**UNIVERSIDADE DO ESTADO DE SANTA CATARINA**
**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**RICARDO JOSÉ PFITSCHER**

# DIAGNÓSTICO DO PROVISIONAMENTO DE RECURSOS PARA MÁQUINAS VIRTUAIS EM NUVENS IAAS

Joinville, SC

**2014**

**RICARDO JOSÉ PFITSCHER**

# DIAGNÓSTICO DO PROVISIONAMENTO DE RECURSOS PARA MÁQUINAS VIRTUAIS EM NUVENS IAAS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada, do Centro de Ciências Tecnológicas, da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Computação.

Orientador: Prof. Dr. Mauricio Aronne Pillon
Coorientador: Prof. Dr. Rafael Rodrigues Obelheiro

**JOINVILLE, SC**
**2014**

P531d Pfitscher, Ricardo José
Diagnóstico de provisionamento de recursos para máquinas virtuais em nuvens IaaS / Ricardo José Pfitscher. Joinville – 2014.
128 p. : il.; 21 cm

Orientador: Mauricio Aronne Pillon
Co-orientador: Rafael Rodrigues Obelheiro
Bibliografia: p. 117-124
Dissertação (mestrado) – Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2014.

1. Computação aplicada. 2. Sistemas de computação. 3. Redes de computadores e sistemas distribuídos. 4. Nuvens computacionais. 5. Provisionamento de recursos I. Pillon, Mauricio Aronne. II. Obelheiro, Rafael Rodrigues. III. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada. IV. Título.

CDD: 004.2 – 20.ed.

**RICARDO JOSÉ PFITSCHER**

**DIAGNÓSTICO DO PROVISIONAMENTO DE RECURSOS PARA MÁQUINAS VIRTUAIS EM MÁQUINAS EM NUVENS IAAS**

Dissertação apresentada ao Curso de Mestrado Acadêmico Computação Aplicada como requisito parcial para obtenção do título de Mestre em Computação Aplicada na área de concentração "Ciência da Computação".

**Banca Examinadora**

Orientador:

Prof. Dr. Maurício Aronne Pillon
CCT/UDESC

**Membros**

Prof. Dr. Rafael Rodrigues Obelheiro
CCT/UDESC

Prof. Dr. Rômulo Silva de Oliveira
UFSC

Prof. Dr. Guilherme Piêgas Koslovski
CCT/UDESC

**Joinville,SC, 18 de fevereiro de 2014.**

Dedico este trabalho à minha Família e à minha amada Helena.

## AGRADECIMENTOS

Para desenvolvimento deste trabalho algumas pessoas se destacam e tornam se merecedoras de reconhecimento, dentre elas podemos citar professores, amigos e familiares:

À minha mãe Elisete, pelo carinho, atenção e educação que a mim foram dedicados, somente com estes fui capaz de concluir mais esta etapa em minha vida.

Ao meu pai Paulo César, pelas palavras de força, pelo orgulho demonstrado, pela educação, e pelo carinho a mim dedicado.

Ao meu irmão Paulo Henrique, pelo incentivo, pelo apoio, e principalmente pelo afeto dedicado.

Ao meu irmão Pedro, pela atenção, pelo afeto, e principalmente por suportar a ausência do irmão e amigo.

Aos meus familiares pelo apoio e dedicação para com minha pessoa, e a força com a qual pude enfrentar este desafio.

Aos meus amigos que me animaram nos momentos de desânimo.

À minha amada esposa, Helena, pela paciência de suportar a minha ausência durante a execução deste trabalho, pelas horas de conforto, afeto e carinho, e pelas inúmeras horas destinadas a ajudar na concepção do mesmo.

Aos meus orientadores e professores, Maurício e Rafael, pelo empenho dedicado na concepção deste trabalho, pelos conselhos de vida e pela orientação. A participação deles foi fundamental nesta etapa de minha vida.

“A ciência nunca resolve um problema sem criar pelo menos outros dez.”

George Bernard Shaw

# RESUMO

PFITSCHER, Ricardo José. **Diagnóstico do provisionamento de recursos para máquinas virtuais em nuvens IaaS**. 2014. 128 f. Dissertação (Mestrado em Computação Aplicada - Área: Sistemas Computacionais) – Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada, Joinville, 2014.

A computação em nuvem trouxe recentemente uma pequena revolução na tecnologia da informação. Neste paradigma, provedores de nuvem assumem o ônus de adquirir, operar e gerenciar infraestruturas computacionais, que são então locadas a clientes de nuvem. Um dos modelos de serviço de nuvens é infraestrutura como serviço (*infrastructure-as-a-service*, IaaS), no qual provedores oferecem como recursos capacidades de processamento, armazenamento e comunicação, normalmente associadas a instâncias de máquinas virtuais. Ainda que clientes IaaS possam ajustar os recursos alocados de acordo com as suas necessidades – uma característica chave das nuvens –, o dimensionamento de recursos permanece um desafio, uma vez que faltam meios para identificar com precisão a capacidade necessária para satisfazer a demanda das aplicações. Muitos usuários incorrem no subprovisionamento, prejudicando o desempenho de suas aplicações, ou no superprovisionamento, pagando por recursos que não são realmente necessários. Este trabalho usa monitoração para prover um modelo de diagnóstico, permitindo que um cliente de nuvem determine se os recursos disponíveis para suas máquinas virtuais estão provisionados corretamente, ou se estão sub- ou superprovisionados. Foram avaliados os recursos de processador, memória e rede, que podem ser facilmente reservados nos ambientes de virtualização atuais. A eficácia da abordagem proposta é demonstrada por resultados experimentais em máquinas virtuais Linux na plataforma de virtualização Xen.

**Palavras-chave:** Computação em Nuvem. Virtualização. Diagnóstico de Provisionamento.

# ABSTRACT

PFITSCHER, Ricardo José. **Diagnosing resource provisioning for virtual machines in IaaS clouds**. 2014. 128 p. Dissertation (Masters Degree in Applied Computation - Field: Computer Systems) – Santa Catarina State University. Graduate Program in Applied Computer, Joinville, 2014.

Cloud computing has recently brought about a small revolution in information technology. In this paradigm, cloud providers assume the burden of acquiring, operating, and managing computing infrastructures, which are then rented to cloud customers. One of the cloud service models is infrastructure-as-a-service (IaaS), where providers offer processing, storage and communication capacity as resources, normally associated to virtual machine instances. Even if IaaS customers are able to adjust resource allocation to their needs – a key cloud feature –, sizing resources remains a challenge due to a lack of means to accurately identify how much capacity is needed to satisfy application demands. Many users end up underprovisioning, hurting application performance, or overprovisioning, paying for resources that are not really necessary. Our work introduces a diagnosis model that uses monitoring to enable a cloud customer to determine if the resources available to his virtual machines are correctly provisioned, or are under-/overprovisioned. We evaluated processor, memory and network resources, which can be easily provisioned in current virtualization environments. Experimental results with the Xen platform demonstrate the effectiveness of the proposed approach.

**Keywords:** Cloud Computing. Virtualization. Provisioning Diagnosis.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| CV | Coeficiente de Variação |
| E/S | Entrada e Saída |
| IaaS | *Infraestructure as a Service* |
| LRU | *Least Recently Used* |
| MMV | Monitor de Máquinas Virtuais |
| MV | Máquina Virtual |
| NIST | National Institute of Standards and Technology |
| PaaS | *Plataform as a Service* |
| QoS | *Quality of Service* |
| RAM | *Random Access Memory* |
| RUBiS | *Rice University Bidding System* |
| SaaS | *Software as a Service* |
| SLA | *Service Level Agreement* |
| SLI | *Service Level Indicator* |
| SLO | *Service Level Objective* |
| SO | Sistema Operacional |
| SSH | *Secure Shell* |
| VCPU | *Virtual CPU* |
| WSS | *Working Set Size* |

# LISTA DE SÍMBOLOS

| | |
|---|---|
| $CV_{lb}$ | Coeficiente de variação da largura de banda |
| $\overline{m_c}$ | Média de memória reservada |
| $\overline{m_r}$ | Média de memória residente |
| $m_c$ | Memória reservada |
| $m_r$ | Memória residente |
| $U_{cpu}$ | Utilização de CPU |
| $\overline{U_{cpu}}$ | Média da Utilização de CPU |
| $N'$ | Número de pontos com CPU em saturação |
| $\alpha$ | Limiar de memória residente alta |
| $\beta$ | Limiar de memória residente baixa |
| $\gamma$ | Limiar de memória reservada relevante |
| $\theta$ | Limiar de saturação de CPU |
| $\varphi$ | Percentual de pontos em saturação |
| $\omega$ | Limiar para o percentual de pontos em saturação |

**SUMÁRIO**

# 1 INTRODUÇÃO

Uma das tecnologias de maior crescimento e discussão na atualidade é a computação em nuvem, muitas organizações tem transferido suas estruturas de tecnologia de informação para as nuvens com o objetivo de otimizar os investimentos e ampliar o acesso as informações. As nuvens trazem como grandes atrativos a elasticidade de recursos e a tarifação por recursos alocados (*pay-per-use*), permitindo que os custos fixos de infraestrutura computacional do cliente sejam transferidos para um provedor (ARMBRUST et al., 2010). Os recursos computacionais são oferecidos como serviços, seja na forma de aplicações, de plataformas de desenvolvimento ou em infraestruturas computacionais. Provedores de *infrastructure-as-a-service* (IaaS) oferecem como recursos máquinas virtuais (processador, memória, disco) e largura de banda de rede, que podem ser alocados conforme a necessidade do cliente (SULEIMAN et al., 2012). Entretanto, um dos desafios pertinentes a este novo paradigma envolve o provisionamento adequado dos recursos computacionais necessários para atender às demandas dos clientes-usuários.

Em uma solução IaaS, usuário e provedor possuem o mesmo objetivo geral, que é o de minimizar custos, mas diferem no modo de atingir esse objetivo. O usuário obtém um custo mais baixo alocando o mínimo de recursos suficientes para obter um desempenho aceitável. De outro lado, o provedor está interessado em maximizar a ocupação dos seus recursos físicos desde que isso não comprometa a qualidade dos serviços oferecidos. Quando a demanda por recursos é variável, e não constante, o conflito entre essas estratégias se torna evidente: se um recurso físico estiver com toda a sua capacidade alocada para clientes, ele será incapaz de acomodar picos de carga sem prejudicar o desempenho das máquinas virtuais. Em contrapartida, a existência de capacidade ociosa (reservada mas não usada) vai de encontro ao interesse do provedor.

