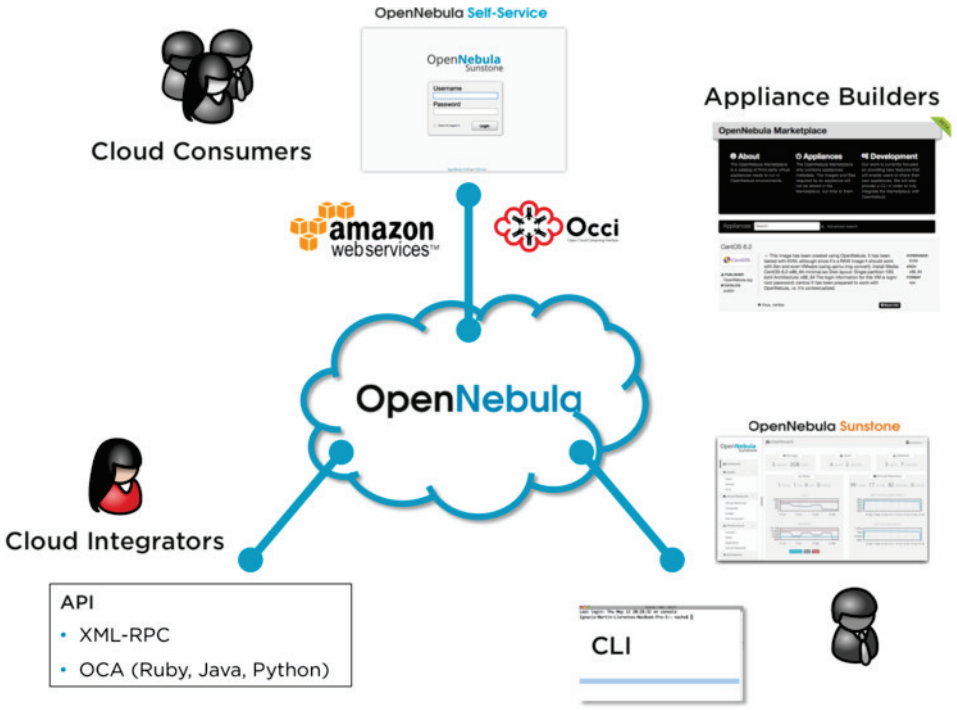


Figura 2.4: OpenNebula: visão geral das interfaces
Fonte: (OpenNebula, 2014)

como um portal de auto-atendimento. Para usuários avançados e administradores são fornecidas interfaces em linha de comando e a interface gráfica *Sunstone*. Para integradores (*Cloud Integrators*) oferece uma API extensível, de baixo nível, em Ruby, Java e XMLRPC. E através do *OpenNebula Marketplace* é disponibilizado um catálogo de dispositivos virtuais prontos para funcionar em ambientes OpenNebula.

2.3.4 OpenStack

OpenStack é um conjunto de projetos de software *open-source* que empresas e provedores de serviços podem usar para configurar e operar sua infraestrutura de computação e armazenamento em nuvem (OpenStack, 2014).

Foi fundado pela *Rackspace Hosting* em conjunto com a NASA (*National Aeronautics and Space Administration*) e atualmente o projeto visa a construção de uma comunidade com pesquisadores, desenvolvedores e empresas, que compartilham um objetivo comum de criar uma nuvem simples de implementar, altamente escalável e com recursos avançados.

OpenStack é um sistema operacional de nuvem que controla grandes *pools* de computação, armazenamento e recursos de rede em um centro de dados, gerenciado através de um painel que permite a provisão de recursos pelos usuários.

Uma visão geral do OpenStack é apresentada na Figura 2.5, onde se observa os serviços compartilhados de computação, rede e armazenamento implantados sobre um *hardware* padrão. A interação com esses serviços é realizada pelas aplicações, através de um conjunto de APIs, ou então diretamente pelo *Openstack Dashboard*, um painel para gerenciamento da nuvem.

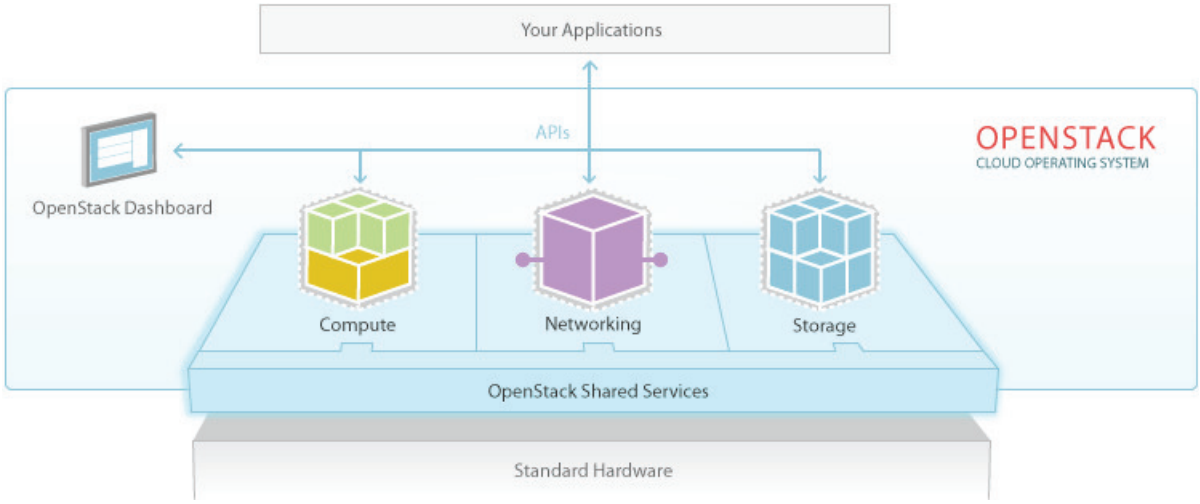


Figura 2.5: OpenStack: visão geral
Fonte: (OpenStack, 2014)

2.4 Computação em nuvem verde

A Computação em Nuvem "Verde"(*Green Cloud*) se refere ao uso eficiente de recursos computacionais minimizando o impacto ambiental, maximizando sua viabilidade econômica e assegurando os deveres sociais. Em geral não difere muito da computação em nuvem, porém ela traz uma preocupação a mais sobre a estrutura: consumir menos energia sem interferir na performance, garantindo a sustentabilidade (Buyya et al., 2010).

O termo surgiu em um momento de expansão da adoção do paradigma da Computação nas Nuvens. Com a alta procura por estes serviços, surgiram novos centros de dados, e os já existentes se expandiram, levando a um aumento no consumo de energia elétrica. Fez-se necessário então a ampliação de suas fontes de fornecimento de energia elétrica, resultando em intervenções que causam impacto no meio ambiente, como a construção de usinas hidrelétricas e termoelétricas (Garg and Buyya, 2012).

Um centro de dados de Computação em Nuvem, devido a sua arquitetura, já traz uma economia de energia se comparada a um centro de dados tradicional. Essa economia é obtida devido a utilização da técnica de virtualização, onde é possível realizar uma maior consolidação nos servidores, reduzindo assim a subutilização dos recursos. A Figura 2.6 apresenta uma visão geral da computação em nuvem verde.

A importância do fator energético é tão grande para os centros de dados, que estes muitas vezes são instalados próximos das fontes geradoras de energia, de modo a conseguir preços mais baixos. Com isso, governos e entidades não governamentais, como o *Greenpeace*, pressionam os provedores de serviço para adotarem métodos que reduzam o consumo de energia dos centros de dados.

Nesse sentido, há algumas iniciativas da indústria visando o desenvolvimento de métodos e técnicas padronizadas para a redução do consumo de energia nos ambientes computacionais. Entre elas destacam-se: *Climate Savers Computing Initiative (CSCI)*, *Green Computing Impact Organization, Inc. (GCIO)*, *Gren Eletronics Council*, *The Green Grid*, *International Professional Practice Partnership (IP3)*, tendo como membros grandes companhias como AMD, Dell, HP, IBM, Microsoft, Sun Microsystems e VMWare (Beloglazov, 2013).

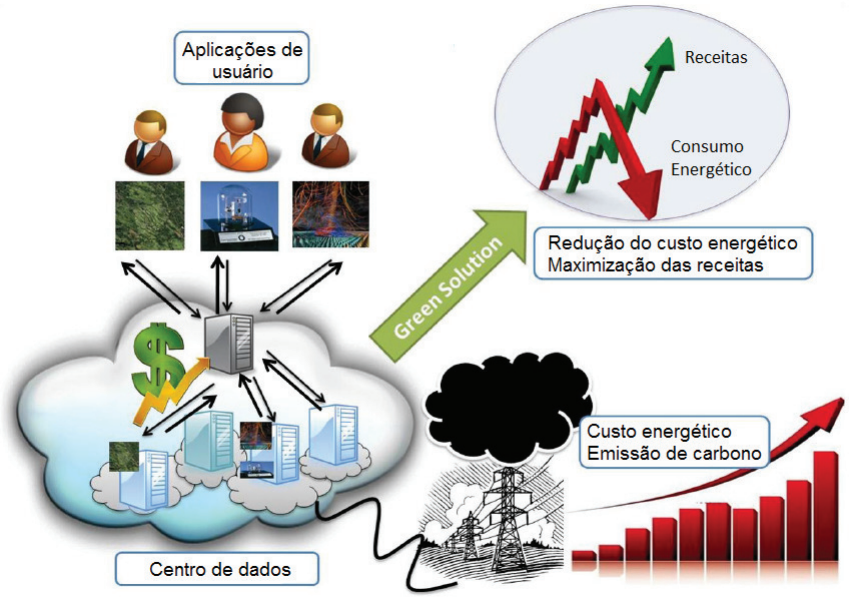


Figura 2.6: Computação em Nuvem Verde
Fonte: Adaptado de Garg and Buyya (2012)

Na literatura é possível encontrar uma grande variedade de trabalhos que tratam da questão energética em um centro de dados, como as apresentadas por (Beloglazov and Buyya, 2010), (Dhiman et al., 2010), (Lago et al., 2011), (Azevedo et al., 2011), (Beloglazov, 2013). Tais soluções trazem benefícios tanto para o meio ambiente quanto para os próprios provedores de serviço, em virtude da diminuição dos custos com energia.

2.5 Virtualização

Apesar da Computação em Nuvem ser um paradigma recente ela é baseada em conceitos e tecnologias bem conhecidas há décadas. Segundo (Zhang et al., 2010) trata-se de um novo modelo de operações que reúne um conjunto de tecnologias existentes para executar o negócio de uma maneira diferente. Entre essas tecnologias está a virtualização, que se tornou uma tecnologia chave para a Computação em Nuvem (Vaquero et al., 2009), (Paul et al., 2012).

A virtualização é uma metodologia que divide os recursos computacionais em múltiplas execuções, criando múltiplas partições, isoladas uma das outras, chamadas máquinas virtuais (VM, do inglês *Virtual Machines*), unidas em um único servidor físico. Uma máquina virtual funciona como uma cópia exata de uma máquina real, incluindo seus recursos de hardware, com a diferença que trabalha com esses recursos de forma isolada.

A virtualização surgiu na década de 60, no IBM S/370, onde a IBM fez um modelo em que cada máquina virtual era uma cópia exata de uma máquina real, porém com uma capacidade de memória reduzida, assim um computador poderia ser dividido em várias máquinas virtuais leves, utilizando recursos tanto quanto o original (Creasy, 1981).

Um ambiente virtualizado consiste basicamente de três partes (Smith and Nair, 2005):

- Sistema hospedeiro (*host system*) ou sistema real que contém os recursos reais de hardware e software do sistema.

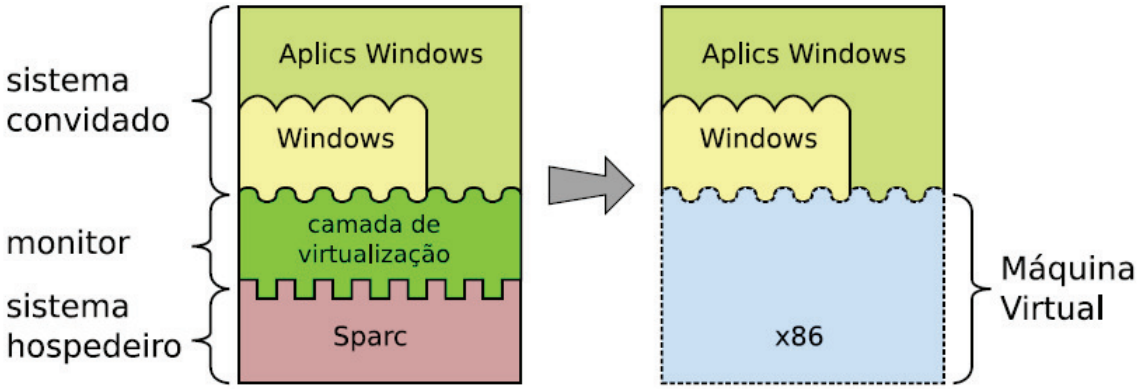


Figura 2.7: Exemplo de uma máquina virtual
Fonte: (Laureano and Maziero, 2008)

- Sistema convidado (*guest system*) ou sistema virtual, que executa sobre o sistema virtualizado. Podem existir vários sistemas virtuais executando sobre um mesmo sistema hospedeiro.

- A camada de virtualização ou *monitor de máquina virtual* (do inglês *Virtual Machine Monitor – VMM*), também chamado de *hipervisor*, que é o responsável pela construção das interfaces virtuais a partir da interface real.

Na Figura 2.7 podem ser observadas essas três partes, ela apresenta uma máquina virtual em que uma camada de virtualização (hipervisor) permite executar um sistema convidado (Windows e suas aplicações) sobre um sistema hospedeiro (Sparc) que apresenta uma plataforma de hardware distinta daquela para a qual esse sistema foi projetado (Intel/AMD).

É possível considerar o hipervisor como o elemento fundamental da virtualização, uma das suas principais funcionalidades consiste em abstrair os recursos da máquina real e oferecê-los às máquinas virtuais. Assim, cada máquina virtual "enxerga" sua própria interface de rede, seus discos rígidos, seus processadores e suas áreas de memória RAM. Esses recursos virtuais usados pelas máquinas virtuais são mapeados pelo hipervisor nos recursos reais presentes no hardware subjacente, de forma eficiente e controlada. Pode-se afirmar que o hipervisor constitui um "sistema operacional para sistemas operacionais", pela flexibilidade que introduz na gestão dos recursos de hardware. A possibilidade de instanciar máquinas virtuais sob demanda e a relativa independência do *hardware* são essenciais para os ambientes de computação em nuvem.

2.5.1 Arquitetura de máquinas virtuais

Existem vários tipos de máquinas virtuais, com diferentes objetivos e implementações, mas de um modo geral elas podem ser divididas em duas grande famílias (Smith and Nair, 2005):

- *Máquinas virtuais de aplicação* (*Process Virtual Machines*): são ambientes de máquinas virtuais destinados a suportar apenas um processo ou aplicação convidada específica. A máquina virtual Java é um exemplo desse ambiente. Ela é criada sob demanda, no momento do lançamento da aplicação convidada, e destruída quando a aplicação finaliza a sua execução. O conjunto *hipervisor* + *aplicação* é visto então como um único processo

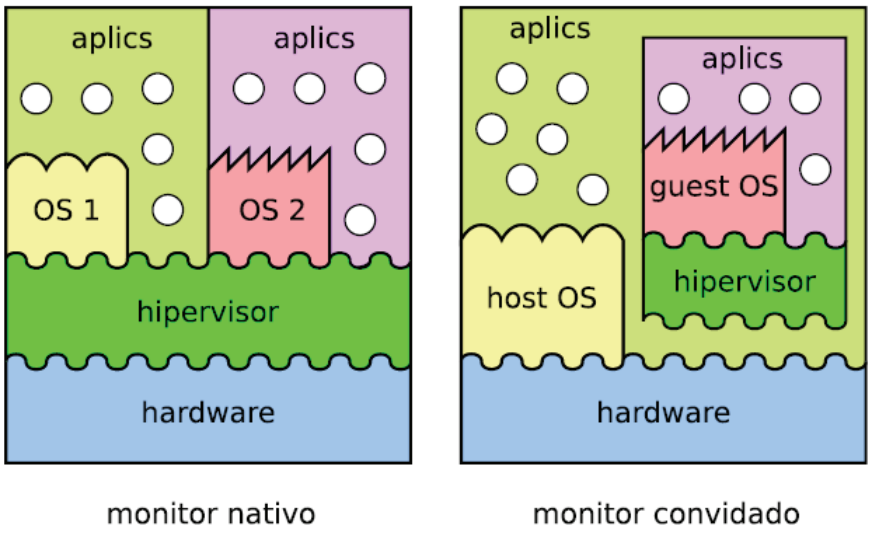


Figura 2.8: Arquiteturas de máquinas virtuais de sistema
Fonte: (Laureano and Maziero, 2008)

dentro do sistema operacional subjacente, submetidas às mesmas condições e restrições que os demais processos nativos.

- *Máquinas virtuais de sistema* (*System Virtual Machines*): são ambientes de máquinas virtuais construídas para suportar sistemas operacionais convidados completos, com aplicações convidadas executando sobre elas. Como exemplos temos os ambientes *VMWare*, *Xen*, *KVM* e *VirtualBox*.

Em relação as máquinas virtuais de sistema, cada sistema operacional convidado tem a ilusão de executar sozinho sobre uma plataforma de hardware próprio, os sistemas operacionais convidados são fortemente isolados um dos outros, podendo interagir apenas através de mecanismos de rede, como se estivessem em máquinas físicas separadas.