Geralmente, o equilíbrio entre os interesses de clientes e provedores é estabelecido por acordos de nível de serviço (SLAs – *Service Level Agreements*), que definem métricas (SLIs – *Service Level Indica-*

*tors*) e respectivos valores-objetivo (SLOs – *Service Level Objectives*) para que o desempenho dos recursos alocados seja considerado satisfatório (SAUVÉ et al., 2005). Muitos trabalhos de pesquisa que propõem estratégias para melhorar a alocação de recursos em ambientes IaaS, tais como (BUYYA; GARG; CALHEIROS, 2011; BELOGLAZOV; BUYYA, 2010; KUNDU et al., 2012), tomam a existência de SLAs como uma premissa. Essa premissa embute um outro pressuposto, o de que o cliente é capaz de identificar corretamente as suas necessidades, e expressá-las em termos de SLIs e SLOs. Na prática, porém, fixar SLOs que garantam o desempenho desejado, a um custo aceitável, para as aplicações que serão executadas na nuvem não é trivial, haja vista a falta de diretrizes confiáveis para a especificação dos parâmetros. Logo, muitos usuários acabam sub- ou superdimensionando os seus objetivos, e consequentemente os seus recursos.

Quando recursos são subprovisionados, o usuário é capaz de perceber que o desempenho está aquém do esperado, mas muitas vezes não consegue dizer quais recursos precisam de mais capacidade. Por outro lado, o efeito do superprovisionamento de recursos é percebido menos pelo desempenho e mais pelo custo financeiro. A ausência de parâmetros de comparação da relação custo $\times$ aplicação (que permitam constatar que aplicações similares conseguem obter desempenho satisfatório a um custo mais baixo) e a dificuldade em conhecer a capacidade suficiente para cada recurso (quais SLOs podem ser reduzidos?) tornam ainda mais complexa a questão.

Tendo em vista esta problemática, este trabalho apresenta um mecanismo para facilitar o diagnóstico da alocação dos recursos. São avaliados processador, memória e rede, que podem facilmente ser reservados nos monitores de máquinas virtuais atuais e também podem ser objeto de SLA em alguns provedores IaaS (SULEIMAN et al., 2012). As métricas de desempenho são todas coletadas dentro das máquinas virtuais, sendo portanto diretamente acessíveis ao usuário, sem a necessidade de intermediação do provedor. Além disso, elas refletem o desempenho efetivamente observado nas máquinas virtuais, considerando a influência tanto da capacidade do *hardware* subjacente como do compartilhamento desse *hardware* com outras máquinas virtuais (MV). As avaliações experimentais são conduzidas com Linux

na plataforma Xen,[1] um dos ambientes de virtualização mais usados por provedores IaaS, sendo a principal solução de código aberto desse mercado (BUTLER, 2012).

## 1.1 OBJETIVOS

O objetivo principal deste trabalho é fornecer ao cliente um diagnóstico sobre o provisionamento dos recursos para máquinas virtuais em nuvens IaaS, indicando se a alocação de recursos para a máquina virtual está adequada, subprovisionada ou superprovisionada.

Para alcançar este objetivo alguns objetivos específicos são elencados:

- Identificar métricas representativas de situação de provisionamento para cada um dos recursos;
- Definir limiares e valores, associados às métricas, que caracterizem subprovisionamento, provisionamento adequado e superprovisionamento;
- Propor um modelo de diagnóstico com base nas métricas e limiares para cada recurso;
- Avaliar experimentalmente o modelo sobre cada um dos recursos isoladamente; e;
- Avaliar o impacto sobre o modelo quando os recursos podem influenciar uns sobre os outros.

## 1.2 CARACTERIZAÇÃO METODOLÓGICA DA PESQUISA

A caracterização metodológica da pesquisa será realizada em três macro-classificações: ciência, objetivo e variáveis. A ideia é agrupar subclassificações semelhantes de acordo com estas macro-classificações.

[1] `http://www.xen.org/`

A primeira subclassificação em termos de ciência está relacionada à área. Segundo Marconi e Lakatos (2010), existe uma hierarquia de áreas, cujas raízes são as formais e factuais. A formal se divide em lógica e matemática, e a factual em natural e social. Assim, o trabalho é categorizado como *formal* na classificação de (MARCONI; LAKATOS, 2010), uma vez que a pesquisa lida com conceitos lógicos, matemáticos e exatos.

A segunda subclassificação define o tipo de ciência e segue a categorização feita em (WAZLAWICK, 2010). As ciências formais estão associadas ao estudo das ideias, enquanto as empíricas ao estudo das coisas (fenômenos reais, interações, fatos). As ciências puras são associadas ao estudo do conhecimento e suas bases, enquanto as aplicadas com a descoberta e sua respectiva aplicabilidade. Nas ciências exatas os resultados são predizíveis e precisos, enquanto que nas inexatas os resultados são imprevisíveis, normalmente estas pesquisas são associadas a fenômenos. As ciências duras caraterizam-se pelo rigor científico, por sua vez, as moles podem-se utilizar de dados específicos, ou episodiais. As pesquisas com característica nomotética são aquelas que estudam fenômenos repetitivos, em contrapartida, as idiográficas analisam fenômenos únicos.

Sendo assim, a pesquisa realizada neste trabalho é definida como: *empírica*, pois faz-se o estudo do comportamento da utilização dos recursos que é um fato, ou fenômeno real; *aplicada*, o resultado do estudo objetiva a implantação em ambientes reais; *exata*, os resultados são precisos e correspondentes aos estímulos; *dura*, a avaliação do comportamento e dos resultados segue um processo sistemático, e; *nomotética*, o diagnóstico sobre os recursos tende a se repetir em diferentes cenários.

Na classificação quanto ao objetivo, uma pesquisa pode ser categorizada como exploratória, descritiva ou explicativa (GIL, 2002; VIEIRA, 2010). A pesquisa exploratória envolve o entendimento de um problema. A descritiva associa comportamentos e fenômenos, mas não busca explicar o porque daquele acontecimento. Na pesquisa explicativa o foco é a descrição do comportamento, investigando os detalhes que justificam a sua ocorrência. Sendo assim, a pesquisa é categorizada como exploratória e explicativa, pois envolve o entendimento

do problema e os objetivos gerais tangem a explicação dos fenômenos (comportamentos) para o diagnóstico.

Por fim, a pesquisa pode ser classificada quanto ao procedimento de coleta das variáveis. Wazlawick (2010) divide em: pesquisa documental, onde os dados são obtidos em documentos não verificados por pesquisadores da área; pesquisa bibliográfica, onde a coleta de informações é feita em documentos que passaram por verificação e validação por autores da área; pesquisa experimental, onde é determinado um objeto de estudo e o pesquisador pode controlar as variáveis; pesquisa *ex-post facto*, semelhante à experimental diferindo quanto ao controle do pesquisador sobre as variáveis (não possui neste caso). Sendo assim este projeto de pesquisa é categorizado como *experimental* e *bibliográfico*, uma vez que tem-se total controle sobre as variáveis e muitas das hipóteses são definidas com base em estudos bibliográficos.

## 1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em cinco capítulos.

No Capítulo 1 é apresentada a introdução do problema, são descritos os objetivos do trabalho e a caracterização metodológica da pesquisa.

No Capítulo 2 faz-se uma revisão teórica sobre os conteúdos importantes ao entendimento deste trabalho, tais como, conceitos de nuvens computacionais, provisionamento de recursos em nuvem, virtualização e monitoração.

No Capítulo 3 é apresentado o modelo de diagnóstico do provisionamento de processador, memória e rede. São descritas as métricas utilizadas em cada recurso e uma sugestão de aplicação dos resultados do modelo para ajuste de alocação. O capítulo encerra com uma discussão de trabalhos relacionados.

No Capítulo 4 é realizada a verificação, através de experimentação, do modelo de diagnóstico, avaliando-o sob diferentes cenários e cargas de trabalho.

Por fim, no Capítulo 5 tem-se as considerações finais do traba-

lho e as perspectivas de trabalhos futuros.

# 2 PROVISIONAMENTO DE RECURSOS EM NUVEM

Este trabalho propõe um modelo de diagnóstico de provisionamento de recursos em nuvens IaaS. O presente capítulo revisa os conceitos necessários para o entendimento do modelo. Inicialmente a Seção 2.1 apresenta as características e definições sobre computação em nuvem. Na sequência, a Seção 2.2 discute virtualização, que é a tecnologia impulsionadora do crescimento das nuvens, e a Seção 2.3 examina os aspectos de monitoração de recursos. Finalmente, a Seção 2.4 apresenta estratégias usadas para provisionar recursos em ambientes de nuvem.

## 2.1 CONCEITOS BÁSICOS DE COMPUTAÇÃO EM NUVEM

Existem diferentes definições sobre o que é computação nas nuvens, o termo tem sido frequentemente utilizado para representar diferentes propósitos. Em casos gerais, nuvens são definidas como elementos abstratos, ou, uma área de computação na *Internet*, ou ambientes de computação com localidade transparente ao usuário. Alguns autores buscaram definir o que é uma nuvem computacional com base em um consenso entre os pesquisadores da época:

> *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.* (FOSTER et al., 2008)

Tendo em vista que existem similaridades entre definições, e que é necessário caracterizar os recursos e clientes utilizadores deste modelo, este trabalho irá basear-se na definição do NIST (National Institute of Standards and Technology) para a computação nas nuvens, que tem sido frequentemente utilizada em trabalhos acadêmicos, tais como (ACETO et al., 2013; ZHANG; CHENG; BOUTABA,

2010; HOEFER; KARAGIANNIS, 2010), definindo nuvens da seguinte forma:

> *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* (MELL; GRANCE, 2009)

Ou seja, é um ambiente onde os recursos podem se ajustar às necessidades do cliente. Estes recursos podem ser redes, servidores, armazenamento, aplicações ou serviços. Segundo o NIST, existem alguns fatores essenciais que caracterizam as nuvens:

- Auto atendimento sob demanda – Clientes podem solicitar recursos computacionais automaticamente sem interação humana, ou seja, através de interfaces de *software*.

- Acesso via rede – As capacidades computacionais são acessíveis via rede.

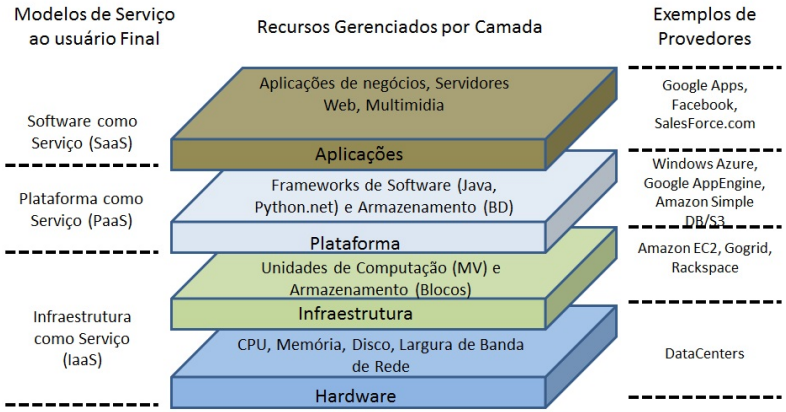
- *Pooling* de recursos – Provedores mantêm um conjunto de recursos computacionais compartilhados por seus clientes. Esses recursos são alocados de acordo com as necessidades e requisitos dos mesmos. A localidade física desses recursos é conhecida somente pelos provedores mas é transparente aos clientes.

- Elasticidade – A capacidade dos recursos computacionais alocados pode ser aumentada e diminuída a qualquer momento. Dependendo do tipo de serviço, isso pode ser feito automaticamente de acordo com as demandas.

- Serviço monitorado – Os serviços fornecidos pelo provedor podem ser acompanhados, permitindo que existam técnicas de controle e gerência de recursos com base nas métricas providas para cada recurso. Assim, provedores e clientes podem acompanhar o consumo de recursos.

As nuvens computacionais oferecem modelos de negócio que podem ser classificados em três modelos de serviço: *Software* como Serviço (SaaS, *Software as a Service*), Plataforma como Serviço (PaaS, *Plataform as a Service*) e Infraestrutura como Serviço (IaaS, *Infraestructure as a Service*) (SULEIMAN et al., 2012; ZHANG; CHENG; BOUTABA, 2010; MELL; GRANCE, 2009; HOEFER; KARAGIANNIS, 2010). A Figura 1 apresenta os modelos de serviço, os recursos providos e exemplos de fornecedores dos mesmos.

Figura 1 – Modelos de Serviço em Nuvens




Fonte: Traduzido de (ZHANG; CHENG; BOUTABA, 2010)

Sendo assim, os modelos de serviço SaaS, PaaS e IaaS são definidos como:

- **SaaS** - *Software* como um Serviço - Neste modelo de negócio os clientes utilizam aplicações de *software* do provedor. Normalmente, os acessos são realizados via navegadores web, mas é possível encontrar aplicações que utilizem de protocolos distintos para acesso remoto (por exemplo, SSH). A capacidade da aplicação aparenta ser infinita, sendo a gerência da infraestrutura de responsabilidade do provedor, que pode utilizar-se da flexibilidade dos recursos disponíveis para adequá-los à necessidade dos *softwares* fornecidos. Neste contexto, o cliente não tem conhecimento, nem gerenciamento, sobre a infraestrutura disponí-

vel.

- **PaaS** - Plataforma como um Serviço - Clientes deste modelo podem desenvolver aplicações sobre uma infraestrutura de nuvem, provedores disponibilizam *frameworks* de desenvolvimento de *software* (linguagem de programação, banco de dados, serviços, bibliotecas, etc.) sob demanda. Os recursos da camada mais baixa (redes, processador, sistema operacional) não são gerenciados pelo cliente, mas é possível ajustar as configurações de capacidade do ambiente, ou solicitar novas instâncias.

- **IaaS** - Infraestrutura como um Serviço - Provedores de IaaS oferecem recursos da camada baixa dos sistemas de computação, e clientes podem adquirir capacidades de processamento, armazenamento e comunicação. Sobre estes recursos os clientes podem ter acesso direto para instalar suas aplicações ou serviços. Nos casos mais comuns, os recursos são fornecidos com base em instâncias de máquinas virtuais configuráveis e gerenciáveis pelo cliente. Entretanto, o acesso do cliente se limita as suas instâncias virtuais, sem permissão de acesso à camada de *hardware* e ambiente de virtualização (gerenciada pelo provedor).

Tendo em vista que este trabalho objetiva prover o diagnóstico de provisionamento dos recursos em uma nuvem, e o principal interessado neste diagnóstico é o cliente, que poderá avaliar se a relação entre custo e benefício é equilibrada, o foco deste trabalho é direcionado para modelos de serviço IaaS, pois neste modelo é possível que o cliente tenha acesso à camada mais baixa de recursos computacionais providos a um ambiente virtual.

Além da classificação por modelos de serviço, é possível caracterizar as nuvens computacionais de acordo com o modelo de implantação utilizado, diferenciando-as pela abertura no acesso e pelo objetivo da nuvem. Existem quatro modelos clássicos de implantação em nuvem (ZHANG; CHENG; BOUTABA, 2010; MELL; GRANCE, 2009):

- Nuvens Privadas - Gerenciadas e providas por uma organização para a organização. Normalmente, os serviços deste modelo de

implantação são oferecidos para atender aos objetivos internos, ou seja, não são acessíveis a clientes externos. Neste tipo de nuvem tem-se o maior nível de controle sobre a segurança dos recursos, uma vez que os clientes, por serem internos, são conhecidos.

- Nuvens Comunitárias - Estas nuvens são providas por uma comunidade de usuários. Esta comunidade pode ser formada por organizações com objetivos similares ou até usuários que tenham os mesmos propósitos. Nestes casos a infraestrutura da nuvem é gerenciada pela comunidade.

- Nuvens Públicas - Um provedor de serviços disponibiliza os recursos da nuvem a um público geral. Isto não significa que o serviço é gratuito, apenas, permite o acesso a qualquer participante. Nestes casos a infraestrutura geral da nuvem é gerenciada pelo provedor de serviços.

- Nuvens Híbridas - Neste modelo de implantação tem-se a composição de dois ou mais modelos anteriores (privado, público e comunitário). Uma organização pode elaborar uma nuvem híbrida através da composição de sua nuvem privada com a utilização dos recursos de uma nuvem pública. Em determinados casos esta composição pode ser feita sob demanda. Como exemplo, tem-se o trabalho de (ASSUNÇÃO; COSTANZO; BUYYA, 2009) que avalia a utilização de nuvens públicas para estender a capacidade computacional da organização.

A proposta de diagnóstico em nuvem deste trabalho pode ser aplicada em todas as categorias de implantação (Privadas, Comunitárias, Públicas e Híbridas), desde que o cliente tenha acesso em nível de sistema operacional às instâncias alocadas, permitindo que sejam coletadas as métricas propostas. Entende-se que o maior beneficiário do modelo é o usuário de aplicações que executam em nuvens públicas, pois é onde existe uma maior preocupação da busca do equilíbrio entre custo e benefício pelos clientes, que almejam o menor custo com desempenho satisfatório. Contudo, os usuários de outras categorias de

implantação podem utilizar o modelo para racionalizar a utilização dos recursos, otimizando as alocações.

## 2.2 VIRTUALIZAÇÃO

A tecnologia de virtualização tem impulsionado o crescimento dos ambientes de computação em nuvens. Em termos de nuvem computacional a virtualização é vista como uma camada de *software* inserida entre o *hardware* e o sistema operacional, permitindo o compartilhamento dos recursos entre os domínios virtualizados. Desta maneira, diferentes clientes de um provedor de nuvens computacionais podem utilizar MVs isoladas, ou seja, o cliente A não tem conhecimento da existência do cliente B, mesmo quando ambos compartilham do mesmo recurso físico subjacente.

Na virtualização os recursos físicos compartilhados pelos diferentes domínios são abstraídos aos sistemas operacionais convidados por um monitor de máquinas virtuais (MMV), também denominado *hypervisor*. A abstração dos recursos pelo MMV pode ser feita em níveis diferentes do sistema computacional: nível de *hardware*, sistema operacional ou aplicação. No primeiro caso, o MMV é inserido entre o *hardware* e o sistema operacional da máquina nativa, provendo ao SO convidado a abstração de um novo *hardware*. Em uma analogia com a relação entre processos e o sistema operacional, neste nível de abstração as MVs podem ser vistas como processos, enquanto o MMV é visto como o sistema operacional (TANENBAUM, 2007).

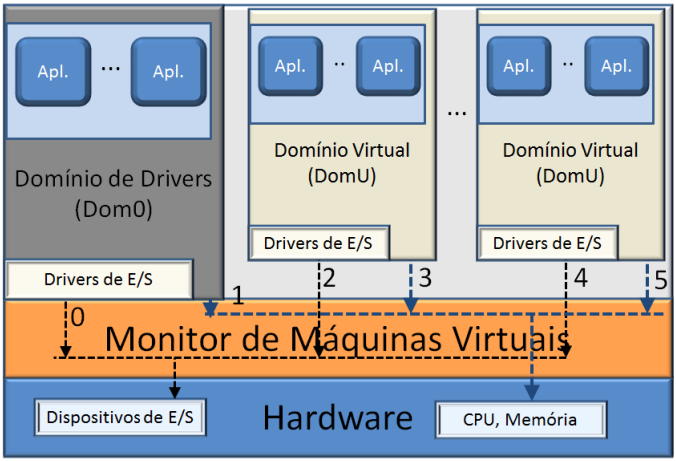
Quando o MMV é inserido a nível de SO, um sistema operacional virtual é abstraído às máquinas virtuais. Neste caso, o *hardware* subjacente é totalmente virtual e emulado, ou seja, as aplicações da MV não terão acesso direto ao *hardware* nativo. No último nível, o MMV é inserido na camada de *software*, onde o recurso abstraído à MV pode ser visto como uma biblioteca, ou um espaço de execução. Um exemplo de MMV inserido à nível de *software* é a máquina virtual Java.

O objetivo deste trabalho está associado ao modelo de serviço de nuvens IaaS. Devido às características deste modelo, onde um cli-

ente adquire infraestrutura, os recursos devem ser abstraídos em nível de *hardware*. O provimento destes recursos pelo MMV pode ser feito de duas formas básicas, através de virtualização completa ou de para-virtualização (BARHAM et al., 2003).

Na virtualização completa os recursos de *hardware* são abstraídos para as máquinas virtuais (ou domínios virtuais). Isto significa que o MMV deve atuar como um mediador entre os sistemas operacionais convidados e a máquina nativa, ou seja, toda comunicação entre o SO convidado e o *hardware* deve ser traduzida pelo MMV. Esta técnica permite a execução em MV de uma plataforma distinta do ambiente nativo. Em alguns casos (como no Xen) é comum o uso de um domínio de *drivers* como uma extensão do *hypervisor* auxiliando o gerenciamento, permitindo ao administrador criar novos domínios virtuais, definir capacidades de recursos destinados aos convidados e monitorar a utilização destes recursos. A Figura 2 apresenta um exemplo do funcionamento deste modo de virtualização.

Figura 2 – Virtualização Completa no Xen




Fonte: Adaptado de (CHERKASOVA; GUPTA; VAHDAT, 2007)

Na Figura 2, o ambiente de virtualização (MMV) é utilizado como um mediador entre o *hardware* e os sistemas operacionais virtuais (domínios virtuais), sendo responsável pela tradução das instruções

entre os SOs e o *hardware*. A figura ilustra a execução de duas máquinas virtuais (DomUs) e um domínio de *drivers* (Dom0) com suas respectivas aplicações. As setas 0, 2 e 4 representam operações de E/S, que são processadas primeiramente nos domínios virtuais antes de serem enviadas ao MMV para, quando traduzidas, serem encaminhadas ao *hardware*. O mesmo ocorre com as operações representadas pelas setas 1, 3 e 5, envolvendo acesso à memória e CPU, onde o gerenciamento ocorre primeiro nos domínios virtuais antes de ser encaminhado ao MMV.