No que diz respeito à arquitetura, existem basicamente dois tipos de hipervisores de sistema, apresentados na Figura 2.8:

- *Hipervisores nativos (tipo I)*: o hipervisor executa diretamente sobre o hardware da máquina real, sem um sistema operacional subjacente. Os recursos de hardware (memória, discos, interfaces de rede, etc) são multiplexados para que as máquinas virtuais vejam um conjunto de recursos próprios e independente. É a forma de virtualização mais antiga, encontrada nos sistemas computacionais de grande porte dos anos 1960-70. Alguns exemplos de sistemas que empregam esta abordagem são *VMware ESX Server* e o ambiente *Xen*.

- *Hipervisores convidados (tipo II)*: o hipervisor executa como um processo normal sobre um sistema operacional nativo subjacente, oferecendo os recursos oferecidos pelo sistema operacional nativo ao sistema convidado que executa sobre ele. o *VMware Workstation* e o *VirtualBox* são exemplos de sistemas que adotam essa estrutura.

2.5.2 Migração de máquinas virtuais

No ambiente dinâmico de um centro de dados podem ser encontradas várias situações envolvendo VMs e servidores físicos:

- Uma VM hospedada em um servidor físico saturado pode necessitar de mais recursos do que está atualmente alocado para ela;
- Pode ser necessário realizar a manutenção de um servidor físico na qual VMs estejam alocadas;
- Servidores com baixa carga de trabalho podem ser desligados para economia de energia;
- Servidores saturados podem ter sua carga de trabalho diminuida para evitar problemas de superaquecimento.

Em todos esses casos, de acordo com critérios pré-estabelecidos, ocorre a necessidade de mover uma VM de um servidor físico para outro. A técnica de virtualização que permite realizar esse procedimento é conhecido como *migração de VMs*.

A migração deve ser transparente ao sistema operacional executado sobre a VM, às aplicações em execução no sistema operacional e aos clientes remotos da VM (Nelson et al., 2005). As soluções de virtualização tipicamente oferecem dois tipos de migração: *stop-and-copy* e *live migration* (Grit et al., 2006).

No tipo de migração *stop-and-copy*, a máquina virtual é suspensa no servidor de origem e seu *status* é salvo, então ela é copiada ao servidor de destino onde é reativada do ponto em que parou. É um processo com um tempo total de migração melhor pois a cópia da VM é realizada como uma cópia de arquivo (Garfinkel and Rosenblum, 2005). No entanto a máquina virtual torna-se indisponível durante esse período (*downtime*), o que pode ser indesejável para o cliente. Além disso, o SLA acordado entre provedor e cliente poderá ser violado, ocasionando penalidades ao provedor de serviços.

O tipo *live migration* aborda essa limitação fazendo com que a VM não seja interrompida durante o processo de migração, porém pode ocorrer uma degradação de performance durante esse período (Grit et al., 2006). Com relação aos aspectos temporais da migração, o tempo total de uma migração é medido no momento do início do processo de migração até o momento de conclusão da migração. Nesse intervalo, haverá um período de tempo que será necessário parar a máquina virtual para transferir as últimas porções de memória. Se comparada a abordagem *stop-and-copy*, esse tempo que a aplicação ficará indisponível pode ser considerada desprezível.

Nessa modalidade de migração existem algumas estratégias que são adotadas para manter a VM ativa enquanto a migração é realizada, duas delas são:

Migração *pre-copy* (pré-cópia): Neste tipo de migração, após selecionado o servidor que receberá a VM, as páginas de memória são transferidas para o destino enquanto a VM continua sendo executada na origem. De forma iterativa as páginas de memória vão sendo atualizadas no destino conforme ocorram modificações na origem. O controle então é transferido para o servidor destino e possíveis reconfigurações são executadas para deixar a VM ativa no novo servidor enquanto a VM do servidor antigo é eliminada. A Figura 2.9 apresenta o esquema de funcionamento da estratégia pré-cópia .

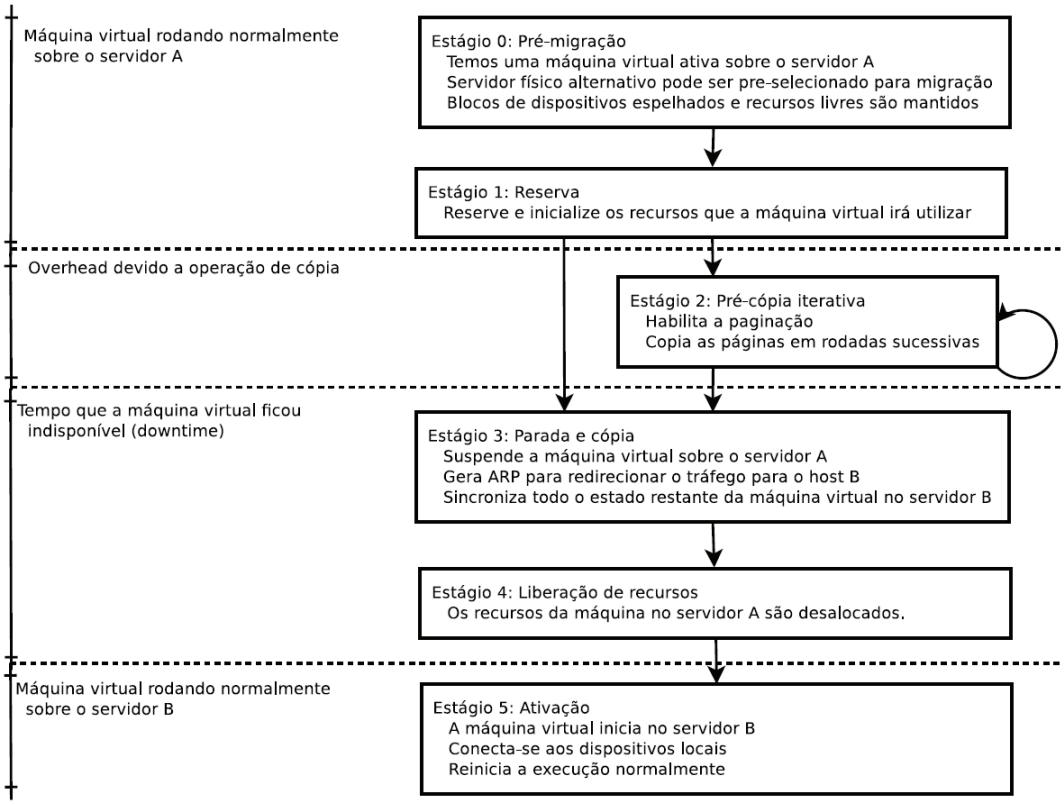


Figura 2.9: *Live migration* utilizando a abordagem *pre-copy*
Fonte: (Amarante, 2013)

Migração *post-copy*: Nesse caso, a VM é suspensa na origem e ativada no destino com uma configuração mínima, antes das transferências das páginas de memória. Após a ativação no destino as páginas são copiadas da origem. Esse processo pode causar falhas de páginas caso a página requisitada ainda não esteja disponível, nesse caso a busca por estas páginas na origem recebe uma prioridade maior. O esquema é mostrado na Figura 2.10

2.5.3 Compartilhamento de páginas de memória

Na computação em nuvem, através da tecnologia de virtualização, múltiplas VMs podem ser alocadas em um mesmo servidor físico e podem operar de forma independente uma das outras. Nesse ambiente virtualizado, os recursos físicos (como processador e memória principal) são gerenciados pelo hipervisor, que tem por objetivo proporcionar um compartilhamento eficiente desses recursos entre as máquinas virtuais em execução (Rosenblum and Garfinkel, 2005).

Dentre esses recursos, a memória é um dos principais limitadores para a consolidação de um maior número de VMs nos servidores físicos (Miller et al., 2013). Enquanto CPU e I/O podem ser compartilhados através da estratégia de divisão de tempo (*time sharing*), e podem ser escalonados por prioridades, a necessidade de memória de uma VM é algo mais desafiador. A medida que a demanda por memória aumenta, com a alocação de novas VMs e com a carga

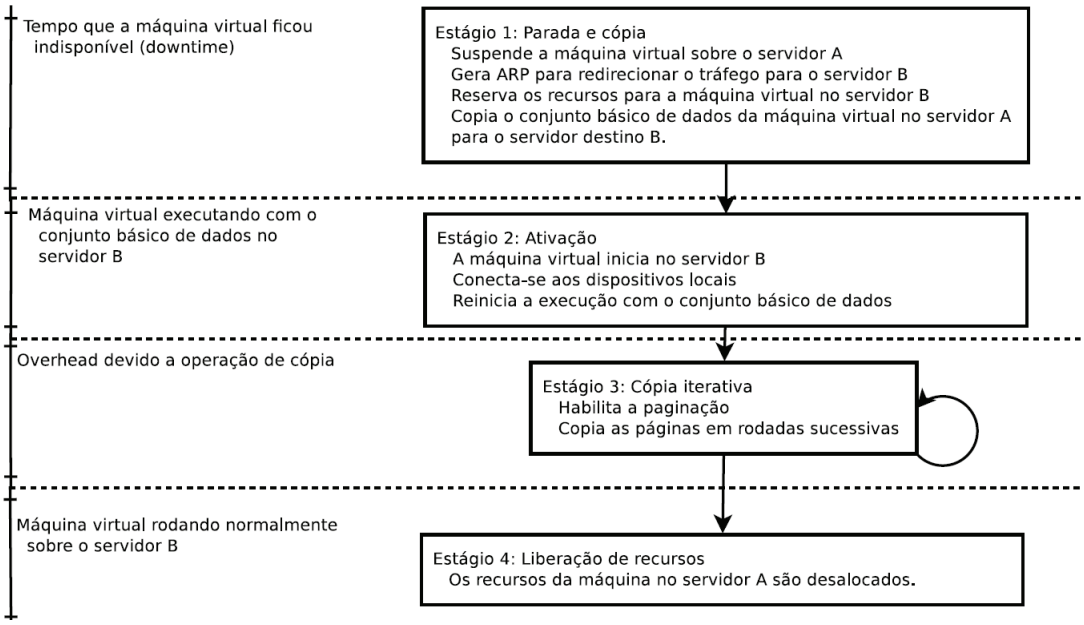


Figura 2.10: *Live migration* utilizando a abordagem *post-copy*
(Amarante, 2013)

de trabalho das aplicações nas VMs, a virtualização coloca grande pressão sobre o sistema de memória, impedindo novas alocações mesmo que outros recursos estejam disponíveis.

Redundância de dados podem ocorrer entre as VMs. Dependendo do sistema operacional e das aplicações suportadas pelas VMs, essa redundância pode ser maior ou menor. Mesmo em uma instância simples, uma quantidade significativa de páginas de memória podem possuir conteúdo idêntico. Em ambos os casos, o consumo de memória pode ser reduzido, mesclando-se as páginas redundantes em uma única cópia na memória física, através de uma técnica de *deduplicação de memória* conhecida como compartilhamento de página baseada em conteúdo (*CBPS*, do inglês *Content-Based Page Sharing*.

Nesse tipo de sistema o hipervisor, que possui uma visão global da memória física em todas as máquinas virtuais, identifica as páginas compartilháveis, ou seja, aquelas que possuam conteúdo idêntico, e então compartilha-as, sem qualquer participação ou colaboração das máquinas virtuais cujas páginas estão sendo compartilhadas. A Figura 2.11 ilustra esse processo, onde de seis páginas de memórias necessárias pelas máquinas virtuais foram necessárias quatro páginas físicas.

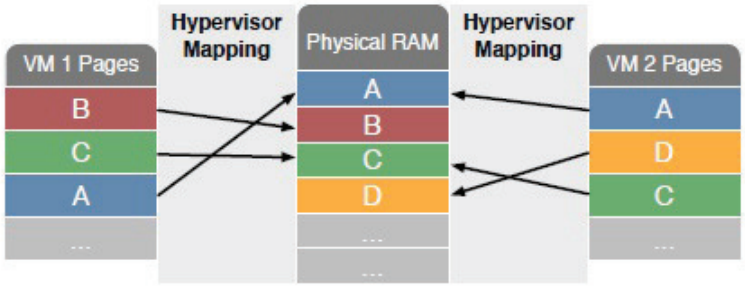


Figura 2.11: Compartilhamento de páginas entre duas VMs

Essa técnica está presente em hipervisores como *VMWare ESX*, no *Difference Engine* do *Xen VMM* e no *KSM - Kernel Samepage Merging* do *KVM*. Estudos apresentam ganhos significativos de economia de memória, a VMWare relata uma economia de memória em torno de 40% (Waldspurger, 2002) enquanto o *Difference Engine* calcula em torno de 50% de economia de memória (Gupta et al., 2010). Outros estudos, entretanto, apresentam resultados mais modestos, variando de 15% a 30% (Barker et al., 2012), dependendo principalmente das características relacionadas ao sistema operacional e aplicações instaladas nas VMs.

O mecanismo de compartilhamento de páginas de memória, de um modo geral, é semelhante entre os hipervisores: um serviço fica responsável por periodicamente analisar o conteúdo das páginas de memória, quando páginas idênticas são encontradas elas são compartilhadas usando a técnica *COW*[3].

No entanto, a forma de implementação varia de acordo com a técnica utilizada por cada hipervisor. O *CBPS* do *VMWare* realiza a comparação através de uma função *hash* para indexar o conteúdo de cada página, se o valor *hash* de uma página for encontrada mais de uma vez em diferentes VMs, então há uma boa probabilidade da página ser idêntica com aquelas de mesmo valor *hash*. Para certificar-se que as páginas são idênticas, uma comparação bit a bit é realizada. Caso as páginas sejam idênticas, elas são reduzidas para uma página usando *COW* (Kolster et al., 2006).

O *Differential Engine* do *Xen*, utiliza três mecanismos distintos que trabalham em conjunto para o compartilhamento de memória. Além de utilizar *COW* para páginas idênticas, ele também trata das páginas similares, mas não idênticas, nesse caso é armazenado um *patch* com as diferenças. Já as páginas que são únicas mas pouco acessadas, são comprimidas na memória para economia de espaço (Gupta et al., 2010).

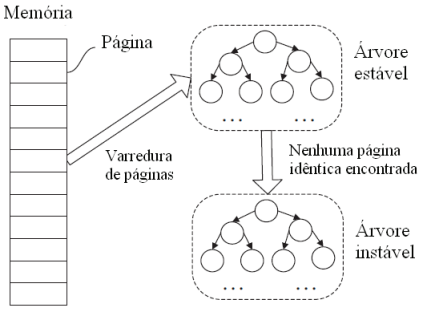


Figura 2.12: Árvore estável e árvore instável no KSM
Fonte: (Chen et al., 2014)

O *KSM* foi incluído no kernel do Linux 2.6.32 e foi desenvolvido para seu monitor de máquina virtual, o (*KVM*), mas não está limitado apenas a máquinas virtuais. O *KSM* faz o gerenciamento das páginas de memória utilizando para comparação duas árvores globais com todas as páginas de memória de um servidor: a árvore estável e a árvore instável, conforme apresentado na Figura 2.12. A árvore estável mantém as páginas compartilhadas enquanto a árvore instável mantém apenas as páginas que não foram compartilhadas (Arcangeli et al., 2009).

[3] *Copy-On-Write* é uma técnica que permite a duplicação de dados através de apontamentos ao dado original, evitando a necessidade de se manter cópias idênticas. Para evitar conflitos, as atualizações nos dados são interceptadas e a referência ao dado compartilhado é desfeita, sendo então criada uma nova cópia do dado.

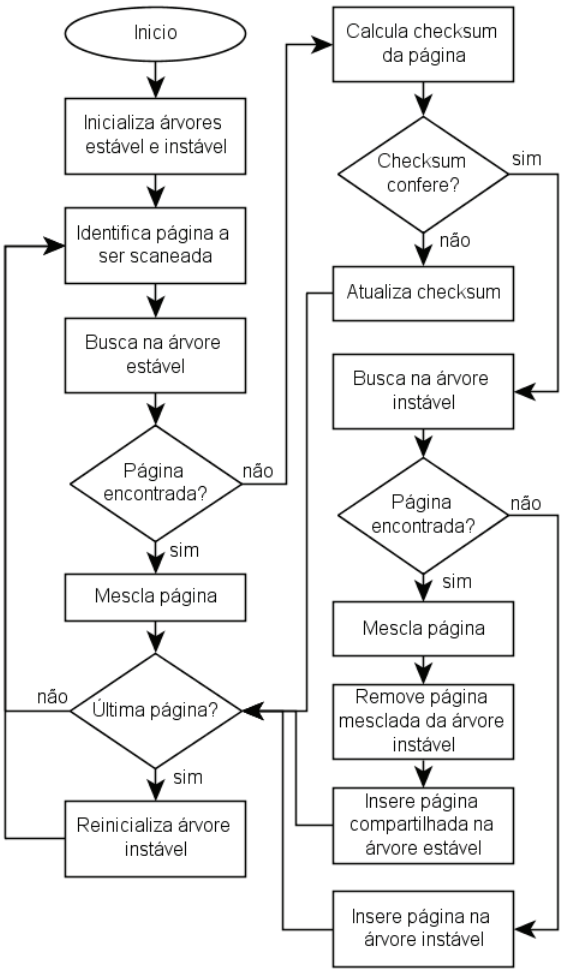


Figura 2.13: Fluxograma do algoritmo KSM
Fonte: Adaptado de (Arcangeli et al., 2009)

A cada varredura, uma página candidata é primeiramente comparada com as páginas na árvore estável. Caso exista alguma página idêntica, a página candidata é mesclada e compartilhada com a página encontrada. Caso contrário, é realizada uma busca na árvore instável: caso alguma página seja encontrada, ela é removida da árvore instável, mesclada com a página candidata e transferida para a árvore estável. Se nenhuma página for encontrada, a página candidata é inserida na árvore instável. O fluxograma deste algoritmo pode ser visto na Figura 2.13.