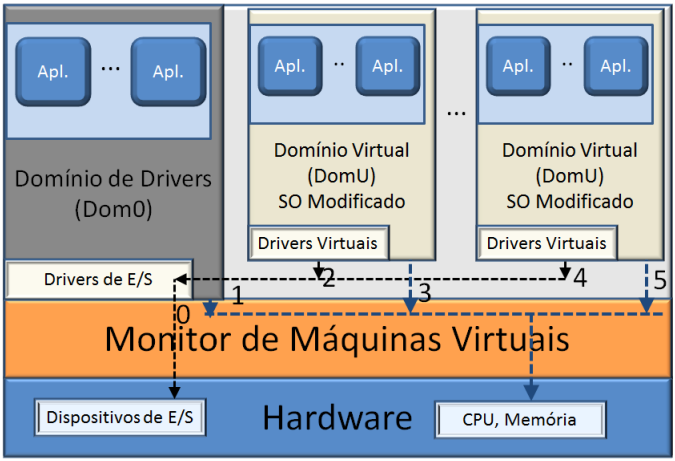
Uma das formas de se implementar a virtualização completa é através da virtualização assistida por *hardware*, neste modelo o MMV executa diretamente sobre uma plataforma de *hardware* com suporte a virtualização, o que privilegia o desempenho no acesso aos recursos. Isto ocorre porque neste modelo não há tradução das instruções pelo MMV, assim, as instruções privilegiadas das MVs são executadas diretamente sobre o *hardware*.

Na paravirtualização o domínio de gerenciamento é utilizado também para compartilhar os *drivers* de E/S entre os domínios virtuais. Para tanto, os sistemas operacionais convidados devem ser modificados. O objetivo da paravirtualização utilizar um domínio virtual para gerenciar os *drivers* dos SOs convidados é o ganho de desempenho. Nesta técnica as instruções associadas a dispositivos de E/S são processadas de forma única no ambiente de virtualização, uma vez que o domínio de *drivers* é uma extensão do MMV. A restrição da paravirtualização é que não se pode virtualizar arquiteturas de *hardware* diferentes da máquina nativa, pois é necessária a compatibilidade entre as plataformas.

A Figura 3 exemplifica o funcionamento da paravirtualização. Observa-se que a organização das camadas é semelhante à virtualização completa. No entanto, é visto que os SOs convidados são modificados em relação aos *drivers* de E/S, utilizando-se de *drivers* virtuais que os direcionam para os *drivers* compartilhados no domínio de *drivers* (Dom0). Assim, quando houver operações de E/S pelas aplicações (setas 2 e 4) elas serão direcionadas para os *drivers* em Dom0. Entretanto, apesar do compartilhamento de E/S, o gerenciamento de processador e memória continua sendo feito pela camada do MMV,

sem processamento em Dom0, idêntico ao que ocorre na virtualização completa.

Figura 3 – Paravirtualização




Fonte: Adaptado de (CHERKASOVA; GUPTA; VAHDAT, 2007)

A proposta de diagnóstico deste trabalho é independente do modelo de virtualização utilizado, uma vez que propõe métricas que podem ser utilizadas internamente na MV, e que são relativas à ocupação dos recursos abstraídos. Para compreender o modelo de diagnóstico proposto é importante o entendimento sobre como é realizada a virtualiazação de rede, processador e memória para que estes recursos virtuais sejam fornecidos às MVs.

Na virtualização de rede, o MMV provê interfaces virtuais para as MVs, estas interfaces virtuais compartilham a largura de banda das interfaces de rede físicas do *hardware* subjacente. Tipicamente, o MMV permite que seja reservada uma fração da largura de banda disponível para cada MV; caso não seja feita uma alocação fixa, ocorre a multiplexação estatística do recurso, com cada MV usando a largura de banda disponível durante um tempo, conforme sua necessidade. Internamente, cada MV possui uma interface de rede virtual, que o seu SO enxerga como uma interface de rede física. O MMV é responsável por transmitir pela interface física os quadros que as MVs enviam por suas

interfaces virtuais, e vice-versa (APPARAO; MAKINENI; NEWELL, 2006).

Na virtualização de processador, o MMV abstrai os detalhes de *hardware* às máquinas virtuais, fornecendo CPUs virtuais (VCPUs) as mesmas. Desse modo, os sistemas operacionais virtuais gerenciam suas VCPUs como se elas fossem CPUs físicas. Em termos gerais, a execução de uma VCPU depende dela ser escalonada sobre uma CPU real, quando um processo do SO virtual estiver em estado de execução o escalonador do MMV irá alocar a VCPU sobre uma CPU física, seguindo uma política de escalonamento própria (WOOD et al., 2009).

A alocação de memória nos ambientes de virtualização é realizada através da reserva de porções de memória para os sistemas operacionais convidados. O MMV distribui a memória disponível por entre as MVs de forma dinâmica ou estática. Em alguns monitores, como VMWare ESX Server e Xen, é possível adequar a quantidade de memória destinada aos domínios virtuais dinamicamente, através de um *Balloon Driver*, adicionando mais capacidade às MVs quando elas necessitam e devolvendo ao MMV a quantidade não utilizada (WALDSPURGER, 2002; BARHAM et al., 2003). Quando o provisionamento é feito de forma estática, não é possível ajustar a capacidade disponível à MV sem que a mesma seja desligada, assim, a parte não utilizada será desperdiçada.

Independentemente da forma como os recursos são providos à máquina virtual, o modelo de diagnóstico pode ser aplicado, uma vez que, as métricas utilizadas são obtidas a nível de sistema operacional da máquina virtual. A Seção 2.3 explica com mais detalhes a técnica de monitoração utilizada neste trabalho para captura das métricas.

## 2.3 MONITORAÇÃO

A monitoração é uma técnica utilizada para observar as atividades de um ambiente computacional. Monitores coletam, através de sensores, métricas relacionadas ao consumo de recursos pelo sistema monitorado. No contexto de diagnóstico de provisionamento de recursos, estas métricas serão posteriormente avaliadas para definir o nível

de adequação dos recursos alocados.

Em uma abstração ideal, a monitoração deve ser a menos intrusiva possível. O monitor deve atuar como um observador, ou seja, não deve provocar interferência no ambiente monitorado. A ocorrência de interferência no desempenho, acrescentando tempo a execução dos sistemas, é denominada *overhead*, e deve ser considerada na avaliação em questão. Basicamente, existem três tipos de monitores: monitor de *hardware*, monitor de *software* e monitores híbridos (MENASCE; ALMEIDA, 2002; ACETO et al., 2013):

- Monitores de *hardware*: São caracterizados por serem inseridos junto ao *hardware* do sistema, e utilizam-se de sensores para capturar as informações relacionadas aos componentes de *hardware*. É possível obter dados relacionados a registradores, posições de memória e canais de E/S. A grande vantagem do uso deste tipo de monitor é que ele não consome os recursos do ambiente monitorado, desta forma é imposto baixo, ou quase nulo, *overhead* sobre os dados coletados. Entretanto, o uso deste tipo de monitor em ambientes virtualizados pode adicionar preocupações de segurança e isolamento, uma vez que o ambiente hospedeiro deverá permitir acesso direto ao *hardware* para a MV. Assim, o domínio virtual poderia, além de conhecer os outros domínios, visualizar as instruções que são executadas, por exemplo.

- Monitores de *software*: Este tipo de monitor é caracterizado como uma aplicação executante junto ao sistema operacional, e reúne informações relacionadas ao consumo de recursos de programas ou do sistema, estas são capturadas através de eventos do sistema. Algumas vantagens podem ser destacadas, tais como a flexibilidade de configuração, a quantidade de dados de desempenho que podem ser registrados e a facilidade de instalação e utilização. A principal desvantagem está relacionada ao *overhead*, se comparado aos monitores de *hardware*, pois, como o monitor de *software* partilha dos mesmos recursos utilizados pelas aplicações do sistema, sua execução poderá influenciar sobre o desempenho das mesmas.

- Monitores híbridos: Monitores híbridos caracterizam-se por combinar características de monitores de *software* e *hardware*. Algumas ferramentas de avaliação de desempenho de rede utilizam-se de monitores híbridos, facilitando a captura simultânea de dados de *software* e *hardware*.

Tendo em vista a restrição do uso de monitores de *hardware* em ambientes virtuais, e consequentemente dos monitores híbridos, os monitores mais apropriados para nuvens IaaS são os monitores de *software* inseridos em nível de SO, pois a coleta de dados é realizada sem que seja necessário permitir acesso direto ao *hardware* pelo ambiente virtualizado.

O uso de monitores em ambientes de computação em nuvens remete a novas classificações, uma vez que existem alguns aspectos não abordados nestas três anteriores, tais como, o tipo de métrica observada, o local onde a monitoração é realizada e o interesse sobre os dados obtidos. Deste modo, classifica-se em dois tipos: *guest-wide*, em nível de máquina convidada, e *system-wide*, em nível de sistema (DU; SEHRAWAT; ZWAENEPOEL, 2011; ACETO et al., 2013).

A monitoração realizada em nível de máquina convidada utiliza métricas interessantes principalmente ao cliente, e são obtidos dados relacionados a uma máquina virtual. O monitor é inserido a nível de sistema operacional da MV, assim, um cliente pode ter a visão sobre o consumo de recursos no intervalo de tempo monitorado.

A monitoração em nível de sistema é interessante principalmente ao provedor da infraestrutura de virtualização, pois pode apresentar métricas e dados relacionados a múltiplas máquinas virtuais. Neste caso, um administrador de ambientes em nuvem pode identificar gargalos de desempenho, máquinas com maiores demandas e níveis de ociosidade das máquinas hospedeiras.

Neste trabalho dá-se foco na monitoração a nível de máquina convidada, pois assim um cliente pode ter um retorno sobre a demanda de recursos de suas aplicações independentemente do provedor. Outro fator que justifica a monitoração a nível de máquina convidada é a garantia de confiança e isolamento da MV por parte do provedor, uma vez que o mesmo não precisa permitir acesso especial ao *hardware* para um domínio virtual específico.

## 2.4 ESTRATÉGIAS DE PROVISIONAMENTO DE RECURSOS

A alocação e distribuição de recursos a máquinas virtuais em ambientes de computação em nuvem é denominado provisionamento. Dependendo do provedor de nuvem, o provisionamento pode ser feito em diferentes níveis de granularidade. A diferenciação está relacionada à definição de como um recurso é tratado. Os recursos podem ser alocados de forma não particionada, como unidades de CPU e interfaces de rede, de forma particionada, como um percentual de CPU ou um percentual de largura de banda, ou até, por instâncias de MV que possuem características fixas de recursos (LAGAR-CAVILLA et al., 2009). Um exemplo de provedor que realiza a alocação por instâncias pré-definidas é a Amazon EC2 (AMAZON, 2014) categorizando-as de acordo com classes de aplicações (ZHANG; CHENG; BOUTABA, 2010).

Neste trabalho, assume-se que a granularidade em que os recursos podem ser oferecidos está relacionada com a capacidade tecnológica do ambiente de virtualização em entregá-los. No Xen, cada tipo de recurso pode ser provisionado de formas diferentes, alguns são compartilhados pelas máquinas virtuais, como é o caso de processador e rede, outros são distribuídos em porções fixas, como é o caso de memória e disco. Contudo, mesmo nos recursos compartilhados é possível definir limites de utilização. Os itens abaixo descrevem a possibilidade de alocação por recursos com base no *hypervisor* Xen, mas características semelhantes de provisionamento podem ser encontradas em outros *hypervisores* tais como o VMware (BARHAM et al., 2003; WALDSPURGER, 2002; LO, 2005):

- Processador: Recurso compartilhado; pode-se provisionar desde o número de processadores virtuais (VCPUS) disponíveis a um ambiente, até percentuais máximos utilizáveis (variável cap no escalonador do Xen). Em alguns *Hypervisors*, estes atributos podem ser modificados em tempo de execução, possibilitando o acréscimo ou decréscimo da capacidade do ambiente virtual.

- Rede: Recurso compartilhado; pode-se definir a largura de banda máxima em Mbps disponível para a MV. Nas versões atuais

(XEN 4.2) a configuração deste parâmetro é feita no momento de criação da MV e não pode ser alterada em tempo de execução. Contudo, de forma alternativa, pode-se utilizar políticas de controle de filas nas interfaces do MMV para definir o consumo máximo de largura de banda em cada domínio virtual.

- Memória: Este recurso é alocado em porções de tamanho delimitado entre as máquinas virtuais, isto se deve principalmente à necessidade de se manter a característica de isolamento entre MVs. A quantidade de memória disponível para um ambiente virtual é definida no momento de criação do mesmo e é possível alocar e desalocar porções de memória para as MVs durante a execução, isto é feito através de mecanismos específicos de cada *hypervisor* (tais como *balloon driver*).

- Disco: Tradicionalmente alocado em porções fixas, basicamente, associa-se um arquivo de disco virtual de tamanho fixo ou variável às MVs. Mais recentemente, estudos apresentam a forma de particionar a largura de banda de acesso ao recurso de disco para os ambientes virtuais (WANG et al., 2009; RAO et al., 2011), uma ferramenta que possibilita este particionamento em MMVs Linux é *dm-ioband* (DM-IOBAND, 2014). Contudo, esta funcionalidade não é disponibilizada nativamente nos ambientes de virtualização, e consequentemente não é encontrada nos ambientes de computação em nuvens (SULEIMAN et al., 2012).

Uma das características da computação em nuvens definidas na Seção 2.1 é que os recursos são entregues sob demanda, e isto tem sido um dos fatores que influenciam no crescimento da utilização das nuvens, uma vez que é possível utilizar modelos do tipo *pay-as-you-go* (ARMBRUST et al., 2010).

O uso de nuvem sob demanda agrega benefícios tanto ao provedor de nuvem quanto ao cliente. O provedor pode prover capacidades de acordo com a necessidade da aplicação e disponibilidade de recursos observando um SLA (VAN; TRAN; MENAUD, 2009). Do ponto de vista do cliente, a alocação sob demanda possibilita ajustar a capacidade reservada à demanda das aplicações, maximizando a utilização dos recursos e o desempenho da solução. O provisionamento,

na forma estudada, pode ser feito de maneira estática ou dinâmica:

- **Provisionamento Estático** - Definido pelo usuário ou sistema, o cliente determina quais são as necessidades de recursos, alocando-os a uma ou múltiplas instâncias de MV. A realocação dos recursos não é permitida em tempo de execução da MV. Por exemplo, um cliente pode solicitar uma instância com 2 GB de memória, e após a monitoração identificar que necessita de 1 GB, assim, pode solicitar ao provedor que adeque às suas necessidades. Para tanto, a máquina deve ser desligada, reconfigurada e reiniciada.

- **Provisionamento Dinâmico** - Observa-se a aplicação e, em tempo de execução os recursos previamente alocados são ajustados, acrescentando-se ou reduzindo-se os mesmos de acordo com a necessidade. Técnicas de aprendizado de máquina e modelos preditivos de desempenho são frequentemente utilizados no provisionamento dinâmico (RAO et al., 2011; KUNDU et al., 2012). Este provisionamento pode ser feito de duas formas, diretamente pelo provedor, ou por ferramentas de *software* fornecidas aos usuários.

O sucesso do modelo de nuvem, em parte, se dá pela flexibilidade no provisionamento. Portanto, o provisionamento adequado às necessidades, indiretamente, torna o modelo em nuvem interessante a provedor e a clientes. Diagnosticar, em tempo de execução via monitoração, as necessidades é o primeiro passo para a adequação correta de recursos às necessidades das aplicações. A partir de um diagnóstico confiável é possível desenvolver ferramentas de provisionamento dinâmicas eficientes.

## 2.5 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foram apresentados os conceitos principais para fundamentação teórica deste trabalho. Foi visto que a nuvem computacional é um modelo de computação em que o conjunto de recursos é ajustável às necessidades dos clientes, ou seja, os recursos computacionais são oferecidos como serviços. Sobre o oferecimento de recursos

como serviço foram identificadas três categorias: IaaS, onde os recursos são itens de infraestrutura; PaaS, onde é oferecida uma plataforma de desenvolvimento; e, SaaS, onde os recursos estão em nível de aplicação.

Outro conteúdo abordado neste capítulo foi a virtualização, o estudo desta tecnologia foi justificado pelo papel importante que ela teve no crescimento das nuvens computacionais. A virtualização permite que múltiplos sistemas operacionais compartilhem um mesmo *hardware*, permitindo que provedores otimizem a utilização de seus recursos físicos.

O modelo de diagnóstico a ser apresentado neste trabalho baseia-se na observação de métricas de utilização dos recursos, e para fundamentar as formas como estas métricas podem ser obtidas, neste capítulo também foram apresentadas algumas classificações a respeito de monitoração de recursos. Basicamente, definiu-se os tipos de monitores (*hardware*, *software* e híbrido) e o tipo de monitoração no contexto de nuvem (*guest-wide* e *system-wide*).

Na última seção deste capítulo abordou-se as estratégias de provisionamento de recursos, neste caso foi definido como os recursos podem alocados para as máquinas virtuais, viu-se que os recursos podem ser entregues de forma compartilhada (rede e cpu) e não compartilhada (memória e disco). Na forma compartilhada os recursos são multiplexados entre as máquinas virtuais que o utilizam, na outra forma, tem-se espaços alocados e destinados a cada instância.

Sobre as características de nuvem computacional, foi delimitado que o modelo proposto neste trabalho deve se aplicar a nuvens IaaS, pois, é neste caso que o cliente tem acesso a informações de utilização de recursos em nível de sistema operacional. A respeito de monitoração definiu-se que, para os propósitos do modelo, ela deverá ser realizada com métricas de sistema operacional e será *guest-wide*. A respeito dos recursos, decidiu-se criar um modelo para diagnosticar processador, rede e memória, pois, considera-se que estes recursos são os que definem o tamanho das instâncias dos provedores atuais. Disco foi descartado porque, apesar de existirem estudos sobre provisionamento de largura de banda, na maioria dos casos eles ainda são oferecidos pelos provedores como espaço ocupado. O próximo capí-

tulo apresenta os modelos de diagnóstico elaborados para diagnosticar os recursos provisionados.

# 3 MODELO DE DIAGNÓSTICO

Este trabalho propõe um modelo para diagnosticar o provisionamento de processador, rede e memória de uma máquina virtual. Utiliza-se a monitoração com métricas representativas de comportamento para classificar o provisionamento dos recursos nos níveis subprovisionado, superprovisionado ou adequado. Isto permite que clientes possam ajustar acordos de níveis de serviço com os provedores, possibilitando que os recursos alocados sejam suficientes para que se tenha o menor custo com um nível de desempenho aceitável (equivalente a um custo maior).

Para diagnosticar o provisionamento dos recursos, é necessário definir métricas que indiquem se esse provisionamento está adequado, sub- ou superprovisionado. Um diagnóstico de superprovisionamento significa que adicionar capacidade ao recurso não deve agregar melhorias relevantes de desempenho às aplicações. De modo oposto, o subprovisionamento significa que a cada acréscimo de capacidade o desempenho será aprimorado significativamente. Um diagnóstico de recurso adequado representa um estado onde reduzir a capacidade deve afetar o desempenho enquanto aumentar a capacidade não promove influência sobre o desempenho.

O objetivo deste capítulo é apresentar os modelos para diagnosticar os recursos provisionados, a validação dos modelos propostos é realizada posteriormente no Capítulo 4. Inicialmente as Seções 3.1, 3.2 e 3.3 apresentam os modelos para diagnosticar processador, rede e memória, respectivamente. Em sequência, na Seção 3.4 é proposto um algoritmo para ajustar as capacidades dos recursos de acordo com os resultados dos modelos. Finalmente, na Seção 3.5 é feita uma análise dos trabalhos relacionados.

## 3.1 MODELO PARA PROCESSADOR

Para diagnosticar a adequação do provisionamento de processador, são avaliadas duas métricas: a utilização de processador ($U_{cpu}$) e

o percentual de "roubo" (*%Steal*). A utilização $U_{cpu}$ representa quanto tempo o processador estava executando tarefas em um espaço de tempo. Nos sistemas operacionais modernos a contabilização do processador pode subdividir a utilização em vários componentes; por exemplo, no Linux, *%User* representa o percentual de tempo em que o recurso foi utilizado por processos de usuário, e *%System* o tempo usado pelo sistema operacional (HOCH, 2010). Deste modo, para simplificar, é possível usar o percentual de ociosidade (*%Idle*) para obter de forma indireta a utilização do processador. Sendo assim, a utilização de processador é dada por $U_{cpu} = 100\% - \%Idle$, sendo sua média em um período denotada por $\overline{U_{cpu}}$.

O tempo de roubo (*steal time*) é uma métrica conhecida em ambientes de *mainframe*, e que representa o percentual de tempo em que uma máquina virtual deseja executar (ou seja, possui processos em estado de pronto) mas sua VCPU não está alocada em uma CPU física, devido à contenção de processador entre MVs (RIEL, 2006). O tempo de roubo pode ser reportado pelo SO de forma absoluta (em segundos, por exemplo) ou relativa, como a fração de tempo de roubo em um dado intervalo (*%Steal*). Essa métrica pode auxiliar a identificar a ausência de recursos, de forma complementar à utilização de processador: quando *%Steal* é maior do que zero, a máquina virtual teve necessidades de execução em processador que não foram supridas. Isto pode ocorrer quando máquinas virtuais concorrentes disputam um recurso escasso, ou quando o recurso é tomado da máquina virtual pelo MMV. Por exemplo, o Xen permite que sejam definidas cotas de processador para cada MV (usando a variável cap, introduzida na Seção 2.4); quando uma MV usa toda a sua cota de CPU, o recurso é retirado à força pelo MMV, o que pode levar à ocorrência de tempo de roubo caso a MV tenha processos prontos para execução.