2.6 Conclusão

Este capítulo apresentou os principais conceitos envolvendo a Computação em Nuvem e um importante componente de sua arquitetura: o chamado Sistema Operacional de Nuvem. Também foi destacada a Computação em Nuvem Verde, uma abordagem que busca a eficiência energética no ambiente de nuvem.

Por ser parte fundamental da computação em nuvem o capítulo também tratou da virtualização e de algumas técnicas utilizadas nesse ambiente, como a migração de máquinas virtuais e o compartilhamento de memória entre VMs. Essas técnicas podem ser usadas para melhorar o gerenciamento da infraestrutura do centro de dados e são, portanto, fonte de interesse para diversas pesquisas.

O próximo capítulo trará uma visão geral sobre o problema de alocação de máquinas virtuais e apresentará algumas estratégias e algoritmos encontrados na literatura, realizando uma comparação entre eles.

Capítulo 3

Alocação de Máquinas Virtuais

Em um ambiente de computação em nuvem, os provedores de serviços, para suprir a demanda por recursos de seus clientes, alocam máquinas virtuais nos servidores de seus centros de dados. Para o centro de dados é importante que essa alocação seja realizada de forma rápida e eficiente, de modo a garantir um melhor uso da infraestrutura, contribuindo para a redução de custos com manutenção, energia e refrigeração, sem afetar o desempenho para os clientes. Este capítulo apresenta um estudo sobre a alocação de máquinas virtuais nos centros de dados, define-se o problema de alocação e as abordagens utilizadas para solucioná-lo. Por fim, são apresentados alguns algoritmos propostos na literatura e que relacionam-se com os objetivos desta pesquisa.

3.1 Introdução

No capítulo anterior, a fundamentação teórica abordou a computação em nuvem e citou a virtualização como peça-chave desse paradigma. Através da virtualização é possível dividir os recursos de um servidor físico entre múltiplas máquinas virtuais.

Neste capítulo é abordada a questão da alocação de máquinas virtuais nos centros de dados de computação em nuvem. A alocação de VMs consiste basicamente em selecionar um servidor físico para receber uma nova máquina virtual. Uma alocação adequada traz benefícios tanto para o centro de dados como para o cliente.

O restante do capítulo está organizado da seguinte maneira: a seção 3.2 trata do problema de alocação de máquinas virtuais, aborda os problema de alocação e realocação de VMs e apresenta alguns critérios utilizados no processo de escolha do servidor; a seção 3.3 aborda alguns problemas envolvendo a alocação de VM e faz uma comparação com alguns problemas clássicos da ciência da computação; a seção 3.4 apresenta algumas técnicas e algoritmos encontrados na literatura e finalmente a seção 3.5 apresenta um estudo comparativo entre esses algoritmos.

3.2 Alocação de Máquinas Virtuais

Em um ambiente de computação nas nuvens, os usuários enviam requisições ao centro de dados para alocação de recursos em seus servidores. Para atender a demanda dos clientes, o centro de dados aloca máquinas virtuais nos servidores físicos de sua infraestrutura. Nesse

cenário, o problema se concentra em determinar qual o servidor físico que receberá a máquina virtual.

Devido essa escolha influenciar diretamente no desempenho e eficiência do ambiente, tanto para o usuário quanto para o centro de dados, a alocação de máquinas virtuais torna-se uma importante questão na utilização da computação nas nuvens.

Diante disso, essa área apresenta muitos estudos relacionados com a melhoria dos mecanismos de alocação de máquinas virtuais, sob os mais diversos aspectos: otimização de desempenho, balanceamento de carga, consolidação de servidores ou eficiência energética.

3.2.1 O problema de alocação

Considerando um conjunto de servidores físicos em um centro de dados, a alocação de VMs consiste em, dada uma VM, encontrar um servidor que seja adequado para instanciá-la. O servidor deve possuir recursos suficientes para garantir a execução da VM sem afetar os acordos de nível de serviço.

Esse problema representa um desafio algorítmico por sua característica combinatória, onde um conjunto de $n \geq 1$ máquinas virtuais, com requisitos distintos de recursos (processamento, memória, disco, rede, etc) deve ser alocado em $m \geq 1$ servidores com diferentes disponibilidades de recursos.

De um modo geral, a alocação de uma VM pode ser dividida em duas etapas: inicialmente deve ser estimada a demanda por recursos da VM; em seguida, escolhe-se o servidor onde essa VM será alocada (Mishra and Sahoo, 2011).

A demanda de recursos de uma VM pode ser difícil de estimar, pois as necessidades de recursos podem variar durante sua execução. Uma abordagem comum é realizar a estimativa com base no histórico de utilização de recursos da VM. A presente pesquisa não trata a questão da estimativa de recursos e considera que as necessidades de VM são previamente conhecidas.

3.2.2 Critérios de alocação

O processo de alocação de VMs usa alguma estratégia para definir onde alocar a VM de modo a alcançar um nível eficiente de utilização dos recursos de um servidor físico. Tal estratégia pode utilizar representações matemáticas ou métricas que representem o grau de utilização dos recursos pelas diversas VMs e servidores físicos. Essas métricas devem levar em consideração as várias dimensões de consumo de recursos, como as capacidades de processador, memória, disco e banda de rede (Hyser et al., 2007), bem como recursos não envolvidos diretamente com as necessidades da máquina virtual, mas que afetam diretamente o centro de dados, como o consumo de energia decorrente da alocação (Buyya et al., 2010).

A seguir são apresentados alguns critérios que podem ser adotados no momento da escolha do servidor físico para realizar a alocação:

- Servidor com menor quantidade de recursos disponíveis: a alocação é realizada no servidor que possuir a menor quantidade de recursos não utilizados. Busca-se aqui a consolidação de servidores, ou seja, utilizar a menor quantidade de servidores possíveis.

- Servidor com maior quantidade de recursos disponíveis: a alocação é realizada no servidor que possuir a maior quantidade de recursos não utilizados. Assim, as máquinas virtuais

são alocadas de modo disperso, evitando a sobrecarga dos servidores e colaborando com o balanceamento da carga de trabalho.

- Aleatório: apenas é selecionado um servidor qualquer que possua recursos suficientes para alocar a máquina virtual.

- Menor consumo de energia: com esse critério busca-se o servidor onde o impacto energético da alocação da máquina virtual seja o menor possível.

Em relação ao consumo de energia nos centros de dados, os trabalhos nessa área podem ser divididos em três vertentes principais:

- Consumo da infraestrutura da rede: tentam concentrar as máquinas virtuais de modo a utilizar uma menor quantidade de equipamentos de rede, a fim de que os não utilizados sejam desligados. Para viabilizar a solução, é necessário a monitoração constante do tráfego de dados para a adequação dinâmica da localização das VMs de acordo com o tráfego atual (Beloglazov and Buyya, 2010)

- Consumo da infraestrutura de refrigeração: em um centro de dados, parte da energia consumida pela infraestrutura computacional é convertida em calor, que se não for adequadamente tratado, pode reduzir a durabilidade e confiabilidade dos dispositivos. Nesse caso, tenta-se realizar uma alocação das máquinas virtuais de modo que o calor seja tratado usando a menor quantidade de energia possível. Um modo de obter economia com a infraestrutura de resfriamento é baseada na localização otimizada de cargas de trabalho a fim de amenizar pontos de calor (*hotspots*) locais (Fakhim et al., 2011).

- Consumo dos servidores: busca alocar as máquinas virtuais nos servidores com o objetivo de utilizar a menor quantidade de energia, de um modo geral é buscado o servidor que apresente o menor incremento de energia ao alocar uma VM (Buyya et al., 2010).

3.2.3 Modo de chegada das máquinas virtuais

Como apresentado previamente, os algoritmos de alocação de VMs utilizam algum critério para o processo de decisão sobre qual servidor utilizar para instanciar uma VM. Além desses critérios, outro fator que influencia os algoritmos de alocação é a forma com que as VMs chegam ao sistema.

As VMs podem chegar de duas formas (Camati, 2013): em um fluxo contínuo, onde cada VM é tratada individualmente e a alocação realizada no momento em que a VM chega, nesse trabalho tratada como *alocação online*; ou em lotes de VMs, onde as VMs são previamente agrupadas e a alocação é realizada para todo o conjunto, aqui chamada de *alocação em lote*.

Algoritmos de alocação em lotes de VMs tem a vantagem de possuir o prévio conhecimento de todo o conjunto de VMs, dessa forma, otimizações podem ser feitas levando em consideração as características das VMs presentes no conjunto.

A presente pesquisa trata a chegada de máquinas virtuais na forma de fluxo contínuo, próximo da realidade dos centros de dados de computação em nuvem, que precisam responder rapidamente à solicitação de uma nova VM e deixá-la pronta para uso o mais rápido possível.

3.2.4 Realocação de máquinas virtuais

Como as cargas de trabalho das VMs frequentemente mudam com o tempo, não é suficiente apenas fazer boas escolhas iniciais de alocação das VMs, mas é necessário dinamicamente alterar as localizações das mesmas se as condições e políticas mudarem no centro de dados (Hyser et al., 2007).

A seguir são apresentadas algumas políticas e condições que podem levar a necessidade de realocação de máquinas virtuais. Alguns desses critérios foram apresentados no capítulo anterior, na tabela 2.2:

- Necessidade de manutenção: eventualmente pode ocorrer a necessidade da realização de manutenções, corretivas ou preventivas, em servidores ou demais equipamentos. Nesses casos as VMs dos servidores afetados deverão ser realocados para outro local.

- Consolidação de servidores: a medida em que VMs diminuem sua carga de trabalho ou vão sendo desligadas, os servidores tendem a ficar subutilizados. A realocação tem o objetivo de liberar os recursos do servidor, de modo que o mesmo possa ser hibernado ou desligado, retornando apenas quando a demanda por recursos exigir.

- Balanceamento de carga: conforme as VMs são alocadas, executadas ou finalizadas, pode ocorrer um desequilíbrio de carga entre os servidores físicos, alguns podem ficar sobrecarregados enquanto outros estejam subutilizados. Nesse contexto, o objetivo das realocações é levar o centro de dados a um estado de equilíbrio, diminuindo a carga de trabalho dos servidores sobrecarregados. Tal ação produz uma série de benefícios, tanto em relação a qualidade de serviço como nos custos operacionais do centro de dados. Um servidor sobrecarregado irá comprometer o desempenho de suas VMs hóspedes ao mesmo tempo que pode gerar pontos de calor, exigindo maior consumo de energia e necessidades de refrigeração.

3.3 Problemas envolvendo alocação de VM

Como mencionado, devido a sua característica combinatória, a alocação de VMs representa um desafio algorítmico interessante do ponto de vista computacional, dando origem a alguns problemas que exploram essa característica. Essa seção apresenta alguns desse problemas, bem como um comparativo entre o problema de alocação de VMs e alguns algoritmos clássicos de computação.

3.3.1 Similaridade com problemas clássicos

De uma forma geral, o problema de alocação de VMs pode ser visto como um clássico problema da mochila (*Knapsack Problem – KP*) (Martello and Toth, 1990). Nesse problema, existe uma determinada quantidade de itens, cada um com seu peso e valor, onde se deseja colocá-los em uma mochila com uma capacidade predefinida (Figura 3.1). O objetivo é colocar os itens na mochila de modo a se obter o maior valor (composto pela soma dos valores dos itens inseridos na mochila) não ultrapassando o peso total suportado pela mochila.

Definindo o problema de um modo formal, tem-se um conjunto de n itens, onde cada item $j \in \{1,...,n\}$ possui um peso w_j e um valor v_j associados. O valor é uma medida que

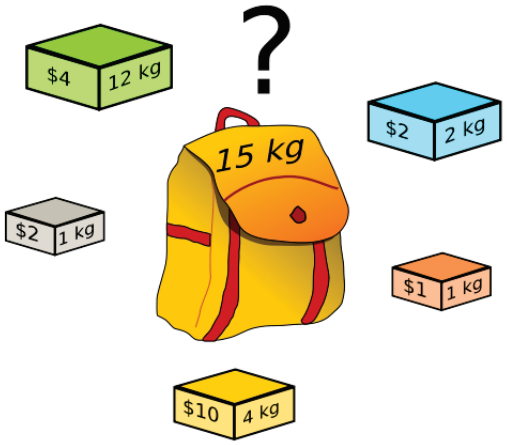


Figura 3.1: Ilustração do problema da mochila

indica a importância daquele objeto. A mochila possui uma capacidade C, determinando o peso máximo suportado por ela. Uma variável x_j é usada para especificar se um item j foi colocado na mochila. Se o item j foi escolhido para ser colocado na mochila, a variável x_j será preenchida com o valor 1; caso contrário, será preenchida com o valor 0. Como objetivo para solução do problema, espera-se maximizar o valor obtido pelos itens inseridos. Matematicamente, o problema da mochila pode ser definido como:

$$\text{Maximizar} \qquad z = \sum_{j=1}^{n} v_j x_j \tag{3.1}$$

$$\text{sujeito a} \qquad \sum_{j=1}^{n} w_j x_j \leq C, \tag{3.2}$$

$$x_j = 0 \, ou \, 1, j \in N = \{1, ..., n\} \tag{3.3}$$

Na equação 3.1 é definido o objetivo de maximizar o valor obtido pelos itens inseridos na mochila. Tal valor é obtido pelo somatório dos valores v dos itens inseridos na mochila, e está sujeito a duas restrições: a restrição 3.2 especifica que a soma dos pesos w dos itens inseridos não pode ser maior que a capacidade C da mochila e a restrição 3.3 especifica os valores que a variável x_j pode receber (0 ou 1), indicando respectivamente se o item não foi inserido ou foi.

No caso da alocação de VMs, a mochila representa um servidor físico, e os itens as VMs que devem ser alocadas, o peso representa a quantidade de recursos que a VM consumirá do servidor e o valor pode representar o custo (ou lucro) que essa alocação trará ao ambiente.

No caso em que o problema de alocação deve levar em consideração múltiplos servidores, tem-se caracterizado o problema de múltiplas mochilas (*Multiple Knapsack Problem – MKP*). É um problema similar ao problema da mochila simples, com a diferença de se possuir várias mochilas ($m > 1$).

Quando o problema de alocação precisa levar em consideração diferentes tipos de recursos (CPU, memória, etc), o problema da mochila se torna multidimensional (*Multidimensional Knapsack Problema - MDKP*) (Grit et al., 2006).

Os algoritmos de alocação de VMs também podem se apresentar como variações do problema *Bin Packing* (Johnson, 1974). O *Bin Packing* pode ser descrito da seguinte maneira: dado um conjunto de n itens, onde cada item possui um peso w associado, e um conjunto de m recipientes (*bins*), cada um com uma capacidade c, encontrar o número mínimo de recipientes

necessários para armazenar os itens, tal que a soma dos itens em cada recipiente não ultrapasse sua capacidade.

Comparando-se com o problema de alocação de VMs, tem-se os recipientes como sendo os servidores físicos, suas capacidades como os recursos disponíveis no servidor e os itens seriam as VMs a serem alocadas.

Várias heurísticas buscam encontrar a melhor solução para o problema, a seguir são apresentadas algumas das mais comuns (Seiden, 2002):

First-Fit : O item será alocado no primeiro recipiente com capacidade disponível. Nesse algoritmo é realizada uma busca em todos os recipientes já utilizados, um novo recipiente só é aberto quando não houver nenhum com capacidade disponível.

Next-Fit : O item será alocado no próximo recipiente com capacidade disponível. Assim, caso o recipiente atual não disponha de capacidade, então um novo recipiente é aberto para receber o item e o recipiente atual não é mais visitado.

Best-Fit : O item será alocado no recipiente que resultar na menor capacidade residual, ou seja, aquele que resultar em um melhor preenchimento do recipiente. Assim como no *First-Fit*, um novo recipiente só é aberto caso não houver nenhum com capacidade disponível.

Worst-Fit : O item será alocado no recipiente que resultar na maior capacidade residual, ou seja, naquele que resultar na maior capacidade disponível após a alocação. O critério para abertura de um novo recipiente seguem as estratégias anteriores.

Variações dessas estratégias se apresentam caso o conjunto de itens tenha sido previamente ordenado. Caso os itens estejam ordenados em forma crescente, tem-se *Next-Fit Increasing*, *First-Fit Increasing*, *Best-Fit Increasing* e *Worst-Fit Increasing*. O mesmo ocorre caso os itens estejam ordenados em forma decrescente, nesse caso as estratégias *Next-Fit Decreasing*, *First-Fit Decreasing*, *Best-Fit Decreasing* e *Worst-Fit Decreasing*.

Assim como o problema da mochila, o *bin packing* tem sua versão multidimensional, que é semelhante ao problema de alocação de VMs considerando múltiplos recursos. Apesar da similaridade, o problema não é exatamente o mesmo. Por exemplo, considerando-se que uma VM é um objeto tridimensional (CPU, memória e disco com sendo as três dimensões) ela assemelha-se ao problema *3D bin packing* (*bin packing* com três dimensões), onde um conjunto de objetos tridimensionais devem ser alocados em um recipiente também tridimensional. O objetivo é o de empacotar o maior quantidade possível de objetos, de modo que o número de recipientes exigidos seja minimizado. Enquanto é realizado o empacotamento, dois objetos podem ser colocados lado a lado ou um acima do outro. Porém, considerando-se as VMs como objetos, colocar as VMs lado a lado ou uma acima da outra não é uma operação válida. Isto porque uma vez que um recurso é utilizado ou ocupado por uma VM, ele não pode ser reutilizado por qualquer outra VM (Mishra and Sahoo, 2011).