A taxa de ocupação de processador tende a variar durante a execução das aplicações no período de tempo. Logo, deve-se definir limiares para caracterizar os três níveis relacionados ao provisionamento do recurso. Entretanto, não há um consenso relacionado aos valores admitidos para os limiares; por exemplo, (HOCH, 2010) aponta que valores entre 0–5% para *%Idle* indicam saturação, enquanto (PADALA et al., 2007) adota 80% para utilização (20% de *%Idle*) como limite

de saturação, com a justificativa de manter uma margem de segurança para acomodar o comportamento variável das aplicações. Considerando que a ocupação de CPU pode alternar entre picos de utilização, define-se o recurso como subprovisionado em duas situações: quando $\overline{U_{cpu}}$ mantiver-se maior ou igual a 80%, ou quando houver capacidade de CPU retirada da MV, ou seja, tempo contabilizado em *%Steal*.

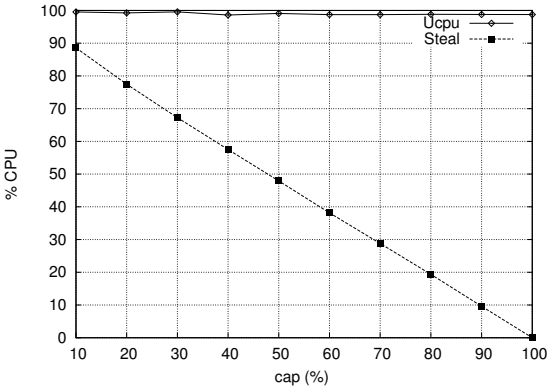
Para melhor entender o comportamento da métrica *%Steal*, efetuou-se um pequeno experimento que considera uma MV com execução de um processo *CPU-Bound* que calcula por tempo indeterminado a raiz quadrada de números aleatórios. Foi destinada uma VCPU à MV e a capacidade de CPU utilizável pela MV foi ajustada através da métrica cap em um intervalo de 10 e 100%. Tipicamente, o melhor desempenho associado a este processo seria com a capacidade de uma CPU inteira (cap = 100%), ter mais do que isto não influenciaria no seu desempenho uma vez que a aplicação é *monothread*.

O resultado da monitoração de *%Steal* para cada capacidade é apresentado na Figura 4. Ao observar estes resultados é possível perceber que a ocorrência de *%Steal* reduz de acordo com o aumento da métrica cap, enquanto a taxa de utilização de CPU mantém-se constante. Além disto, observa-se que o valor de *%Steal* é igual a diferença entre a utilização contabilizada e o percentual de cap. Este comportamento está de acordo com o esperado, uma vez que a aplicação é *CPU-Bound* e *monothread*, o que caracteriza, para capacidades inferiores a 100%, um subprovisionamento do recurso. Portanto, pode-se dizer que a métrica *%Steal* pode ser utilizada de forma complementar à utilização de CPU.

A partir destes resultados questiona-se: o que aconteceria se fosse provisionada à MV mais do que uma VCPU? Seria possível utilizar somente a utilização total como métrica de diagnóstico? Para responder estas questões o experimento anterior foi refeito, onde, destinou-se à MV duas VCPUs e a capacidade foi variada entre 10 e 200% através da métrica cap. Estes resultados são expostos pela Figura 5.

O comportamento do experimento apresentado pela Figura 5 é o esperado: a medida que o cap aumenta, o *%Steal* reduz. O *%Steal* sugere exigência da aplicação por CPU, uma vez que, aumentando-se cap, sacia-se gradativamente a necessidade da aplicação, e quando cap

Figura 4 – Monitoração da utilização de CPU e *%Steal* de acordo com a variação da capacidade de CPU destinada a uma MV entre 10 e 100% com execução de um processo *CPU-Bound*.



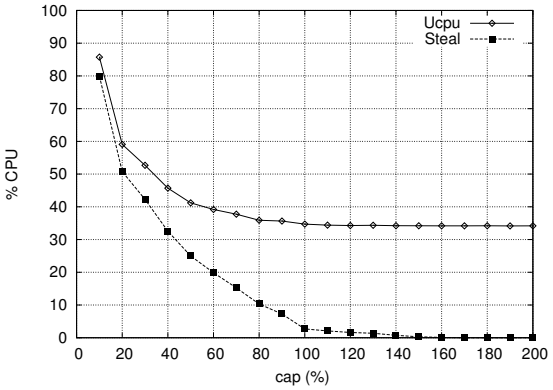

Fonte: o autor

é superior a 100% (ou uma VCPU) há valores mínimos de *%Steal*. Isto reforça a ideia do uso de *%Steal* como métrica complementar a utilização para diagnosticar o subprovisionamento.

Entretanto, atendo-se às características da aplicação *CPU-Bound* e *monothread*, observa-se que a utilização de CPU não atinge 100% da capacidade total quando cap$<$100%, nem 50% quando cap$\geq$100%, o que é diferente do comportamento observado na Figura 4, que satura o recurso. Desta forma, conclui-se que o comportamento de $\overline{U_{cpu}}$ inferior ao esperado é causado por influência dos controles do MMV no escalonamento e distribuição dos créditos. Tal discrepância já fora observada em (CHERKASOVA; GUPTA; VAHDAT, 2007), evidenciando que o *credit-scheduler* (algoritmo usado pelo Xen para escalonamento de VCPUs) apresenta erros na entrega do recurso com a utilização de cap. Ressalta-se que este comportamento não invalida o diagnóstico de subprovisionamento baseado em *%Steal*, pois a ocorrência dele caracteriza a ausência de recurso.

Com base na discussão acima, é possível estabelecer limiares para $\overline{U_{cpu}}$ e *%Steal* que indiquem a saturação do processador, sendo necessário definir valores que caracterizem o superprovisionamento, algo inexplorado na literatura. Considerando que o valor apropriado

Figura 5 – Monitoração da utilização de CPU e *%Steal* de acordo com a variação da capacidade de CPU destinada a uma MV entre 10 e 200% com execução de um processo *CPU-Bound*.




Fonte: o autor

para utilização de processador para uma MV é o equilíbrio entre o máximo e mínimo disponível, espera-se que o diagnóstico de provisionamento superdimensionado ou adequado considere a proximidade ao equilíbrio (50%), permitindo que a capacidade provisionada suporte oscilações positivas e negativas de carga a um custo aceitável. Sendo assim, cenários onde a média de utilização de processador estiver entre 50% e 80% serão diagnosticados como adequados, pois não há carga suficiente para saturar o recurso e a redução da capacidade pode afetar o desempenho. Nos cenários com utilização inferior a 50%, a média de utilização pode não expressar corretamente um superprovisionamento do recurso, pois a carga pode ter $\overline{U_{cpu}}$ baixa mas apresentar picos de utilização que impeçam uma redução no recurso alocado (caso de C3 na Fig. 6(a)).

Para lidar com esse caso, avalia-se a variabilidade da utilização nas situações em que $\overline{U_{cpu}}$ é menor que 50%, verificando qual a fração de pontos que atingem o pico de utilização do recurso no intervalo monitorado. Seja $U^i_{cpu}$ um ponto de medição de $U_{cpu}$, $\theta$ o limiar de saturação de CPU (i.e., $U^i_{cpu} \geq \theta$ indica CPU saturada), $N$ o número de pontos no intervalo e $N'$ o número de pontos com CPU saturada ($U^i_{cpu} \geq \theta$). Quando $N'/N > \omega$, onde $\omega$ é o limiar para o percentual

de pontos em saturação, a MV utiliza toda a capacidade de CPU com frequência, e o provisionamento é adequado; caso contrário, ele é superdimensionado. O cálculo da frequência de utilização próxima ao topo é feito através das Equações (3.1)–(3.3).

O diagnóstico de provisionamento de processador conforme as métricas propostas é expresso pela Tabela 1, que define subprovisionamento, principalmente, pela ocorrência de *%Steal*, neste caso, admite-se um limiar (1%) para contemplar as oscilações de comportamento. Quando a média de utilização contabilizada em *%Steal* for maior que o limiar há subprovisionamento, nos demais casos avalia-se a situação da média de utilização em relação aos demais limiares. Com base em observações experimentais, utilizou-se $\theta = 95\%$ e $\omega = 20\%$.

$$u_i = \begin{cases} 1, & \text{se } U^i_{cpu} \geq \theta \\ 0, & \text{se } U^i_{cpu} < \theta \end{cases} \tag{3.1}$$

$$N' = \sum_{i=0}^{N} u_i \tag{3.2}$$

$$\varphi = \frac{N'}{N} \tag{3.3}$$

Tabela 1 – Diagnóstico de provisionamento de CPU

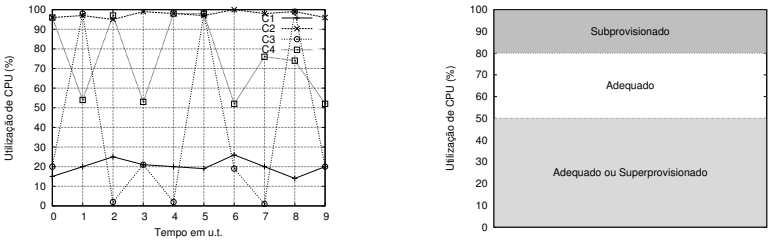
| | $\%Steal \geq 1\%$ | $\%Steal < 1\%$ |
|---|---|---|
| $\overline{U_{cpu}} \geq 80\%$ | Subprovisionado | Subprovisionado |
| $50\% \leq \overline{U_{cpu}} < 80\%$ | Subprovisionado | Adequado |
| $\overline{U_{cpu}} < 50\% \wedge \varphi > \omega$ | Subprovisionado | Adequado |
| $\overline{U_{cpu}} < 50\% \wedge \varphi \leq \omega$ | Subprovisionado | Superprovisionado |

Fonte: o autor

Para exemplificar, seja um cenário hipotético com 10 medições de $U^i_{cpu}$ associadas a quatro cargas de trabalho distintas (C1–C4) executando sobre uma MV com uma VCPU sem limitação por cap, portanto sem a ocorrência de *%Steal*, mostrado na Fig. 6(a). Observa-se que, para C1, a média de $U_{cpu}$ no período é de 20% (processador

superprovisionado). Para C2, tem-se $\overline{U_{cpu}}$ próxima a 100% (processador subprovisionado). A carga de trabalho C3 apresenta média de $U_{cpu}$ igual a 38%, com quatro pontos onde a utilização é próximo ao topo ($\varphi = 40\%$); assim, o provisionamento é adequado, pois o recurso alocado suporta a variação entre picos de utilização e ociosidade. A carga C4 apresenta média de $U_{cpu}$ igual a 75%, e assim o provisionamento é considerado adequado.

Figura 6 – Utilização de CPU em um cenário hipotético (a) e Limiares para diagnóstico de provisionamento (b). C1 está superprovisionado, C2 está subprovisionado, C3 e C4 têm provisionamento adequado.




(a) Cenário hipotético (b) Limiares

Fonte: o autor

A Fig. 6(b) resume os possíveis diagnósticos do provisionamento de processador com base em medições de $U_{cpu}$ (sem *%Steal*). No caso de utilização média situada na região superior ($\overline{U_{cpu}} \geq 80\%$), tem-se subprovisionamento. Para utilização na região intermediária ($50\% \leq \overline{U_{cpu}} < 80\%$), o provisionamento é adequado. Caso a utilização média corresponda à região inferior ($\overline{U_{cpu}} < 50\%$), o recurso pode ser adequado (quando $\varphi > \omega$) ou superprovisionado (quando $\varphi \leq \omega$).

## 3.2 MODELO PARA REDE

A métrica básica para verificar a adequação da capacidade de rede alocada a uma máquina virtual é a largura de banda. Considera-se apenas a largura de banda de transmissão, pois é a única que pode ser analisada com mais profundidade apenas com dados internos à MV;

para a largura de banda de recepção, a análise possível é se a banda consumida se aproxima da banda alocada, não sendo possível inferir se há demanda além da alocação. Do mesmo modo que a utilização de processador, a largura de banda consumida pode variar bastante ao longo do tempo, o que faz com que a média das observações seja insuficiente para caracterizar esse consumo. (PRAS et al., 2009) sugere que se utilize a variância entre os pontos medidos. Neste trabalho, a variabilidade do consumo de banda é mensurada pelo coeficiente de variação (CV), que é a razão entre o desvio padrão e a média dos pontos medidos: se o CV for de até 5%, considera-se comportamento de consumo constante. Em relação à variância, essa métrica tem a vantagem de ser relativa, e não absoluta. O limiar de 5% foi estabelecido com base em observações experimentais.

Todavia, apenas a análise da largura de banda não permite obter um diagnóstico preciso da adequação do provisionamento da rede. Em particular, quando a utilização de largura de banda se aproxima do máximo disponível, é necessário avaliar se há mesmo necessidade de banda adicional (ou seja, se a capacidade disponível está sendo um fator limitante para a MV) ou se a MV está apenas explorando ao máximo a banda disponível, sem precisar de banda extra. Protocolos de rede que implementam controle de congestionamento, como o TCP, adaptam suas taxas de transmissão às condições da rede, tentando encontrar a largura de banda disponível na rede, que é limitada pela menor largura de banda ao longo do caminho de transmissão (JACOBSON, 1988). Para fins de diagnóstico da adequação dos recursos, deve-se identificar se essa limitação está na largura de banda local (seja física ou virtual), ou se ela está ocorrendo em algum ponto no interior da rede. No primeiro caso (que é o único em que se pode tomar alguma providência local), o MMV não consegue transmitir os quadros das máquinas virtuais com a rapidez necessária, e ocorre formação de fila na interface de rede da MV. No segundo caso, não há formação de fila na interface local. Portanto, o comprimento da fila da interface de rede da MV pode ser usado para verificar a adequação da alocação de recursos quando a largura de banda consumida se aproxima do limite.

Com base nas duas métricas propostas, coeficiente de variação do consumo de banda e fila na interface de rede, definem-se

as três categorias de provisionamento (subprovisionado, adequado e superprovisionado). Quando existe fila na interface de rede da MV, diagnostica-se o recurso como subprovisionado. Quando a banda consumida é constante e não há enfileiramento, o recurso está provisionado de forma adequada. O superprovisionamento ocorre quando o consumo de banda é variável, sem enfileiramento na interface de rede da MV. O diagnóstico de acordo com estas categorias é obtido pelas relações apresentadas pela Tabela 2, onde $CV_{lb}$ é o coeficiente de variação da largura de banda consumida e $\overline{fila}$ é a quantidade média de pacotes na fila da interface.

Tabela 2 – Diagnóstico de provisionamento de Rede

| | $\overline{fila} > 0$ | $\overline{fila} = 0$ |
|---|---|---|
| $CV_{lb} > 5\%$ | Subprovisionado | Superprovisionado |
| $CV_{lb} \leq 5\%$ | Subprovisionado | Adequado |

Fonte: o autor

## 3.3 MODELO PARA MEMÓRIA

Diferentemente de processador e rede, a memória não pode ser compartilhada pelo conjunto de MVs em execução sobre o hospedeiro, uma vez que este recurso trabalha com alocação de espaços para armazenamento. Deste modo, o espaço total de memória disponível na máquina física deve ser dividido entre máquinas convidadas e o monitor de máquinas virtuais.

Normalmente, a avaliação sobre o desempenho de memória é dado pela utilização média de cada domínio. Um indicativo de pressão sob este recurso é o percentual de utilização de área de *swap*. O sistema operacional utiliza a área de *swap* quando não há mais memória física disponível para os processos em execução (TANENBAUM, 2007). Todavia, esta métrica é reativa (WOOD et al., 2007), pois o desempenho da MV já foi afetado quando a utilização de *swap* é percebida. Deste modo, buscou-se avaliar as métricas de memória que poderiam ser utilizadas para diagnosticar comportamentos das MV em relação a este recurso de modo preventivo, ou seja, antes do desempe-

nho ser degradado.

A abordagem de avaliação preventiva em relação à memória justifica-se pelo fato de que a utilização deste recurso em servidores na nuvem apresenta uma taxa de oscilação menor do que em outros recursos como processador e rede (BIRKE; CHEN; SMIRNI, 2012), ou seja, o crescimento da utilização tende a ser mais demorado, permitindo abordagens preventivas.

O *kernel* do sistema operacional Linux contém diferentes informações relativas ao estado da memória em `/proc/meminfo`. Para os propósitos deste trabalho, as mais relevantes são *MemTotal* (quantidade de memória física disponível), *MemFree* (quantidade de memória não utilizada), *Buffers* (memória utilizada para *buffers* de E/S), *Cached* (memória utilizada para *cache* de páginas), e *Committed_AS* (memória reservada para o conjunto de processos em execução). Uma vez que nem toda memória requisitada por um processo deve estar efetivamente em uso, o Linux realiza *overcommit* de memória, garantindo que todas as requisições de memória serão satisfeitas, mesmo que um processo solicite mais memória (usando as chamadas *sbrk* ou *mmap*, por exemplo) do que a disponível no conjunto RAM+swap (GORMAN, 2004). Assim, *Committed_AS* pode ser vista como um pior caso da quantidade de memória necessária para acomodar o conjunto de processos inteiramente na memória RAM.

Estas métricas sugerem algumas abordagens simplistas para diagnosticar o provisionamento de memória. Dentre as possibilidades, a observação de baixo *MemFree* (entre 10 e 20% de *MemTotal*) poderia indicar memória subprovisionada, enquanto grande quantidade de *MemFree* (mais que 50% de *MemTotal*) poderia apontar que a memória está superprovisionada. Outra abordagem direta seria comparar *Committed_AS* com *MemTotal*, definindo limiares para cada nível de provisionamento (por exemplo, a memória estaria superprovisionada quando *Committed_AS* $< 50\%$ de *MemTotal*, subprovisionada quando *Committed_AS* $> 90\%$ de *MemTotal*, e provisionada adequadamente nos demais casos).

Entretanto, estas abordagens simplistas invariavelmente erram no diagnóstico, uma vez que elas não observam detalhes importantes do gerenciamento de memória. Por exemplo, aplicar a primeira

abordagem sobre os sistemas de intensa atividade de E/S pode gerar diagnóstico incorreto, pois estes sistemas apresentam baixo *MemFree*, porque a memória não utilizada pelos processos será destinada aos *buffers* de E/S e para *cache* de páginas. Na segunda abordagem, basear-se somente em *Committed_AS* pode ser muito conservador para diagnosticar o provisionamento de memória, simplesmente porque esta é uma estimativa de pior caso de utilização de memória (e ser conservador em ambientes de nuvem implica em fazer o usuário pagar por recursos que ele não irá utilizar efetivamente).

Esta discussão sugere que basear-se em uma única métrica para diagnosticar o provisionamento de memória não é eficaz. Considerando as limitações apresentadas sobre a métrica de memória disponível (*MemFree*), optou-se por utilizar a memória residente (*MemRes*) como métrica de ocupação de memória, pois ela indica a quantidade de memória utilizada pelos processos e o SO, e pode ser obtida através da relação $MemRes = MemTotal - MemFree - Buffers - Cache$. Para o propósito de diagnóstico, a memória residente observada pode ser categorizada nos níveis *alta*, *confortável* e *baixa*. Estas categorias são informalmente definidas como se segue (uma definição formal é apresentada mais à frente nesta seção):

- Memória residente alta: quando *MemRes* está muito próxima de *MemTotal*. Nesta situação, o sistema operacional convidado começa a mover páginas para o espaço de *swap* a fim de disponibilizar memória física, e assim, o desempenho pode ser afetado sensivelmente.

- Memória residente baixa: ocorre quando observa-se o valor de *MemRes* muito abaixo de *MemTotal*. Nesta situação, a memória disponível pode ser reduzida sem impactar o desempenho das aplicações na MV.

- Memória residente confortável: Quando a memória residente não está nem baixa nem alta, caracterizando uma situação ideal: a máquina virtual tem memória suficiente para garantir o nível de desempenho esperado e boa parte dessa memória está em uso.

Considerando que a memória reservada (obtida em sistemas

Linux através de *Committed_AS*) indica a quantidade de memória alocada para o conjunto de processos no sistema, ela pode ser utilizada como um indicativo de que a memória residente irá crescer a ponto de afetar o desempenho. Assim, a quantidade de memória reservada é classificada como *relevante* quando *Committed_AS* é alta a ponto de sugerir risco considerável, ou *irrelevante* quando no risco é desprezível.

Tendo em vista essas categorizações, a intuição do modelo é a seguinte. Quando a memória residente está alta, a memória está subprovisionada. Se a memória residente for baixa ou confortável, é necessário observar a memória reservada: se a memória residente for confortável e a memória reservada relevante, a memória está subprovisionada. A memória é considerada adequada quando a memória residente for confortável e a memória reservada for irrelevante, ou quando a memória residente for baixa e a memória reservada for relevante. O diagnóstico de superprovisionamento é observado quando a memória residente for baixa e a memória reservada for irrelevante.