A Figura 3.2 ilustra essa situação. Para simplificar foram utilizadas apenas duas dimensões de recursos para o servidor físico (CPU e memória). No servidor, uma certa quantidade de recursos está sendo utilizada por outra(s) VM(s) (representado pelo retângulo claro). Quando uma nova VM (retângulo escuro) chega para ser alocada nesse servidor, a única posição disponível está marcada com $\surd$. No caso do problema do *bin packing* as áreas marcadas com $\times$ também estariam disponíveis.

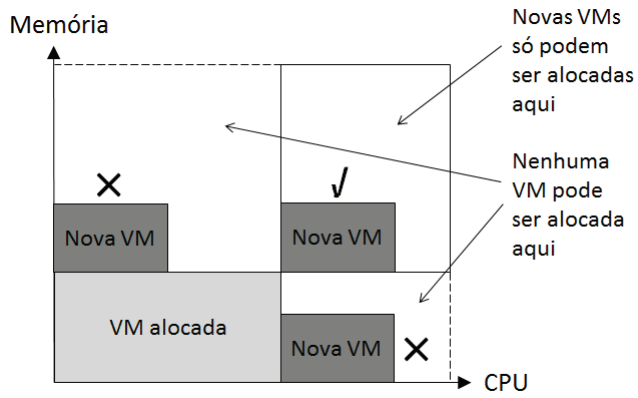


Figura 3.2: Alocação de VM em um espaço bidimensional
, adaptado de (Mishra and Sahoo, 2011)

Em relação a presente pesquisa, tal situação não é totalmente verdadeira, como se busca aproveitar a possibilidade de compartilhamento de memória, a dimensão referente a memória pode sofrer intercalação entre VMs. Esse cenário está representado na Figura 3.3, onde a área hachurada apresenta a memória compartilhada entre as VMs.

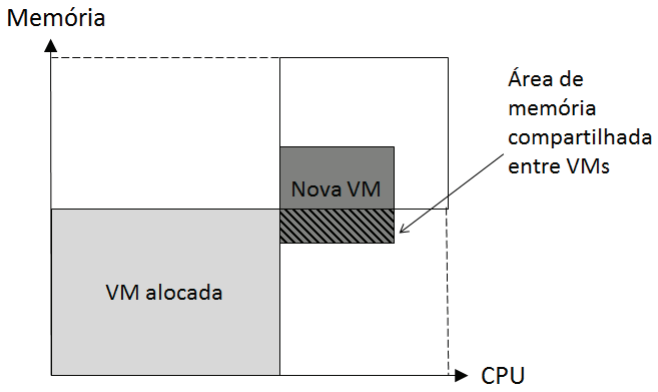


Figura 3.3: Alocação de VM considerando compartilhamento de memória

3.3.2 Problemas de otimização

Em muitos casos, os algoritmos de alocação de VMs buscam otimizar o uso dos recursos da infraestrutura, buscando principalmente a redução de custos. A seguir são apresentados dois exemplos de problemas de otimizações envolvendo um conjunto conhecido de VMs (Sindelar et al., 2011):

VM Maximization (maximização de VMs): nesse tipo de problema, é dado um conjunto de n VMs, onde cada VM está associada a um valor de lucro, que é obtido ao alocar a VM. Também são dados m servidores, cada um com uma capacidade de recursos. O objetivo é determinar o subconjunto das VMs que podem ser hospedados nos m servidores de modo que o lucro total obtido seja maximizado. Este problema caracteriza o desejo de um

prestador de serviços em maximizar o lucro obtido por um conjunto fixo de recursos de servidor.

VM Packing (empacotamento de VMs): para esse problema é dado um conjunto de VMs que devem ser hospedadas em servidores físicos. O objetivo é alocar todas as VMs no menor número possível de servidores físicos, considerando que cada servidor tem uma capacidade de recursos limitada. Este problema caracteriza fundamentalmente os objetivos de um provedor de serviços em hospedar um conjunto de VMs usando a menor quantidade de recursos de servidor.

3.3.3 Problema do Rearranjamento Iterativo

Conforme as políticas e condições do centro de dados mudam, torna-se necessário realizar a realocação das VMs através de sequências de migrações. É importante que após as migrações, as VMs se apresentem com a mesma estabilidade e comportamento que estavam antes da migração.

O problema do planejamento das migrações é tratado por (Hyser et al., 2007) como o "Problema do Rearranjamento Iterativo" e pode ser formulado da seguinte maneira: dado um conjunto de servidores físicos, um conjunto de VMs distribuídas entre os servidores e uma nova reatribuição das VMs, encontrar uma sequência de migrações que convergem à nova atribuição com um custo mínimo. Os custos, no caso, incluem transferência e armazenamento de dados, interrupção de serviços e o número de migrações.

O problema começa com uma configuração inicial onde as VMs já estão alocadas nos servidores físicos, e qualquer algoritmo que solucione esse problema deve gerar uma solução a partir desse *layout* inicial. Tais soluções são listas de migrações seriais e paralelas que deverão ser executadas para levar o centro de dados do estado atual para o estado desejado. É importante que após as migrações, as VMs se apresentem com a mesma estabilidade e comportamento que estavam antes da migração.

Como parte da avaliação do mapeamento, deverá ser previsto como as atuais cargas de recursos ficarão no novo mapeamento. Considerando várias dimensões de consumo de recursos é possível predizer o efeito que as migrações terão nos servidores físicos de origem e de destino. (Hyser et al., 2007) em seu trabalho consideraram as dimensões de memória, CPU, banda de rede e banda de disco.

É possível perceber que trata-se de um problema combinatório em que as possibilidades de mapeamento tendem a crescer de forma exponencial, conforme aumentam o número de servidores físicos e máquinas virtuais. O número de possíveis soluções seria da ordem de S^V, onde S é o número de servidores físicos e V o número de máquinas virtuais. Isso torna impraticável, mesmo para sistemas automatizados, analisar todas as soluções. Tal situação caracteriza o problema como sendo do tipo *NP-hard* (*Non-deterministic Polynomial-time hard*) (Grit et al., 2006), (Singh et al., 2008).

Algoritmos como *Simulated Annealing*, algoritmos genéticos ou otimizações por colônia de formigas podem solucionar o problema (Hyser et al., 2007).

3.4 Estratégias e algoritmos propostos na literatura

A literatura apresenta uma série de estratégias e algoritmos propostos para otimizar a carga de trabalho nos centros de dados através da alocação e realocação de VMs. Tais algoritmos variam conforme os recursos analisados para definir a alocação (CPU, memória, disco, rede, consumo de energia, número de migrações, etc), se levam ou não em consideração restrições quanto ao acordo de garantia do nível de serviço e quanto ao momento no qual o algoritmo é utilizado (alocação ou realocação).

3.4.1 VectorDot

O VectorDot é um algoritmo para balanceamento de carga que leva em consideração as limitações hierárquicas e multi-dimensionais de um ambiente integrado virtualizado de servidores e armazenamento. O algoritmo é inspirado no método Toyoda para o problema da mochila multidimensional (Singh et al., 2008).

O método busca manter sob controle a carga de trabalho incidente sobre os nós do sistema. Esses nós podem se referir a servidores, armazenamento ou *switches* de rede. A utilização dos recursos dos nós e os requisitos por recursos das VMs são expressos como vetores.

Inicialmente busca-se identificar a existência de algum nó sobrecarregado, para isso o algoritmo recebe o estado atual do centro de dados, incluindo o estado dos nós servidores, dos nós de armazenamento e demais nós de rede. Também é recebido o conjunto de VMs e discos virtuais do sistema.

Cada tipo de nó possui um conjunto de recursos associados; para cada recurso são considerados três parâmetros relevantes: valor em uso atual, valor da capacidade e um valor limitador representado por uma fração entre 0 e 1. Esse limitador é utilizado pelo balanceador de carga para manter o uso dos recursos abaixo desse limite. A tabela 3.1 apresenta o conjunto de parâmetros para cada nó que é utilizado pelo algoritmo balanceador.

Tabela 3.1: Conjunto de parâmetros de cada nó

Item	**Tipo de recurso**	**Parâmetros**
Servidor	CPU	cpuU, cpuCap, cpuT (0..1)
	Memória	memU, memCap, memT (0..1)
	Banda de rede	netU, netCap, netT (0..1)
	Banda de disco	ioU, ioCap, ioT (0..1)
Armazenamento	Espaço	spaceU, spaceCap, spaceT (0..1)
	Taxa de I/O	ioU, ioCap, ioT (0..1)
Rede	Taxa de I/O	ioU, ioCap, ioT (0..1)

Se os nós do sistema estiverem abaixo de seus limitadores, então considera-se que o sistema está em um bom estado e nenhum balanceamento é necessário. No entanto, caso algum nó exceder seu limitador em qualquer uma das dimensões, o nó é considerado sobrecarregado e o objetivo do algoritmo balanceador é realizar realocações, movendo um ou mais itens (VMs ou discos virtuais) para trazer o nó sobrecarregado abaixo de seus limitadores.

Para medir o grau de sobrecarga dos nós e do sistema como um todo é calculado um índice de desequilíbrio: o *IBScore* (*Imbalance Score*). Este índice permite penalizar os servidores indicando em quanto seus recursos estão além dos limitadores estabelecidos. O *IBScore* de um

recurso é calculado pela seguinte função exponencial, onde f representa a carga em uso e T o limitador estabelecido para o recurso:

$$IBScore(f,T) = \begin{cases} 0 & sef < T \\ e^{(f-T)/T)} & sef >= T \end{cases}$$

O *IBscore* de um nó é obtido então pelo somatório dos *IBScore* de cada um de seus recursos e, por consequência, o desequilíbrio total do sistema (*TotalIbScore*) é dado pelo somatório dos *IBScore* de todos os servidores do sistema.

Uma vez calculado o índice de desequilíbrio total do sistema, o objetivo do algoritmo de balanceamento é reduzi-lo o máximo possível através da migração de uma ou mais VMs.

É realizado então uma busca por servidores para realocar as VMs dos nós sobrecarregados. Para isso são utilizados dois vetores multidimensionais. O primeiro, denominado de *NodeLoadFracVec(u)*, representa a fração de uso de cada recurso para o servidor u e o vetor é dado por $\left\langle \frac{cpuU}{cpuCap}, \frac{memU}{memCap}, \frac{netU}{netCap}, \frac{ioU}{ioCap} \right\rangle$. O segundo vetor representa a carga da VM *vi* em relação ao servidor u, é denominado *ItemNodeLoadFracVec(vi, u)* e é construído como: $\left\langle \frac{cpuU(vi)}{cpuCap(u)}, \frac{memU(vi)}{memCap(u)}, \frac{netU(vi)}{netCap(u)}, \frac{ioU(vi)}{ioCap(u)} \right\rangle$.

Para calcular a atratividade de um servidor em receber uma VM, os vetores *NodeLoadFracVec(u)* e *ItemNodeLoadFracVec(vi, u)* são ajustados para considerar os custos da migração e então é realizado o cálculo do produto escalar entre esses vetores. O servidor físico que gerar o menor valor será o escolhido para receber a VM.

3.4.2 Autonomic Resource Management

O trabalho de Nguyen Van et al. (2009) apresenta um sistema autônomo para gestão de recursos que traz a capacidade de automatizar o provisionamento dinâmico e a alocação de VMs tendo em vista tanto o cumprimento dos acordos de nível serviço (SLA) para o nível das aplicações quanto os custos da utilização dos recursos.

Os estágios de provisionamento e colocação de VMs foram separados, estabelecendo objetivos distintos para cada fase:

- Provisionamento: dirigido aos objetivos de performance, associado ao nível de negócio e SLA.
- Alocação: dirigido as políticas do centro de dados, relacionado aos custos de gerenciamento dos recursos.

A arquitetura da solução é apresentada na Figura 3.4. Nessa arquitetura, o centro de dados consiste de um conjunto de máquinas físicas, cada uma hospedando várias máquinas virtuais (VM) sobre um hipervisor. É assumido que o número de máquinas físicas é fixo e é possível migrar uma VM entre duas máquinas quaisquer. Cada VM é associada a um ambiente de aplicação (AE, do inglês *Application Environment*), que encapsula uma aplicação hospedada pelo sistema de nuvem. As VMs requisitadas pelas aplicações devem estar pré-definidas em classes específicas onde são determinadas a capacidade de CPU e de memória.

Cada AE possui associado um módulo de decisão local – LDM (do inglês, *Local Decision Module*) que, de acordo com a carga de trabalho, avalia a necessidade de se alocar mais

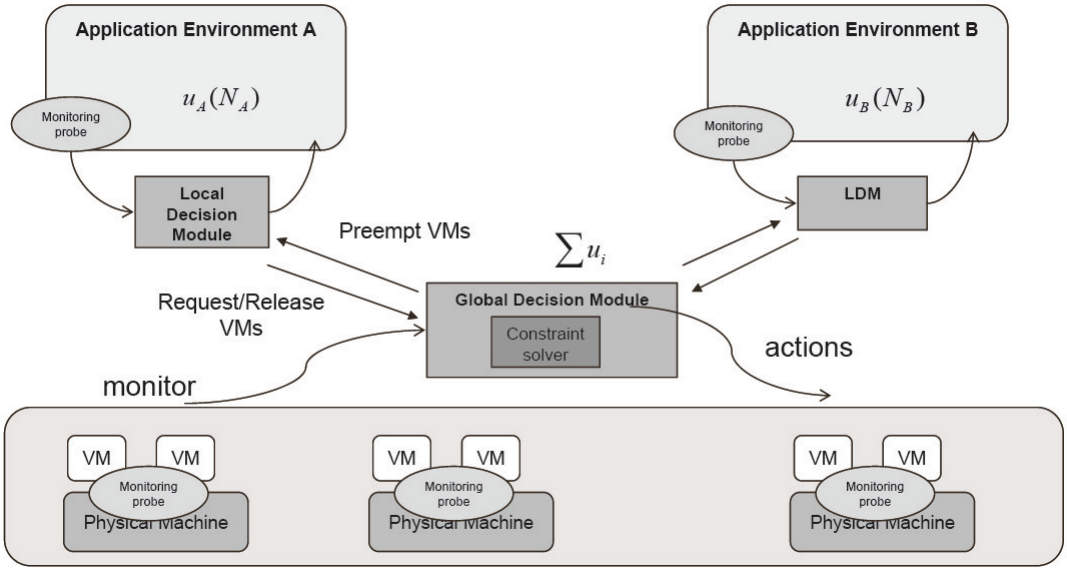


Figura 3.4: *Autonomic Resource Management* – Arquitetura
(Nguyen Van et al., 2009)

VMs ou liberar VMs não mais utilizadas. Sua tarefa principal é calcular uma função de utilidade, que representa uma medida da satisfação em relação aos recursos alocados de CPU e memória.

Cada LDM interage com um módulo de decisão global – GDM (do inglês, *Global Decision Module*). O GDM é responsável por duas tarefas principais:

- Provisionar as VMs, determinando o conjunto de VMs necessárias à aplicações;
- Alocar as VMs nos servidores físicos de modo a utilizar a menor quantidade possível de servidores.

A execução periódica do algoritmo pode gerar resultados diferenciados, de acordo com o estado do centro de dados, esses resultados podem ser usados para a realização de migrações buscando a consolidação de servidores.

3.4.3 Energy-Efficient Manager

Dentro do contexto da computação em nuvem verde, o trabalho de Buyya et al. (2010) busca melhorar a eficiência energética através de uma alocação de VMs eficiente, que minimize sobrecargas de processamento e de comunicação e que use o menor número possível de recursos.

O problema de alocação de máquinas virtuais foi dividido em dois: o primeiro é relacionado à admissão de uma nova VM para colocação nos servidores físicos e a segunda é em relação a otimização das alocações correntes de VMs.

A primeira parte é abordada como um clássico problema de empacotamento *Bin packing*, com recipientes de tamanhos variáveis (servidores físicos) onde devem ser alocados itens (VMs) com custos diferenciados (custo energético da alocação). A solução busca a utilização da menor quantidade de servidores.

A economia de energia obtida ao utilizar essa abordagem é devido ao fato de uma quantidade menor de servidores serem utilizados para atender as requisições por máquinas virtuais. Desse modo, as máquinas ociosas podem ser hibernadas, de modo a consumir menos energia.

Para solucionar o problema foi utilizado o algoritmo *Modified Best Fit Decreasing* (MBFD), baseado na heurística BFD (*Best Fit Decreasing*). Nesse algoritmo, as máquinas virtuais a serem alocadas são ordenadas de maneira decrescente de acordo com a taxa de utilização da CPU. Em seguida, cada VM é alocada no servidor que trará o menor consumo de energia após a alocação.

Além do algoritmo de alocação, o trabalho também traz a proposta de uma política de migração de máquinas virtuais para otimização da alocação corrente de VMs. Nessa política são estabelecidos os limites inferior e superior de utilização dos recursos. A taxa de utilização dos servidores deve estar entre esses limites, portanto podem ocorrer duas situações:

- A taxa de utilização da CPU está abaixo do limite mínimo: as máquinas virtuais presentes no servidor são migradas e o servidor é hibernado, reduzindo o consumo de energia.
- A taxa de utilização da CPU está acima do limite máximo: é selecionado um conjunto de máquinas virtuais para serem migradas, de modo a diminuir a carga de trabalho e manter a taxa de utilização da CPU dentro dos limites. Esse conjunto deve conter o menor número possível de máquinas virtuais, de modo a causar o menor impacto possível e evitar problemas de violação do SLA.

Uma vez selecionadas as VMs que deverão ser migradas, elas são realocadas nos servidores utilizando o algoritmo MBFD, com a condição de manter inviolado o limite máximo de utilização nos novos servidores.

3.4.4 Memory Buddies

O estudo de Wood et al. (2009) traz uma estratégia que usa a possibilidade de compartilhamento de páginas de memória para melhorar a colocação de VMs nos servidores físicos. A colocação é determinada através de um cálculo de probabilidade de existência de páginas compartilháveis.

Como o compartilhamento de páginas é um processo dinâmico, só é possível conhecer o número de páginas compartilháveis após a efetiva colocação da VM em um servidor físico. Antes disso não há como saber de antemão se uma VM terá ou não páginas compartilhadas em qualquer servidor. Diante disso a estratégia utilizada é de se calcular uma “impressão digital” para cada página de memória, que é gerada através de uma função hash, e comparar essa impressão digital com todas as impressões digitais de todas as máquinas físicas de modo a se calcular quais teriam as maiores probabilidades de compartilhamento.

O sistema consiste então de um núcleo, que executa em cada servidor, e um plano de controle que executa em um servidor distinto, usado para controle, conforme ilustrado na Figura 3.5. Cada núcleo gera as impressões digitais de todas as páginas de memórias existentes nas máquinas virtuais daquele servidor e agrega esses resultados por servidor. Com isso é possível calcular o potencial de compartilhamento das potenciais VM candidatas a migração.

O plano de controle é o responsável pela colocação das VM nos servidores e pela mitigação de sobrecargas. Quanto a colocação de VMs, é provido suporte para três tipos de alocação:

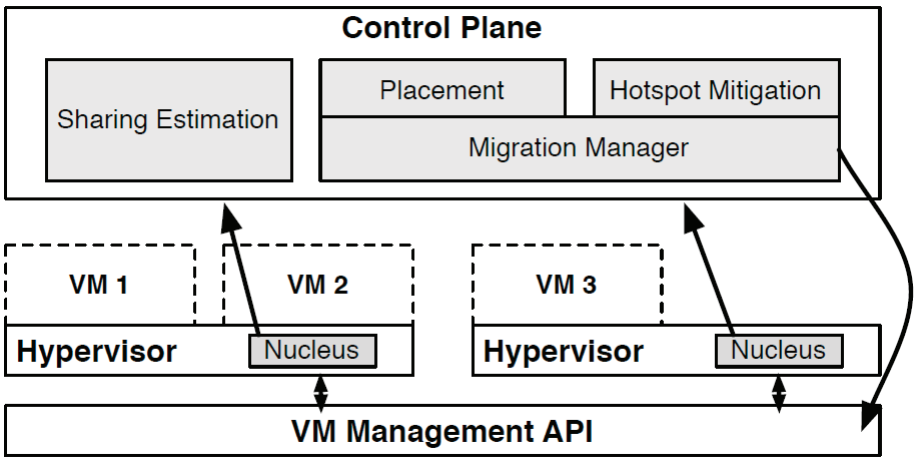


Figura 3.5: *Memory Buddies* – Arquitetura
Wood et al. (2009)

- Alocação inicial: quando uma nova VM é adicionada ao centro de dados ela é inicialmente colocada em um servidor de "teste". Após um período de estabilização são observadas as impressões digitais geradas para a VM e então, dentre um conjunto de servidores aptos a recebê-la, é escolhido o servidor que trará o maior potencial de compartilhamento, calculado através da técnica de comparação de impressões digitais.

- Consolidação de servidores: busca selecionar os servidores candidatos a serem desligados e tenta migrar as máquinas virtuais para servidores que tragam grandes chances de compartilhamento de memória. O algoritmo de consolidação compreende a três fases:

  1. Identificar os servidores a serem consolidados;
  2. Determinar os servidores de destino;
  3. Migrar as VMs para os novos servidores.

- Planejamento de colocação *off-line*: usado para estimar a capacidade necessária do centro de dados para hospedar um conjunto de VMs.

Em relação a mitigação de sobrecargas, a técnica trabalha em conjunto com o mecanismo de consolidação de servidores, para tratar de problemas de pressão sobre a memória decorrente de mudanças no comportamento das máquinas virtuais. O sistema detecta tais pontos de sobrecarga e mitiga os efeitos distribuindo a carga entre os demais servidores físicos.

3.4.5 Sharing Aware

A estratégia de Sindelar et al. (2011) trabalha com a possibilidade de compartilhamento de páginas de memórias e faz uso de um modelo hierárquico para definir o mapeamento das VMs para os servidores físicos.

Esse modelo leva em consideração as características das máquinas virtuais como sistema operacional, CPU, RAM, versão do sistema operacional, aplicações instaladas. Segundo os autores, VMs executando uma mesma plataforma de sistema operacional têm maiores chances

de compartilhamento de memória, o mesmo ocorre para sistemas operacionais com a mesma versão. Por exemplo, a probabilidade de compartilhamento de memória será muito maior para uma máquina executando duas VMs com o Windows XP do que a mesma máquina executando uma versão do Windows XP e a outra o Windows 7.

De posse da classificação hierárquica busca-se determinar as melhores localizações para as VMs. O processo é de rearranjamento, pois ele não trata da chegada de uma nova VM, mas apenas da reorganização das VMs existentes. Para isso o algoritmo usa a heurística *First Fit Decreasing* para solução do problema como *bin packing*.

3.4.6 Lago Allocator

No trabalho de Lago et al. (2011) é proposto um mecanismo em que a alocação é realizada no servidor com a maior eficiência energética. Para verificação do consumo de energia são utilizados dois critérios: a utilização da CPU e a temperatura do computador.

Para isso, o algoritmo faz uso de duas técnicas relacionadas à arquitetura de computadores:

- *Dynamic Voltage and Frequency Scaling* (DVFS): técnica na qual é possível reduzir a voltagem e a frequência do *clock* do processador, gastando assim uma quantidade menor de energia
- *Fan Control*: um dos nomes usados para representar a tecnologia que monitora a temperatura do computador e, de acordo com as condições térmicas, regula a intensidade dos *coolers* ativos.

A alocação se baseia no consumo de energia consumido após a alocação dos recursos feitas através de previsão. Em linhas gerais o algoritmo tem o seguinte funcionamento:

1. É realizada uma busca por servidores que possuam recursos suficientes para atender a requisição. A condição analisada é a quantidade de CPU.
2. Para os servidores selecionados é calculado a eficiência energética, identificando o servidor com melhor desempenho energético. A eficiência é calculada relacionando-se a quantidade máxima de MIPS do servidor pela quantidade máxima de energia consumida.
3. Caso mais de um servidor possua o mesmo desempenho, o servidor com menor consumo de energia é o selecionado.
4. Caso ainda ocorra empate, é utilizado o servidor com maior utilização de CPU, com o objetivo de concentrar as VMs em um número menor de servidores.
5. Se o empate ainda persistir, o servidor com maior capacidade de MIPS é escolhido.

O algoritmo foi validado utilizando o simulador de infraestrutura de nuvem *CloudSim* (Calheiros et al., 2011).

3.5 Estudo comparativo

A tabela 3.2 apresenta os resultados do estudo das estratégias de alocação de máquinas virtuais apresentadas na seção anterior. Pretende-se através dessa tabela identificar as principais características de cada método, principalmente em relação aos tipos de critérios analisados por cada algoritmo para estabelecer a estratégia de alocação de máquinas virtuais: CPU, memória, memória compartilhável, rede, disco, consumo de energia e custo de migrações. Também estão identificados a forma de utilização do algoritmo, se é voltado para alocação inicial de uma VM ou para realocações de VMs existentes (migração de VMs entre servidores).

Na tabela os algoritmos estão identificados por acrônimos: VD = *VectorDot* (Seção 3.4.1), ARM = *Autonomic Resource Management* (Seção 3.4.2), EEM = *Energy-Efficiente Management* (Seção 3.4.3), MB = *Memory Buddies* (Seção 3.4.4), SA = *Sharing Aware* (Seção 3.4.5), LA = *Lago Allocator* (Seção 3.4.6).

Tabela 3.2: Estratégias de alocação de VMs

Estratégia	VD	ARM	EEM	MB	SA	LA
CPU	√	√	√			√
Memória	√	√				
Memória compartilhável				√	√	
Rede	√					
Disco	√					
Consumo de energia			√			√
Custo de migrações	√					
Alocação Inicial		√	√	√	√	√
Realocação	√		√	√	√	√

Percebe-se que o uso de CPU é o critério mais comum para seleção dos servidores para alocação. O VD é o algoritmo que analisa a maior quantidade de recursos (CPU, memória, disco e rede) além de considerar o custo das migrações de VMs. O VD também é o único que não contempla a alocação inicial de VMs, concentrando-se apenas nas migrações de VMs para o balanceamento de carga.

Em geral, os algoritmos que contemplam alocação inicial também podem ser usados para realocação, a única exceção é o algoritmo ARM. Contudo, seus autores abrem essa possibilidade, afirmando que a execução periódica do mesmo geraria novos resultados que poderiam ser usados em alguma estratégia para a consolidação de servidores.

A preocupação com a eficiência energética está presente em dois algoritmos: EEM e LA, que além de analisar o uso de CPU também verificam o consumo de energia gerado pelas alocações.

Em relação ao uso da memória compartilhável, os algoritmos MB e SA são os que levam em consideração essa possibilidade, no entanto os mesmos não tem preocupação com as demais dimensões de recursos.

Nesse cenário, a presente pesquisa encaixa-se como um algoritmo para alocação inicial de VM, que analisa múltiplas dimensões de recursos (CPU, memória, disco e rede) e considera a possibilidade de compartilhamento de memória.

3.6 Conclusão

Este capítulo apresentou algumas considerações sobre a alocação de máquinas virtuais nos ambientes de computação em nuvem bem como algumas estratégias e algoritmos usados para tratá-la. O estudo comparativo buscou, em linhas gerais, determinar as principais diferenças existentes entre os algoritmos, principalmente em relação aos recursos utilizados para estabelecer os critérios de avaliação, e nesse cenário posicionou o algoritmo da presente pesquisa.

O próximo capítulo apresenta o algoritmo proposto nessa pesquisa, faz sua modelagem e detalha o seu funcionamento, bem como os elementos necessários à sua execução.

Capítulo 4

Algoritmo Proposto

A proposta da presente pesquisa consiste na elaboração de um algoritmo para alocação de máquinas virtuais que garanta um uso equilibrado dos recursos físicos e, principalmente, faça proveito da capacidade de compartilhamento de memória entre máquinas virtuais. Neste capítulo é modelado o problema de alocação de VMs e apresentado o algoritmo utilizado na solução. Também é feito uma explanação sobre o fator de compartilhamento, índice utilizado no algoritmo para expressar a possibilidade de compartilhamento de memória entre VMs.

4.1 Introdução

Nos capítulos anteriores foram apresentadas a fundamentação teórica necessária à compreensão do ambiente na qual esse pesquisa está inserida e uma visão geral sobre o problema de alocação de máquinas virtuais e as diversas abordagens adotadas para solucioná-lo.

Este capítulo aborda a elaboração de um mecanismo de alocação *online* de máquinas virtuais nos servidores de um centro de dados de computação nas nuvens, considerando as demandas de recursos das máquinas virtuais, os recursos disponíveis nos servidores e a possibilidade de compartilhamento de memória entre as máquinas virtuais alocadas em um mesmo servidor físico. Entende-se por *alocação online*, conforme explicado na Seção 3.2.3, o fato das máquinas virtuais serem processadas a medida em que chegam ao sistema, em um fluxo sequencial.

O restante do capítulo está organizado da seguinte maneira: a seção 4.2 apresenta a modelagem do problema de alocação de máquinas virtuais; a seção 4.3 apresenta o funcionamento do algoritmo e por fim a seção 4.4 trata do fator de compartilhamento de memória utilizado no algoritmo.

4.2 Modelagem do problema de alocação de VMs

O problema de alocação de máquinas virtuais tratado nesta pesquisa, pode ser descrito da seguinte maneira: dado o conjunto de servidores U do centro de dados, e uma máquina virtual v a ser alocada, encontrar um servidor $u \in U$ adequado para hospedá-la.

A alocação deve levar em consideração as seguintes restrições:

- O servidor físico deve possuir recursos suficientes de CPU, memória, disco e rede para alocar a nova VM;

- A nova alocação não pode sobrecarregar nenhuma dimensão dos recursos do servidor físico;
- O uso dos recursos no servidor físico deve manter-se o mais equilibrado possível;
- Deve ser levado em consideração a possibilidade de compartilhamento de memória existente entre a nova VM e as VMs já alocadas nos servidores.

4.3 O algoritmo *VectorAlloc*

Para solução do problema de alocação de máquinas virtuais, conforme apresentado na seção anterior, foi desenvolvido um novo algoritmo denominado *VectorAlloc*. Este algoritmo tem o objetivo de realizar a alocação de máquinas virtuais nos servidores físicos do centro de dados. As VMs são processadas individualmente a medida que chegam no sistema, em forma de um fluxo sequencial. Os critérios de decisão para escolha do servidor levam em consideração os recursos de memória, CPU, disco e rede, além da possibilidade de compartilhamento de memória existente entre a VM e o servidor físico.

O algoritmo foi inspirado no *VectorDot* (Singh et al., 2008), um algoritmo de balanceamento de carga que utiliza múltiplas dimensões de recursos (CPU, memória, disco e rede) em uma representação vetorial, para análise de atratividade dos servidores físicos em receber uma VM a ser realocada.

As principais semelhanças entre o *VectorAlloc* e o *VectorDot* são:

- Representação vetorial dos recursos de cada servidor e das demandas das VMs;
- Cálculo da atratividade através do produto escalar entre os vetores;
- Preocupação com o uso equilibrado de recursos.

Apesar de inspirado no *VectorDot* o *VectorAlloc* apresenta diferenças significativas:

- O *VectorDot* é um algoritmo para balanceamento de carga que trata um conjunto de VMs que já se encontram alocadas nos servidores. O *VectorAlloc* trata de alocações *online* de VMs, ou seja, não há um prévio conhecimento do conjunto de máquinas virtuais; a alocação ocorre a medida em que estas chegam ao sistema.
- Os vetores do *VectorDot*, além dos valores dos recursos, expressam também os custos de migração envolvidos, considerando o caminho que uma VM realizará para migrar do seu atual servidor até o servidor escolhido. Como o *VectorAlloc* trata apenas de alocação inicial, tais ajustes não se fazem necessários.
- O *VectorAlloc* considera a possibilidade de compartilhamento de memória entre máquinas virtuais hospedadas em um mesmo servidor físico. No *VectorDot* não existe essa preocupação.

Para melhor compreensão do funcionamento do algoritmo, antes de descrevê-lo, será apresentada a noção de representação vetorial do uso dos recursos.

4.3.1 Representação Vetorial

O *VectorAlloc* utiliza quatro dimensões de recursos para avaliar a atratividade de um servidor em receber uma nova VM, são eles: memória, CPU, disco e rede. Esses recursos são agrupados e representados em forma de vetor, onde cada elemento do vetor representa um recurso.

O algoritmo *VectorAlloc* utiliza quatro diferentes vetores, a notação apresentada a seguir foi adaptada de (Mishra and Sahoo, 2011):

- $RUV(u)$ – *Resource Utilization Vector*: Vetor de utilização de recursos, que representa a carga atual dos recursos de um servidor físico u. Consiste na soma dos recursos utilizados por todas as VMs hospedadas no servidor em relação a capacidade total de recursos do servidor.

- $RRV(u,v)$ – *Resource Requirement Vector*: Vetor de requisitos de recursos, que representa os recursos requisitados por uma VM v em relação aos recursos de um servidor físico u. Indica a parcela de recursos do servidor que será consumida pela VM.

- $RTVmin$ – *Resource Threshold Vector minimum*: Vetor de limites mínimos de recursos, que contém os limitadores mínimos para cada tipo de recurso. Os recursos alocados em um servidor devem estar acima desses limites. O objetivo é garantir um certo nível de consolidação de servidores, não mantendo servidores sob baixa utilização.

- $RTVmax$ – *Resource Threshold Vector maximum*: Vetor de limites máximos de recursos, que contém os limitadores máximos para cada tipo de recurso. Os recursos alocados em um servidor devem estar abaixo desse limite. O objetivo aqui é proporcionar um melhor balanceamento de carga, evitando sobrecarregar os servidores em quaisquer das dimensões.

Apresentados os vetores utilizados, a seguir são estabelecidos a forma de cálculo de cada um deles.

4.3.2 Cálculo dos vetores

Considerando que cada servidor u tem uma conhecida capacidade total de recursos, expressos por $MemCap(u)$, $CPUCap(u)$, $DiskCap(u)$ e $NetCap(u)$; uma conhecida parcela dessa capacidade em uso, representado por $MemUse(u)$, $CPUUse(u)$, $DiskUse(u)$ e $NetUse(u)$; e uma quantidade de memória $MemShared(u)$ sendo compartilhada pelas VMs do servidor; o vetor $RUV(u)$ é obtido da seguinte maneira:

$$RUV(u) = \left[\frac{MemUse(u) - MemShared(u)}{MemCap(u)}, \frac{CPUUse(u)}{CPUCap(u)}, \frac{DiskUse(u)}{DiskCap(u)}, \frac{NetUse(u)}{NetCap(u)}\right]$$

Ou seja, $RUV(u)$ é calculado realizando-se o quociente entre os valores do recurso já alocado pela capacidade total do recurso. O elemento memória considera a quantidade de memória compartilhada entre as VMs. Por exemplo, um servidor físico que hospede duas máquinas virtuais vm_1 e vm_2, onde vm_1 demanda 100 MB de memória e vm_2 demanda 200 MB, utilizaria

300 MB de memória. Todavia, se 50 MB de memória estiverem compartilhados entre vm_1 e vm_2, o total de memória realmente utilizada será de 250 MB.

Considerando também uma máquina virtual v que necessite ser alocada e conhecendo-se os requisitos de recursos exigidos pela VM, expressos por $MemReq(v)$, $CPUReq(v)$, $DiskReq(v)$, $NetReq(v)$; e existindo um fator de compartilhamento α que indique o potencial de compartilhamento de memória para a VM; para qualquer servidor u pode ser calculado o vetor $RRV(u,v)$ da seguinte maneira:

$$RRV(u,v) = \left[(1-\alpha(v,u))\frac{MemReq(v)}{MemCap(u)}, \frac{CPUReq(v)}{CPUCap(u)}, \frac{DiskReq(v)}{DiskCap(u)}, \frac{NetReq(v)}{NetCap(u)}\right]$$

Assim, o vetor $RRV(u,v)$ representa a razão entre a demanda de recursos de uma VM v e os recursos disponíveis em um servidor físico u. O cálculo do recurso memória usa um *fator de compartilhamento* $\alpha(v,u)$. Este fator representa o potencial de compartilhamento de memória que uma VM v possui em relação ao servidor u. Por exemplo, uma VM que tenha um fator de compartilhamento de 20%, precisará alocar somente 80% de sua demanda de memória. O cálculo de α será discutido na próxima seção.

Finalmente, pode ser de interesse do centro de dados, manter o uso de recursos dentro de certos limites, de modo a garantir uma melhor distribuição da carga de trabalho, evitando assim a presença de servidores com recursos sobrecarregados ou subutilizados. Tais limitadores representam os percentuais mínimos ($MemTmin$, $CPUTmin$, $DiskTmin$, $NetTmin$) e máximos ($MemTmax$, $CPUTmax$, $DiskTmax$, $NetTmax$) de consumo de recursos; com isso é possível montar os vetores $RTVmin$ e $RTVmax$ que serão utilizados como parâmetros para as alocações no centro de dados:

$$\begin{aligned} RTVmin &= [MemTmin, CPUTmin, DiskTmin, NetTmin] \\ RTVmax &= [MemTmax, CPUTmax, DiskTmax, NetTmax] \end{aligned}$$

Os vetores $RTVmin$ e $RTVmax$ indicam os limites mínimos e máximos de uso dos recursos em cada servidor. Seu objetivo é garantir que a alocação de uma nova VM não desequilibre o uso de recursos, sobrecarregando ou subutilizando algum servidor. Assim, uma VM v só poderá ser alocada no servidor u se o uso de seus recursos se mantiver entre os limites inferior e superior, ou seja, se $RTVmin \leq RUV(u) + RRV(u,v) \leq RTVmax$. Apenas em casos extremos, em que não existe um servidor adequado dentro desses limites, o algoritmo irá desconsiderar esses limitadores.

4.3.3 Funcionamento do algoritmo

Uma vez definidos os vetores utilizados pelo algoritmo, é possível apresentar o seu funcionamento. O algoritmo consiste essencialmente em realizar uma busca por servidores que atendam as condições de recursos exigidos pela VM, dentro dos limites estabelecidos, sendo escolhido aquele que trouxer o uso mais equilibrado de recursos, já considerado o compartilhamento de memória.

Considerando o conjunto de servidores U do centro de dados e uma máquina virtual v a ser alocada em um servidor $u \in U$, determina-se o conjunto $D(v, \gamma, \delta)$ de servidores disponíveis para alocar v como:

$$D(v, \gamma, \delta) = \{u \in U \mid \gamma \leq RUV(u) + RRV(u, v) \leq \delta\}$$

Com $\gamma = RTVmin$ e $\delta = RTVmax$, $D(v, \gamma, \delta)$ contém os servidores de U onde a alocação de v respeita os limites $RTVmin$ e $RTVmax$. Caso $D(v, \gamma, \delta) = \emptyset$, calcula-se $D(v, 0, \delta)$; caso este seja vazio, calcula-se $D(v, 0, 1)$; caso este também seja vazio, a alocação não é possível. Do conjunto $D(v) \neq \emptyset$ escolhe-se um servidor u_a que satisfaça:

$$u_a = \arg\min_{u \in D(v)} (RUV(u) \cdot RRV(u, v))$$

Em outras palavras, a máquina virtual v será alocada no servidor $u_a \in D(v)$ com o menor produto escalar entre $RUV(u)$ e $RRV(u, v)$, buscando respeitar os limites de alocação. A ideia subjacente é alocar a VM em um servidor para o qual RRV seja complementar a RUV, resultando em um produto escalar mínimo, o que induz um uso balanceado dos recursos. Por exemplo, uma VM com baixa demanda de CPU e alta demanda de memória será alocada em um servidor com alto uso de CPU mas baixo uso de memória. Para o algoritmo, o fator de compartilhamento de memória entra como um redutor da capacidade de memória requerida, assim um servidor que possua uma alta probabilidade de compartilhamento terá redução nos requisitos de memória, fazendo com que o produto escalar venha a cair e a chance de alocação no servidor seja maior. A Figura 4.1 apresenta esse processo em forma de fluxograma.

4.4 Fator de compartilhamento

Técnicas como CBPS e KSM buscam otimizar o uso da memória através do compartilhamento de páginas de memória idênticas entre VMs. Estudos indicam que esse compartilhamento tem maiores chances de sucesso conforme aumenta a similaridade entre as VMs (Chang et al., 2011), (Barker et al., 2012). VMs que possuam o mesmo sistema operacional, com a mesma versão, e executando sobre uma mesma plataforma (32/64 bits) tem muito mais chances de compartilharem memória do que VMs com sistemas operacionais diferentes. Isso deve-se ao fato de uma grande quantidade de memória ser utilizada para armazenar o sistema operacional e bibliotecas na memória.

Diante disso, a presente pesquisa propõe um fator de compartilhamento, nesta pesquisa identificado por α, que estabelece o potencial de compartilhamento de memória que uma VM possui em relação a um servidor.

Esse fator de compartilhamento deve ser calculado levando-se em consideração as características da máquina virtual a ser alocada e das máquinas virtuais já hospedadas no servidor físico. Quanto maior o grau de similaridade entre as VMs, maior será o fator de compartilhamento.

O fator de compartilhamento $\alpha(v, u)$ é usado pelo algoritmo *VectorAlloc* para estabelecer o potencial de compartilhamento de memória que uma VM v possui em relação a um servidor u. É um valor que varia entre 0 e 1, sendo 0 quando não há nenhuma chance de v compartilhar memória com as demais VMs presentes em u, e 1 caso toda a memória de v possa ser compartilhada.

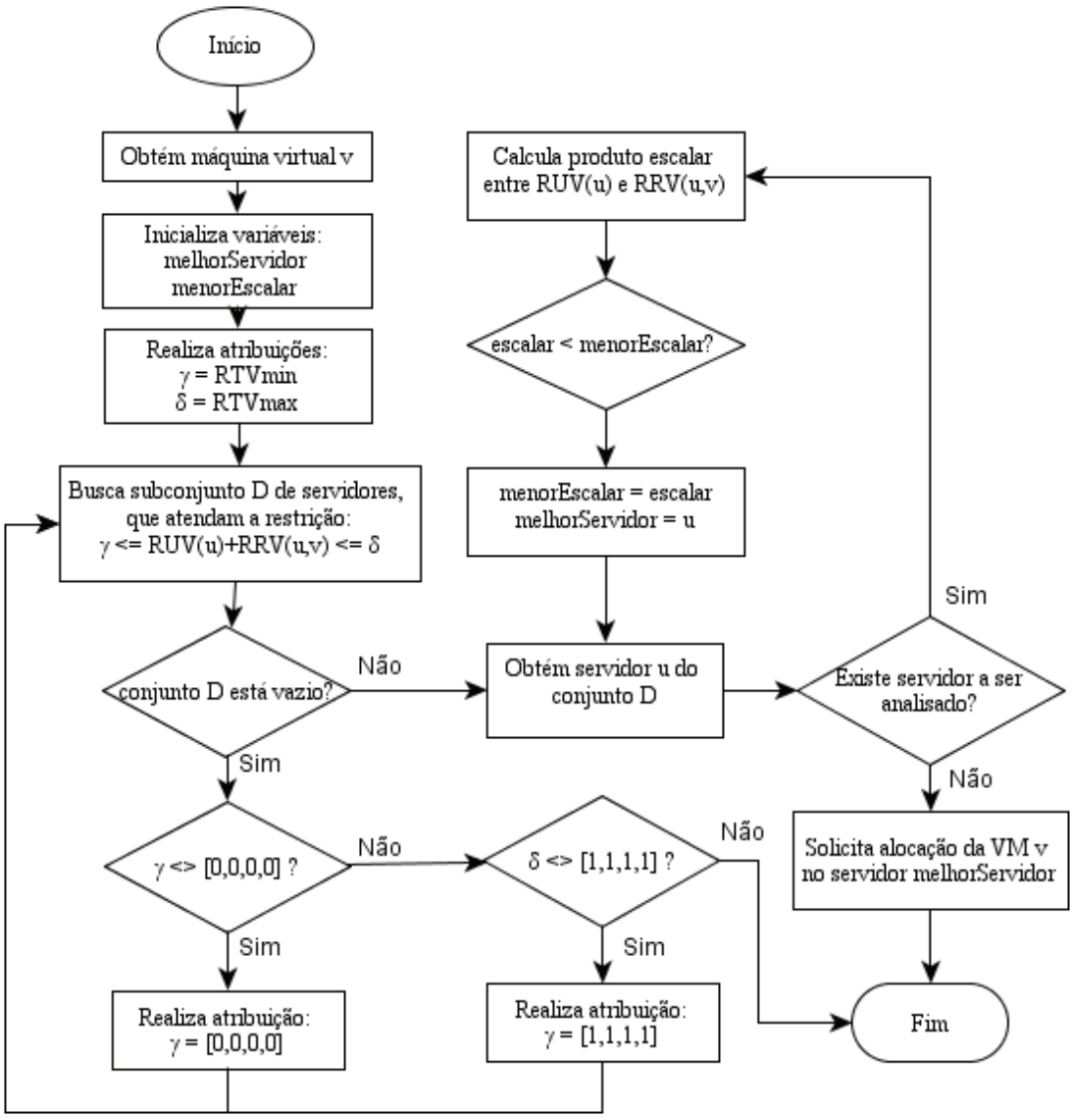


Figura 4.1: Fluxograma do algoritmo *VectorAlloc*

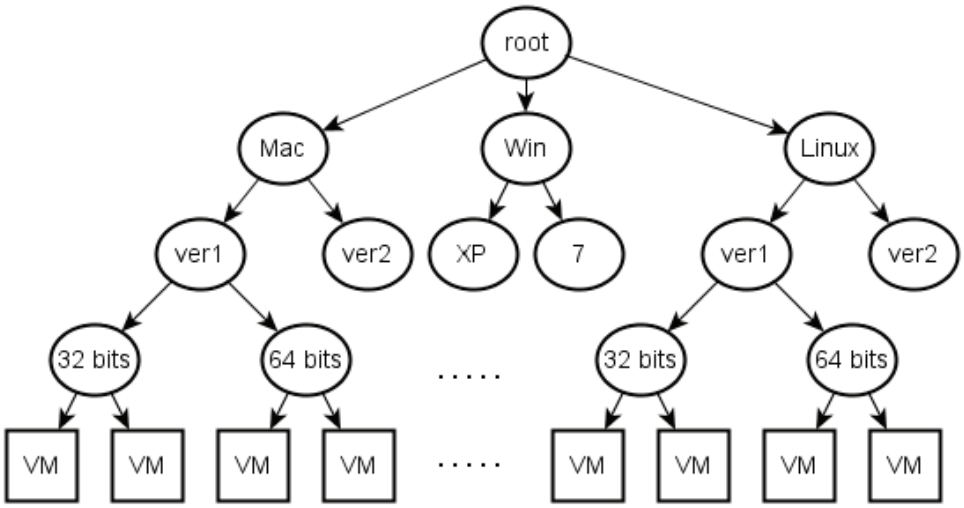


Figura 4.2: Modelo hierárquico em árvore
(Sindelar et al., 2011)

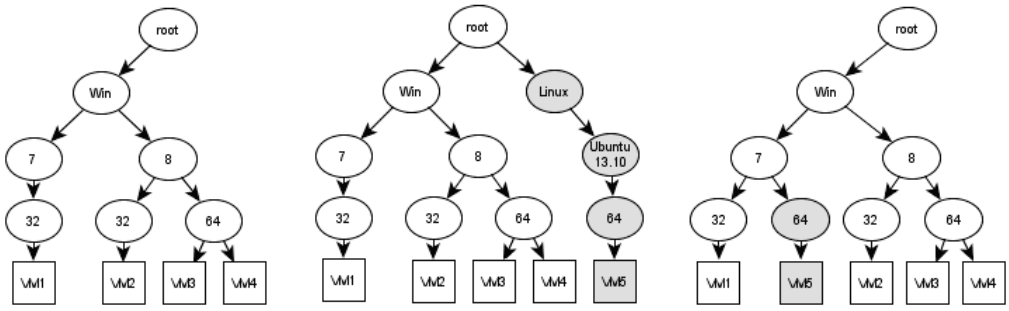
O valor de α corresponde a uma estimativa, já que a memória efetivamente compartilhada só poderá ser determinada após a alocação da VM, quando então os mecanismos do hipervisor buscam por páginas idênticas. Este trabalho não se aprofunda no cálculo preciso de α, pois foge do seu escopo. Aqui, inclusive, abre-se uma possibilidade de pesquisa para o estabelecimento de métodos para o cálculo de α, que poderiam fazer uso do histórico de alocações para determinar índices mais próximos da realidade. Alguns trabalhos como (Wood et al., 2009) e (Sindelar et al., 2011) apresentaram formas de capturar a memória compartilhada entre as máquinas virtuais e utilizaram tal informação para criação de algoritmos de alocação, otimização e balanceamento de carga.

Neste trabalho, o fator de compartilhamento é calculado usando uma representação hierárquica de máquinas virtuais, similar à apresentada em (Sindelar et al., 2011). Cada servidor referencia as máquinas virtuais nele alocadas usando uma árvore similar à apresentada na Figura 4.2. Nessa árvore, cada nível representa um novo grau de especialização das informações das VMs. Partindo da raiz, o próximo nível representa o sistema operacional (SO), em seguida vem a versão do SO e então a arquitetura do SO (32 ou 64 bits). As folhas da árvore representam as máquinas virtuais alocadas no servidor.

Sindelar et al. (2011) utiliza essa árvore para capturar as páginas de memória das VMs alocadas. Em seu modelo, a raiz da árvore contém todas as páginas de memória compartilhadas entre as VMs, os nós no nível do SO contêm as páginas compartilhadas pelo mesmo SO e o mesmo ocorre nos nós para a versão do SO e para a arquitetura do SO. As folhas contêm as páginas de memória que não são compartilhadas, ou seja, são exclusivas de cada VM. Nosso trabalho adotou uma interpretação diversa: cada nível indica o potencial de compartilhamento para uma dada VM. Conforme aumenta o grau de similaridade entre as VMs, também aumenta a probabilidade dessas VMs compartilharem páginas de memória comuns entre si. Dessa forma, ao realizar-se uma busca na árvore por VMs semelhantes a uma dada VM, quanto maior a profundidade alcançada, maior será a probabilidade de compartilhamento de memória.

A Figura 4.3(a) ilustra um exemplo da árvore de alocação em um servidor físico u. Nesse exemplo existem quatro VMs alocadas: vm_1 é uma máquina Windows 7 32 bits, vm_2 é Windows 8 32 bits e vm_3 e vm_4 são Windows 8 64 bits. Supondo que uma VM Linux Ubuntu

13.10, 64 bits (vm_5) deva ser alocada em u. Como vm_5 não é similar a nenhum nó da árvore, $\alpha(vm_5, u) = 0$. Se mesmo assim vm_5 for alocada em u, uma nova ramificação será criada na árvore (Figura 4.3(b)). Caso a vm_5 fosse Windows 7 64 bits, seriam encontrados dois níveis de similaridade (SO e versão), portanto $\alpha(vm_5, u) > 0$. Caso vm_5 seja alocada em u, sua árvore ficaria conforme a Figura 4.3(c).



(a) Cenário original

(b) Alocação de uma VM Ubuntu 13.10 64 bits

(c) Alocação de uma VM Windows 7 64 bits

Figura 4.3: Exemplo de árvore de alocação em um servidor

Para a realização da simulação, foram estabelecidos pesos fixos de 0,1 para cada nível hierárquico da árvore. O fator α é calculado somando-se esse peso até o último nível comum entre as VMs. Assim, uma VM que tenha dois níveis de similaridade com outras terá $\alpha = 0,2$. O valor de 0,1 por nível foi escolhido de forma empírica, apenas para avaliar o algoritmo proposto; em servidores reais o compartilhamento de memória é muito variável, sofrendo oscilações durante o ciclo de vida das VMs.

4.5 Conclusão

Este capítulo apresentou o algoritmo proposto para a solução do problema de alocação de máquinas virtuais considerando o compartilhamento de memória. Foi apresentado o funcionamento do algoritmo e foram feitas as considerações sobre o fator de compartilhamento de memória, índice utilizado pelo algoritmo para estimar o potencial de compartilhamento.

No próximo capítulo será apresentado o ambiente de simulação utilizado e os experimentos realizados para validação do algoritmo. Também será realizada uma análise sobre os resultados obtidos.

Capítulo 5

Experimentos Realizados

Para testar o funcionamento do algoritmo, uma série de experimentos foram realizados em um ambiente de simulação, comparando o algoritmo proposto em relação a uma abordagem padrão de alocação. Neste capítulo é apresentado o simulador utilizado bem como os experimentos realizados. Na apresentação dos resultados obtidos são feitas as considerações acerca da eficiência do algoritmo, que apresentou resultados positivos e ganhos significativos em relação ao uso eficiente de recursos.

5.1 Introdução

O capítulo anterior apresentou o algoritmo *VectorAlloc* e o fator de compartilhamento, índice proposto para considerar a possibilidade de uso compartilhado da memória. Para validação do algoritmo foram realizados experimentos que comprovaram a eficiência do mesmo.

Estes experimentos foram realizados utilizando-se um ambiente de simulação: o *CloudSim*, que está detalhado na próxima seção. A opção pelo uso de um ambiente de simulação foi motivada por alguns fatores:

- Dificuldade de acesso a um ambiente de larga escala real;
- Resultados de testes realizados em ambientes de pequena escala não poderiam ser generalizados para ambientes maiores. Os resultados obtidos com poucos servidores poderiam não se repetir na presença de vários servidores;
- Devido a sua característica dinâmica, testes em ambientes reais dificilmente poderiam ser reproduzidos;
- Existência de trabalhos de pesquisa que utilizam o mesmo simulador, provando os benefícios de sua aplicação.

Neste capítulo é apresentado o ambiente de simulação utilizado para os testes e validação do algoritmo e os experimentos realizados sobre ele. O capítulo está organizado da seguinte maneira: a seção 5.2 apresenta o simulador de nuvens *Cloudsim* e os ajustes que se fizeram necessários para considerar o compartilhamento de memória. A seção 5.3 apresenta os cenários testados e os experimentos realizados. Finalmente a seção 5.5 apresenta os resultados dos experimentos e as considerações sobre o algoritmo em relação aos testes executados.

5.2 O ambiente de simulação *Cloudsim*

Para avaliação do algoritmo proposto neste trabalho, foi utilizado o simulador *CloudSim* (Calheiros et al., 2011), um *framework* extensível, de código aberto, feito em *Java* e que permite a modelagem e simulação de um ambiente de computação em nuvem. O *CloudSim* foi desenvolvido pelo *The Cloud Computing and Distributed Systems (CLOUDS) Laboratory* da Universidade de Melbourne, Austrália. Com o *CloudSim* é possível modelar vários aspectos do funcionamento de uma nuvem, como a configuração de um centro de dados, nuvens federadas, cargas de trabalho dinâmicas, consumo de energia e políticas de alocação de máquinas virtuais.

O *CloudSim* apresenta uma arquitetura multi-camadas, conforme ilustrado na Figura 5.1. Na base fica a camada de simulação, que oferece suporte para modelagem e simulação de ambientes virtualizados, inclui interfaces para gerenciamento de máquinas virtuais, memória, armazenamento e largura de banda. Dispõe de recursos para o gerenciamento de aplicações e monitoramento dinâmico do estado da nuvem. Nessa camada são estabelecidas as políticas de provisionamento e alocação de VMs.

A camada do topo representa o código do usuário, que expõe as entidades básicas para definição dos *hosts* (número de máquinas e suas especificações), aplicações (número de tarefas e seus requisitos), máquinas virtuais, número de usuários e políticas de escalonamento. É nessa camada que são construídos os cenários de simulação e onde os experimentos são realizados.

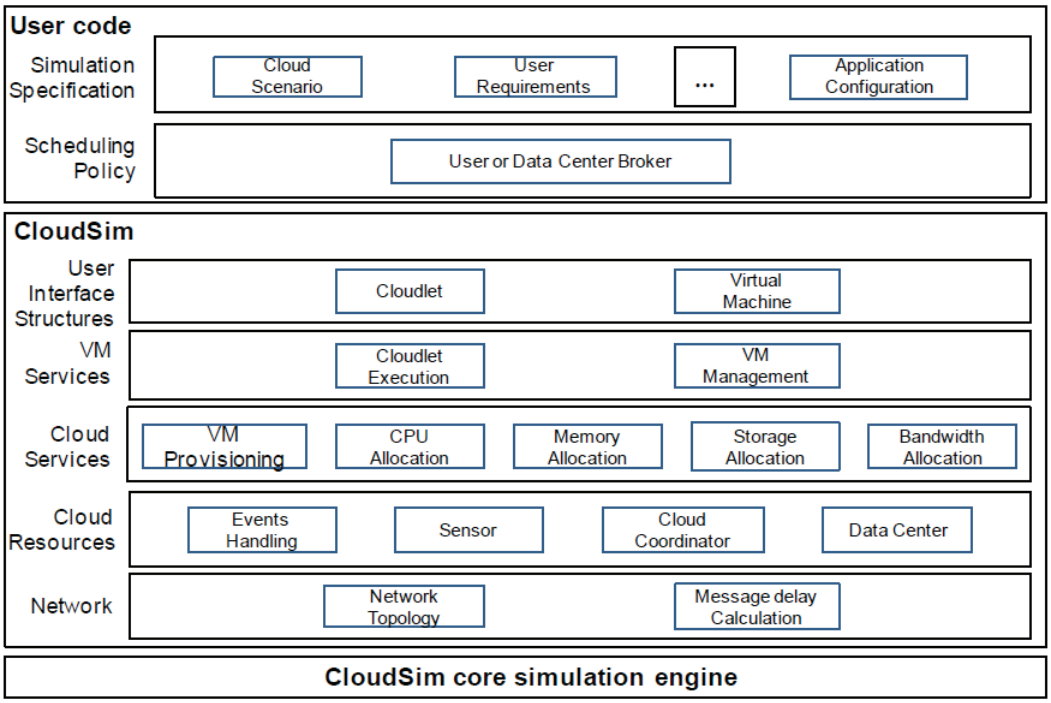


Figura 5.1: Arquitetura em camadas do *CloudSim*
(Calheiros et al., 2011)

Para que o *CloudSim* atendesse os requisitos do algoritmo proposto, foi necessário realizar algumas modificações no *framework*. A Figura 5.2 apresenta parte do diagrama de classes, sendo visíveis apenas as classes de maior relevância; as classes em fundo cinza foram incluídas ao *framework* neste trabalho, para dar suporte à simulação do compartilhamento de memória e aos requisitos necessários ao algoritmo.

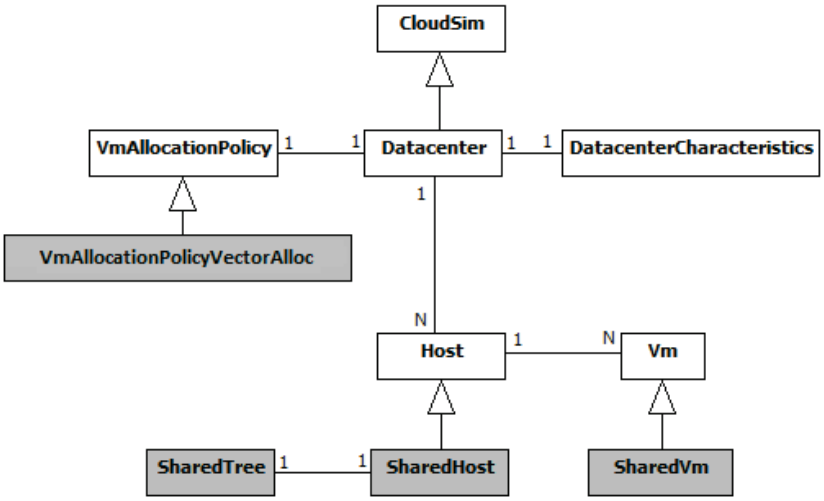


Figura 5.2: Diagrama simplificado de classes do *CloudSim*, com modificações

A classe *Host* modela as características de um servidor físico. Ela foi estendida para *SharedHost* incluindo as informações necessárias para o tratamento da memória compartilhada. Nesse caso, além de atributos para gerenciamento da memória disponível e da memória alocada, foi incluído também um atributo para a memória compartilhada. Esse atributo é atualizado à medida em que máquinas virtuais semelhantes são alocadas no mesmo servidor. O mesmo fator de compartilhamento usado no algoritmo foi utilizado para estimar uma quantidade compartilhável de memória.

A classe *Vm*, que modela as características de uma máquina virtual, foi estendida para *SharedVm*. Nessa classe foram adicionadas informações sobre o Sistema Operacional (SO) da VM. Essas informações incluem o nome, versão e arquitetura do SO, que serão utilizadas para estabelecer o grau de similaridade entre as VMs.

A nova classe *SharedTree* foi criada para modelar a árvore de alocações de VMs. Ela está presente em cada *SharedHost*. Conforme detalhado na Seção 4.4, a cada alocação realizada em um servidor, uma nova entrada na árvore é criada para a nova VM. Através dessa estrutura é possível determinar o grau de similaridade entre VMs e calcular o fator de compartilhamento α para cada servidor físico apto a receber uma nova VM.

Por fim, a classe *VmAllocationPolicyVectorAlloc* foi estendida da classe abstrata *VmAllocationPolicy*. Esta classe abstrata representa a política de alocação e distribuição das máquinas virtuais nos servidores físicos, ela fornece o método *allocateHostForVm*, que é utilizado pelo simulador para realizar as alocações. Qualquer política de alocação deve obrigatoriamente implementar esse método. Portanto, é na classe *VmAllocationPolicyVectorAlloc* que o algoritmo *VectorAlloc* foi implementado. Ao iniciar a simulação, um objeto dessa classe é responsável por receber cada VM e encontrar o melhor servidor para alocá-la.

5.3 Experimentos

Inspirados no trabalho de Lago et al. (2011), os experimentos foram modelados em três cenários distintos, simulando ambientes de centros de dados de diferentes portes. O centro

de dados de pequeno porte contém 10 servidores onde devem ser alocadas 30 máquinas virtuais. De médio porte contém 100 servidores para alocação de 300 máquinas virtuais e o centro de dados de grande porte possui 1000 servidores para 3000 máquinas virtuais.

Os servidores físicos foram configurados de modo a formar um ambiente heterogêneo, ou seja, as configurações são diferenciadas de um servidor para outro. Como este trabalho procura analisar sobretudo o uso da memória, foi esse recurso que recebeu valores diferenciados, os demais recursos foram configurados de forma idêntica. Assim, as configurações de CPU, banda de rede e armazenamento nos servidores foram fixadas, respectivamente, em 10.000 MIPS, 1 Gbps e 1 TB. Para a memória adotou-se uma distribuição circular do conjunto {12, 16, 20 e 24 GB}, simulando diferentes capacidades.

Para as máquinas virtuais, as configurações de banda de rede e de armazenamento ficaram fixas em 100 Mbps e 50 GB, com distribuição circular para a CPU em {1.000, 1.500, 2.000 e 2.500 MIPS e para a memória em 512 MB, 1, 2, 4 e 8 GB. O sistema operacional de cada VM pode ser Windows 7, Windows 8, Linux Ubuntu 10.4 ou Linux Ubuntu 12.4, todos podendo assumir a plataforma de 32 ou 64 bits.

Em relação as configurações gerais do algoritmo, adotou-se $RTVmin = 0,1$ e $RTVmax = 0,9$.

A tabela 5.1 apresenta as possíveis configurações dos servidores e das máquinas virtuais. Essa configuração é distribuída uniformemente entre os servidores e as VMs de cada cenário.

Tabela 5.1: Configurações dos servidores e VMs

Servidores		**Máquinas Virtuais**	
Recurso	**Valores**	**Recurso**	**Valores**
CPU (MIPS)	10.000	CPU (MIPS)	1000, 1500, 2000, 2500
Memória (GB)	12, 16, 20, 24	Memória (GB)	0.5, 1, 2, 4, 8
Rede (mbps)	1000	Rede (mbps)	100
Disco (GB)	1000	Disco (GB)	50
		Sistema Operacional	Windows 7 32 bits Windows 7 64 bits Windows 8 32 bits Windows 8 64 bits Ubuntu 10.4 32 bits Ubuntu 10.4 64 bits Ubuntu 12.4 32 bits Ubuntu 12.4 64 bits

Para efeito de comparação, em cada cenário foram executados três algoritmos de alocação:

a) Um algoritmo de alocação usando a técnica *First-Fit*, chamado aqui de algoritmo padrão, onde uma VM é simplesmente alocada no primeiro servidor com recurso de processamento disponível;

b) O algoritmo *VectorAlloc* sem considerar o compartilhamento de memória, ou seja, fixado com o valor de $alpha = 0$;

c) O algoritmo *VectorAlloc* considerando o compartilhamento de memória.

A alocação das VMs ocorre de forma *online*, ou seja, não há um pré-conhecimento do conjunto a ser instanciado. Assim, cada cenário parte de um estado "limpo", onde todos os recursos estão disponíveis. Ao iniciar a simulação as VMs são colocadas em uma fila de alocação, e são alocadas, uma a uma de forma sequencial, respeitando a ordem de chegada na fila.

5.4 Métricas

Para medir a eficiência do algoritmo de alocação foram coletadas uma série de dados das simulações efetuadas. Os dados apresentam o estado dos recursos físicos de cada servidor após o processo de alocações. A partir desses dados foi possível determinar algumas métricas para realização das análises.

Uma das métricas é o *percentual de utilização dos recursos*, onde foi observado se a alocação pelo algoritmo resulta em uma melhor distribuição do uso dos recursos físicos dos servidores, ou seja, se as VMs foram distribuídas de modo a não criarem uma situação de excesso de carga ou de subutilização de recursos.

Outro parâmetro utilizado nesse trabalho é tratado como *gap entre recursos*. Define-se por *gap* o módulo da diferença entre o percentual de utilização de dois recursos. Nessa pesquisa foi considerado o *gap* entre o uso de memória e de CPU. Desse modo é possível avaliar se esses recursos estão sendo alocados de forma equilibrada, não sobrecarregando um recurso do servidor enquanto o outro estaria subutilizado; um *gap* elevado pode fazer com que servidores com grande disponibilidade de um recurso não possam alocar uma nova VM devido a limitação de outro recurso.

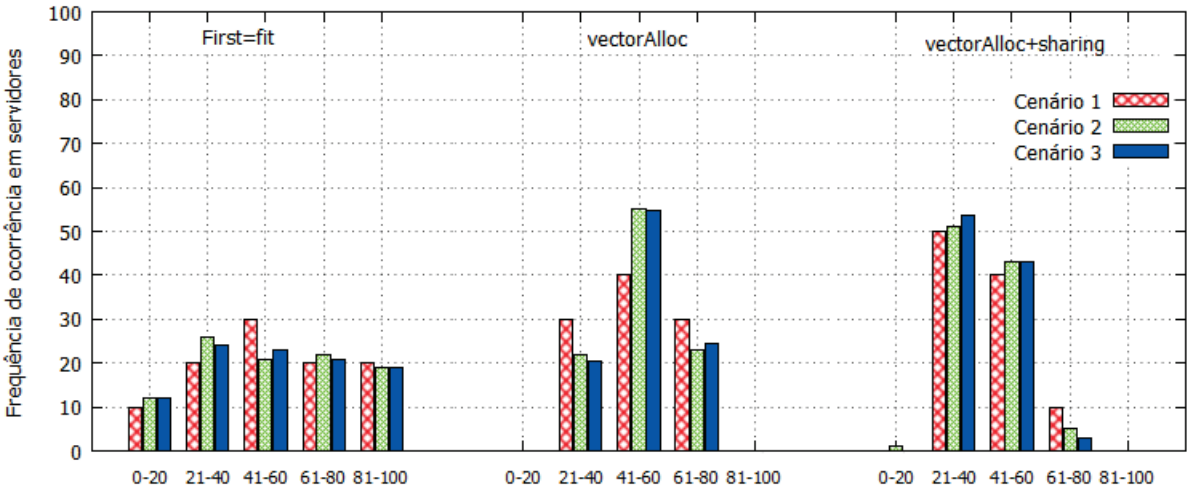
Finalmente, a *memória economizada* permite avaliar a quantidade de memória que pode ser poupada pelo algoritmo e se essa economia impactou no uso dos outros recursos, em especial a CPU. Ao se disponibilizar mais memória para as alocações em virtude do compartilhamento de páginas de memória, tem-se por consequência uma utilização maior dos demais recursos. A preocupação aqui continua sendo manter o uso equilibrado dos recursos.

5.5 Resultados

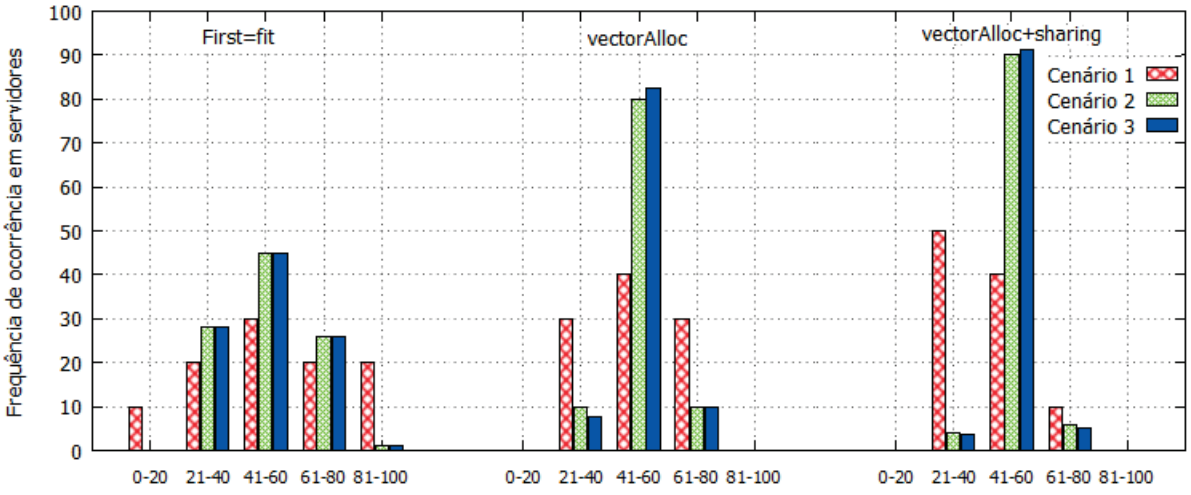
A Figura 5.3 apresenta o percentual de utilização dos recursos de memória (a) e CPU (b) após a execução de cada algoritmo nos três cenários. O eixo X apresenta os intervalos do percentual de utilização do recurso e o eixo Y a frequência com que o intervalo ocorre.

É possível perceber que no algoritmo padrão o uso da memória está desequilibrado, atingindo inclusive extremos de uso: cerca de 10% dos servidores tiveram um uso de memória abaixo de 20%, enquanto outros 20% tiveram seu uso de memória entre 81% e 100%. No *VectorAlloc* tal situação não ocorre e o uso do recurso está equilibrado, concentrado na faixa central.

Na execução sem compartilhamento de memória, os cenários apontaram que cerca de 55% dos servidores tiveram um percentual de utilização entre 40% e 60%. O restante ficou dividido entre os intervalos vizinhos. Não houveram casos de utilização abaixo de 20% nem acima de 80%. Na execução com compartilhamento de memória os servidores em geral tiveram a utilização dos recursos reduzida. Os cenários apresentaram uma redução dos intervalos mais



(a) Uso de memória



(b) Uso de CPU

Figura 5.3: Uso de recursos após alocações

altos e aumento dos intervalos mais baixos, com alguma utilização de memória abaixo de 20%, mas sem casos de utilização de recursos acima de 80%.

O uso de recursos de CPU seguiu a mesma tendência, com o algoritmo *First-Fit* o uso ficou bem distribuído entre as várias faixas de utilização do recurso, para o *VectorAlloc* sem compartilhamento e com compartilhamento, o uso ficou em sua maioria concentrado na faixa central de utilização.

Em relação ao *gap* existente entre a utilização de recursos de memória e de CPU, novamente o algoritmo *VectorAlloc* mostra-se mais equilibrado. A Figura 5.4 ilustra essa situação. No algoritmo padrão, entre 70% e 80% das alocações criaram um *gap* de até 40%, mas nos ambientes de médio e grande porte, quase 10% dos servidores atingiram uma diferença entre 61% e 80%. Percebe-se que nessas situações um recurso está sendo muito sobrecarregado em relação ao outro. O *VectorAlloc* sem compartilhamento reduziu muito essa diferença, deixando em torno de 85% dos servidores com um *gap* inferior a 20%, o que indica uso equilibrado dos recursos. Com o compartilhamento de memória esse *gap* volta a ter um aumento, mas isso é

explicado pelo fato do uso de CPU se manter o mesmo, enquanto o uso de memória foi reduzido pela possibilidade de compartilhamento.

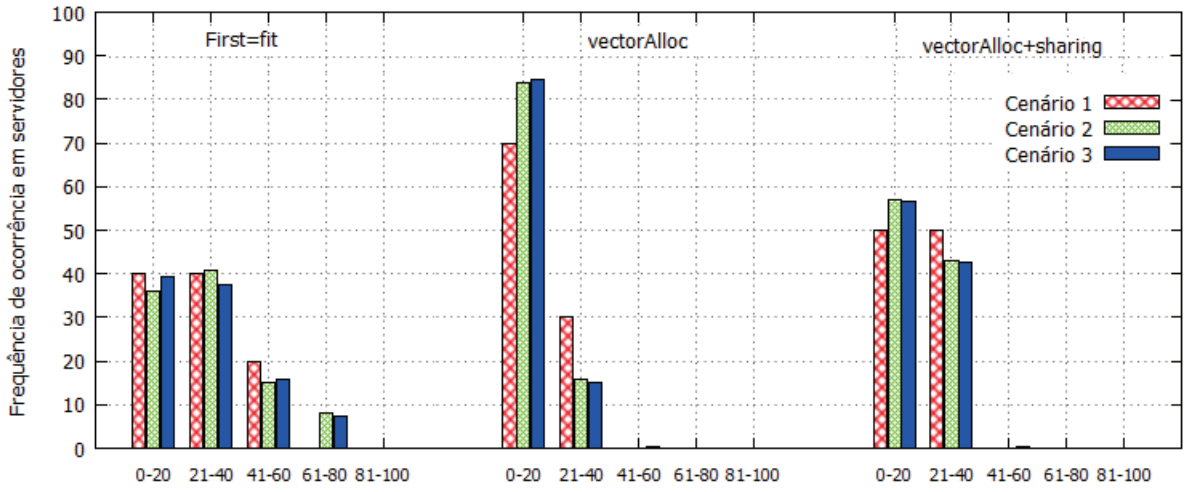


Figura 5.4: Gap entre uso da memória e de CPU

Os últimos resultados obtidos aparecem na Tabela 5.2 e mostram o uso médio de memória nos servidores, para cada algoritmo executado. Dos três algoritmos, o *VectorAlloc* com compartilhamento de memória gerou os melhores resultados, resultando numa média de utilização de memória em torno de 38%, cerca de 15% menor que o algoritmo padrão e 12,5% menor que o *VectorAlloc* sem compartilhamento de memória (coluna *ganho*).

Tabela 5.2: Média e desvio padrão de uso da memória dos servidores

cenário	*First-Fit*		*VectorAlloc*			*VectorAlloc + mem sharing*		
	média	desvio	média	desvio	ganho	média	desvio	ganho
1	58,7%	27,9%	53,5%	12,0%	5.2%	43,9%	12,9%	14.8%
2	53,5%	27,2%	50,7%	12,6%	2.8%	38,2%	13,3%	15.3%
3	53,6%	27,1%	50,8%	12,4%	2.8%	38,0%	13,3%	15.6%

5.6 Conclusão

Este capítulo apresentou o ambiente de simulação *CloudSim* e os ajustes necessários para a execução do algoritmo *VectorAlloc*. Foram modelados três diferentes cenários e os experimentos foram realizados utilizando três algoritmos.

Os resultados da execução de cada algoritmo, para cada cenário, foram comparados utilizando os critérios de uso de recursos, *gap* entre CPU e memória e o uso médio de memória. Através destes fatores, foi percebido que o algoritmo *VectorAlloc* trouxe um ganho significativo no uso equilibrado dos recursos e ofereceu ganhos em relação ao uso da memória.

No próximo capítulo serão apresentadas as considerações finais, onde serão destacadas as contribuições da pesquisa e propostos trabalhos futuros.

Capítulo 6

Conclusão

A presente pesquisa abordou o problema de alocação de máquinas virtuais em ambientes de computação em nuvem. A literatura apresenta mecanismos e algoritmos para resolver esse problema, com diferentes objetivos e usando os mais variados critérios. Nesta pesquisa foi proposto um algoritmo que utiliza como critério de alocação as múltiplas dimensões de recursos de um servidor físicos: CPU, memória, disco e rede. Além disso, o algoritmo considera o potencial de compartilhamento de memória que existe entre as máquinas virtuais quando alocadas em um mesmo servidor físico.

O algoritmo, denominado *VectorAlloc*, utiliza uma representação vetorial para modelar o uso e os requisitos de recurso, onde através destes vetores é determinada a melhor alocação.

Outra contribuição da pesquisa foi a proposta de um fator de compartilhamento, utilizado para determinar o potencial de compartilhamento de memória entre VMs em um mesmo servidor. Nessa pesquisa o fator de compartilhamento levou em consideração o grau de similaridade entre as VMs, sendo determinado através das árvores de alocação.

O algoritmo foi testado em três diferentes cenários, modelados em um simulador de infraestrutura de computação em nuvem. Foram simuladas alocações utilizando três diferentes algoritmos: um algoritmo padrão de alocação usando a heurística *first-fit*, o *VectorAlloc* sem considerar o compartilhamento de memória e o *VectorAlloc* considerando o compartilhamento. A ideia de usar o *VectorAlloc* em dois modos distintos (com compartilhamento e sem compartilhamento) foi para determinar o real impacto do fator de compartilhamento.

Através dos resultados obtidos, foi mostrado que o *VectorAlloc* teve um ganho significativo na questão do uso equilibrado de recursos, não causando casos de sobrecarga ou subutilização. Essa característica também se observa ao analisar o *gap* entre o uso de CPU e de memória, onde grande parte das alocações apresentaram valores de *gap* reduzidos, o que indica que nenhum recurso foi sobrecarregado em detrimento de outro.

Como trabalhos futuros, podem ser feitas análises de desempenho do algoritmo e implementações de melhorias visando otimizar as alocações. Outra proposta seria a de ajustar o algoritmo para uma solução de balanceamento de carga, a qual trabalharia a partir de máquinas virtuais já hospedadas que seriam realocadas.

O fator de compartilhamento também se apresenta como uma interessante área para estudo. Nessa pesquisa os índices utilizados foram determinados através das características do sistema operacional da VM. Além dessas características, também poderia levar-se em consideração o conjunto de aplicações instaladas na VM, seja para efeito de desempate, caso duas VMs

possuam o mesmo sistema operacional, ou nas situações em que os sistemas operacionais sejam diferentes mas executam as mesmas aplicações.

Ainda em relação ao fator de compartilhamento, a pesquisa utilizou-se de valores empíricos para formação do índice, porém estudos mais aprofundados deveriam ser realizados para determinar valores mais precisos e próximos da realidade. Por exemplo, através de análises históricas de alocação determinariam-se índices mais apropriados a cada caso.

Referências Bibliográficas

Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities. *IEEE Data Eng. Bull.*, 32(1):3–12.

AlZain, M. A., Pardede, E., Soh, B., and Thom, J. A. (2012). Cloud computing security: From single to multi-clouds. *45th Hawaii International Conference on System Sciences*, pages 5490–5499.

Amarante, S. R. M. (2013). Utilizando o problema de múltiplas mochilas para modelar o problema de alocação de máquinas virtuais em computação nas nuvens. Master's thesis, Universidade Estadual do Ceará, Fortaleza - CE.

Arcangeli, A., Eidus, I., and Wright, C. (2009). Increasing memory density by using KSM. *Proceedings of the linux symposium (OLS'09)*, pages 19–28.

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Azevedo, M. V., Monteiro, A. F., and Sztjanberg, A. (2011). Resource optimization and energy saving in clusters using virtualization. In *Dependable Computing (LADC), 2011 5th Latin-American Symposium on*, pages 74–83. Ieee.

Barker, S., Wood, T., Shenoy, P., and Sitaraman, R. (2012). An empirical study of memory sharing in virtual machines. *Usenix ATC*.

Beloglazov, A. (2013). *Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing*. PhD thesis, The University of Melbourne.

Beloglazov, A. and Buyya, R. (2010). Energy efficient allocation of virtual machines in cloud data centers. *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 577–578.

Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, (Vm):1–12.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599–616.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. a. F., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Camati, R. S. (2013). Novos algoritmos para alocação de máquinas virtuais e um novo método de avaliação de desempenho. Master's thesis, Pontifícia Universidade Católica do Paraná, Curitiba - PR.

Chang, C.-R., Wu, J.-J., and Liu, P. (2011). An empirical study on memory sharing of virtual machines for server consolidation. In *Proceedings of the 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications*, ISPA '11, pages 244–249, Washington, DC, USA. IEEE Computer Society.

Chen, L., Wei, Z., Cui, Z., Chen, M., Pan, H., and Bao, Y. (2014). Cmd: Classification-based memory deduplication through page access characteristics. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '14, pages 65–76, New York, NY, USA. ACM.

Creasy, R. J. (1981). The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490.

Dhiman, G., Marchetti, G., and Rosing, T. (2010). vgreen: A system for energy-efficient management of virtual machines. *ACM Trans. Des. Autom. Electron. Syst.*, 16(1):6:1–6:27.

el Khameesy, N. and Rahman, H. A. (2012). A proposed model for enhancing data storage security in cloud computing systems. *Journal of Emerging Trends in Computing and Information Sciences*, 3(6):970–974.

Eucalyptus (2014). The eucalyptus cloud. `http://www.eucalyptus.com`. Acessado em 01/06/2014.

Fakhim, B., Behnia, M., Armfield, S., and Srinarayana, N. (2011). Cooling solutions in an operational data centre: A case study. *Applied Thermal Engineering*, 31(14–15):2279 – 2291.

Garfinkel, T. and Rosenblum, M. (2005). When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10*, HOTOS'05, pages 20–20, Berkeley, CA, USA. USENIX Association.

Garg, S. K. and Buyya, R. (2012). *Harnessing Green IT: Principles and Practices*, chapter Green Cloud Computing and Environmental Sustainability, pages 315–340. Wiley Press, UK.

Grit, L., Irwin, D., Yumerefendi, A., and Chase, J. (2006). Virtual machine hosting for networked clusters: Building the foundations for "autonomic"orchestration. *First International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*, (Vtdc):7–7.

Gupta, D., Lee, S., Vrable, M., Savage, S., Snoeren, A. C., Varghese, G., Voelker, G. M., and Vahdat, A. (2010). Difference engine: Harnessing memory redundancy in virtual machines. *Commun. ACM*, 53(10):85–93.

Hayes, B. (2008). Cloud computing. *Communications of the ACM*, 51(7):9–11.

Hyser, C., Mckee, B., Gardner, R., and Watson, B. (2007). Autonomic virtual machine placement in the data center. Technical report, HP Laboratories, Palo Alto.

Johnson, D. S. (1974). Fast algorithms for bin packing. *J. Comput. Syst. Sci.*, 8(3):272–314.

Kepes, B. (2011). Understanding the cloud computing stack: Paas, saas, iaas.

Kolster, J. F., Kristensen, J., and Mejlholm, A. (2006). Efficient memory sharing in the xen virtual machine monitor. Technical Report January, Department of Computer Science - Aalborg University.

Lago, D., Madeira, E., and Bittencourt, L. (2011). Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. In *9th International Workshop on Middleware for Grids, Clouds and e-Science*, New York USA. ACM Press.

Laureano, M. A. P. and Maziero, C. A. (2008). Virtualização: Conceitos e aplicações em segurança. In *Livro-Texto de Minicursos SBSeg*, chapter 4, pages 1–50.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

Miller, K., Franz, F., Rittinghaus, M., Hillenbrand, M., and Bellosa, F. (2013). Xlh: More effective memory deduplication scanners through cross-layer hints. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference*, USENIX ATC'13, pages 279–290, Berkeley, CA, USA. USENIX Association.

Mills, K., Filliben, J., and Dabrowski, C. (2011). Comparing vm-placement algorithms for on-demand clouds. *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pages 91–98.

Mishra, M. and Sahoo, A. (2011). On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. *2011 IEEE 4th International Conference on Cloud Computing*, pages 275–282.

Moreno-Vozmediano, R., Montero, R. S., and Lloente, I. M. (2012). Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45:65–72.

Nelson, M., Lim, B.-H., and Hutchins, G. (2005). Fast transparent migration for virtual machines. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pages 391–394, Berkeley, CA, USA. USENIX Association.

Nguyen Van, H., Dang Tran, F., and Menaud, J.-M. (2009). Autonomic virtual resource management for service hosting platforms. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 1–8, Washington, DC, USA. IEEE Computer Society.

Nimbus (2014). Nimbus project. `http://www.nimbusproject.org`. Acessado em 28/05/2014.

OpenNebula (2014). Opennebula.org: The open source solution for data center virtualization. `http://opennebula.org`. Acessado em 01/06/2014.

OpenStack (2014). Openstack cloud software. `http://www.openstack.org`. Acessado em 28/05/2014.

Paul, I., Yalamanchili, S., and John, L. K. (2012). Performance impact of virtual machine placement in a datacenter. *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*, pages 424–431.

Rosenblum, M. and Garfinkel, T. (2005). Virtual machine monitors: Current technology and future trends. *Computer*, (May):39–47.

Seiden, S. S. (2002). On the online bin packing problem. *J. ACM*, 49(5):640–671.

Sindelar, M., Sitaraman, R. K., and Shenoy, P. (2011). Sharing-aware algorithms for virtual machine colocation. *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures - SPAA '11*, page 367.

Singh, A., Korupolu, M., and Mohapatra, D. (2008). Server-storage virtualization: Integration and load balancing in data centers. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12.

Smith, J. E. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2009). A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.

Voorsluys, W., Broberg, J., Venugopal, S., and Buyya, R. (2009). Cost of virtual machine live migration in clouds: A performance evaluation. *Cloud Computing*, pages 1–12.

Waldspurger, C. A. (2002). Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194.

Wood, T., Tarasuk-Levin, G., Shenoy, P., Desnoyers, P., Cecchet, E., and Corner, M. D. (2009). Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 31–40, New York, NY, USA. ACM.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.