Formalmente, o modelo proposto para diagnosticar o provisionamento de memória utiliza duas métricas, média de memória residente ($\overline{m_r}$) e média de memória reservada ($\overline{m_c}$), e três limiares, limiar de memória residente alta ($\alpha$), limiar de memória residente baixa ($\beta$), e limiar de memória reservada relevante ($\gamma$). $m_r$ e $m_c$ são definidas respectivamente pelas Equações (3.4) e (3.5). Os valores são capturados periodicamente, sendo $\overline{m_r}$ e $\overline{m_c}$ a média dos valores obtidos no intervalo de diagnóstico. Assim, a memória residente é alta quando $\overline{m_r} > \alpha$, confortável quando $\beta \leq \overline{m_r} < \alpha$, e baixa quando $\overline{m_r} < \beta$. E ainda, tem-se memória reservada relevante quando $\overline{m_c} \geq \gamma$ e irrelevante quando $\overline{m_c} < \gamma$. Um resumo do modelo de diagnóstico é apresentado na Tabela 3. Os valores utilizados para os limiares foram obtidos empiricamente, sendo $\alpha = 70\%$, $\beta = 50\%$, e $\gamma = 150\%$.

$$m_r = \frac{MemRes}{MemTotal} \tag{3.4}$$

$$m_c = \frac{Committed\_As}{MemTotal} \tag{3.5}$$

Apesar da discussão sobre a medição de memória residente e

Tabela 3 – Diagnóstico do provisionamento de memória

| | $\overline{m_c} \geq \gamma$ (relevante) | $\overline{m_c} < \gamma$ (irrelevante) |
|---|---|---|
| $\overline{m_r} \geq \alpha$ (alta) | subprovisionado | subprovisionado |
| $\beta \leq \overline{m_r} < \alpha$ (confortável) | subprovisionado | adequado |
| $\overline{m_r} < \beta$ (baixa) | adequado | superprovisionado |

Fonte: o autor

reservada ser feita sobre o ambiente Linux, entende-se que esta abordagem é mais genérica, e o mesmo modelo pode ser utilizado em outros sistemas operacionais convidados que permitam a obtenção destas mesmas métricas a partir de outras variáveis, como por exemplo em sistemas operacionais Windows e variantes de Unix. Entretanto, como nem todos os SOs realizam *overcommit* de memória, os limiares $\alpha$, $\beta$ e $\gamma$ podem ter de ser ajustados. Até mesmo em ambiente Linux os usuários podem utilizar outros valores para limiares a fim de ajustar a sua relação custo/benefício. Uma discussão sobre os possíveis ajustes é realizada na Seção 4.5, após a apresentação dos resultados experimentais.

## 3.4 AJUSTE DE PROVISIONAMENTO DE RECURSOS

O modelo de diagnóstico proposto pode ser aplicado a cada recurso, identificando se o mesmo atende às necessidades das cargas de trabalho em execução. Contudo, nas nuvens IaaS, um cliente pode aplicar o modelo a múltiplos recursos simultaneamente, e fazer uso dos resultados para ajustar a capacidade provisionada. Nestes casos, é possível que ocorram diagnósticos idênticos para múltiplos recursos em um período de avaliação (Por exemplo: Processador - SUB, Rede - SUB e Memória - SUB; ou: Processador - SUB, Rede - SUPER e Memória - SUPER), e assim, podem surgir dúvidas sobre qual recurso ou diagnóstico ajustar primeiro.

As abordagens de diagnóstico propostas privilegiam o desempenho em detrimento ao custo, e assim, entende-se que primeiramente deve-se eliminar as situações de subprovisionamento. Para identificar a ordem de ajuste nos casos em que há igualdade de diagnóstico para mais de um recurso, aplicou-se o modelo de diagnóstico sobre os dados do experimento com RUBiS (Seção 4.4) com cinco níveis de granularidade para cada recurso, avaliando qual a ordem de ajuste que apresenta o menor número de iterações para encontrar a combinação de alocação ótima. As ordens de ajuste e o número de iterações necessárias são apresentadas pela Tabela 4.

Tabela 4 – Iterações para ajuste de provisionamento, onde P é processador, R é rede e M é memória.

| Ordem | Iterações |
|---|---|
| P - R - M | **8** |
| P - M - R | **8** |
| R - P - M | **8** |
| R - M - P | 18 |
| M - P - R | 12 |
| M - R - P | 15 |

Fonte: o autor

Estes resultados mostram que, para a carga de trabalho da ferramenta RUBiS, deve-se ajustar primeiramente processador e rede e por último a memória. Assim, elaborou-se um algoritmo para ajuste de provisionamento exposto pela Figura 7. A intuição que rege o modelo é de primeiro ajustar os pontos de subprovisionamento, seguindo a ordem R-P-M (linhas 4 a 11), em seguida, ajustar os pontos de superprovisionamento até que não se tenha mais combinações selecionáveis (linhas 12 a 21). Em ambos os casos, para que o ajuste seja realizado é necessário que a nova capacidade esteja disponível, por exemplo, em uma situação de incremento é necessário que haja mais recurso, este teste é feito nas linhas 6 e 14 do pseudocódigo. Para o ajuste de redução de capacidade a nova combinação de alocação deve ser verificada, caso ela já tenha sido avaliada o recurso não deverá ser ajustado (linha 15).

Para o propósito de verificação do diagnóstico, apresentado na

Figura 7 – Algoritmo de ajuste e provisionamento de recursos

```
 1: while true do
 2:     executa_diagnóstico(rede,cpu,memória)                ▷ obter dados
 3:     ajuste ← 0                                           ▷ variável de controle
 4:     for recurso in [rede, cpu, memória] do
 5:         if ajuste == 0 then
 6:             if recurso == sub E pode_aumentar(recurso) then
 7:                 aumenta(recurso)
 8:                 ajuste ← 1                               ▷ ajuste realizado
 9:             end if
10:         end if
11:     end for
12:     for recurso in [memória, cpu, rede] do
13:         if ajuste == 0 then
14:             if recurso == super E pode_diminuir(recurso) then
15:                 if permite_nova_combinação() then
16:                     diminui(recurso)
17:                     ajuste ← 1
18:                 end if
19:             end if
20:         end if
21:     end for
22: end while
```

Fonte: o autor

Seção 4.4, utilizou-se a abordagem de parar o ajuste quando não há mais alocações selecionáveis. Todavia, quando aplicado em um ambiente de nuvem os ajustes devem seguir por tempo indefinido, uma vez que podem ocorrer mudanças de comportamento entre um diagnóstico e outro.

## 3.5 TRABALHOS RELACIONADOS

Os trabalhos relacionados associados a este trabalho podem ser divididos em duas categorias: identificação da demanda de recursos de máquinas virtuais e provisionamento dinâmico de recursos em ambientes de nuvem.

**Identificação da demanda.** Métricas para identificar demandas de recursos de máquinas virtuais executando Linux são propostas em (RIEL, 2006). Para o processador é usada uma análise similar à apresentada

na Seção 3.1, envolvendo *%Idle* e *%Steal*; no entanto, não são definidos limiares para esses valores, apenas noções subjetivas como "*%Idle* alto" e "*%Steal* baixo". Para as demandas de memória, van Riel introduz o conceito de distância de refalta, que pode ser utilizado para identificar o número de faltas de páginas que poderiam ser evitadas ao aumentar a memória disponível de uma MV, e sugere a análise sobre o quanto da memória interna da MV foi referenciada (se existem muitas páginas não referenciadas e a MV não está aguardando por páginas, ela tem mais páginas do que precisa). Entretanto, estatísticas de refalta e contadores de referência a páginas não estão disponíveis para aplicações de usuário nas versões atuais do Linux, nem em outros sistemas operacionais. O artigo menciona o uso de largura de banda/vazão como métrica para avaliar as demandas de disco e de rede, mas sem uma proposta concreta, além disso, ele não apresenta resultados experimentais.

Em (MUKHERJEE et al., 2013) faz-se o uso de *probes* inseridos nas máquinas virtuais para identificar a contenção em recursos compartilhados ocasionada pela interferência entre elas. A ferramenta de diagnóstico é composta por dois *probes*, um sensor e um gerador de carga. O *probe* sensor executa, em uma MV, um servidor *web* Apache e uma aplicação em duas fases, uma para diagnosticar contenção de acesso à rede e uma para identificar contenção ocasionada por acesso à memória. O gerador de carga envia as requisições ao servidor *web* e define a fase do *probe* sensor; a pressão sobre os recursos é encontrada quando o tempo de resposta da aplicação é degradado e a causa é identificada pela fase em que o *probe* sensor se encontra. Apesar de bastante eficiente para identificar a contenção ocasionada pela interferência, a proposta é mais intrusiva sobre o desempenho da aplicação do que o nosso modelo, uma vez que implica em maior *overhead*. Como o foco do artigo é outro, não há propostas para identificar as situações adequada e de superprovisionamento. Além disto, a proposta deste trabalho é baseada em um diagnóstico realizado pelo cliente, enquanto no artigo o diagnóstico é mais interessante ao provedor de nuvens.

(PRAS et al., 2009) propõe avaliar a variabilidade do consumo de largura de banda, associada à utilização média, para dimensionar enlaces de rede. A fila nas interfaces é usada como indicativo da variabilidade do consumo de banda. Em comparação com a proposta da

seção 3.2, nota-se que tanto a variabilidade da largura de banda como a fila das interfaces são consideradas. No entanto, a medida da variabilidade é diferente, e em nosso trabalho o enfileiramento é usado para identificar o subprovisionamento da rede.

Alguns trabalhos (DU; SEHRAWAT; ZWAENEPOEL, 2011; NIKOLAEV; BACK, 2011) propõem o uso de contadores de desempenho de *hardware* – registradores que contabilizam a ocorrência de eventos de baixo nível, como *cache hits/misses* e instruções executadas – para a monitoração de ambientes virtuais. Embora a monitoração em nível de *hardware* geralmente tenha um *overhead* menor do que em nível de sistema operacional (como a proposta neste trabalho), contadores de desempenho são mais apropriados para *profiling* de aplicações individuais, enquanto o nosso foco está na análise de MVs como um todo. Ademais, as métricas usadas em nosso trabalho são bem mais portáveis entre monitores de máquinas virtuais e arquiteturas de *hardware* do que os contadores de desempenho: mesmo existindo no processador, estes nem sempre estão acessíveis às máquinas virtuais, além de serem específicos de cada processador (o conjunto de contadores disponíveis varia mesmo entre diferentes processadores da mesma arquitetura, como Intel x86).

**Provisionamento dinâmico de recursos.** Uma ferramenta para realizar provisionamento de recursos orientado ao cumprimento de SLA é apresentada em (BUYYA; GARG; CALHEIROS, 2011). As requisições de recursos do usuário são alocadas sob a avaliação do cumprimento de SLA e métricas de qualidade de serviço (QoS). Assume-se que o cliente conhece as necessidades de suas aplicações, e que a ferramenta conhece as aplicações em execução. As principais diferenças em relação a este trabalho são que, no nosso caso, as métricas monitoradas não estão associadas a uma aplicação específica, e o cliente não precisa conhecer seus SLOs *a priori*, pois o provisionamento é diagnosticado em tempo de execução.

Baruchi e Midorikawa (BARUCHI; MIDORIKAWA, 2010) propõem uma estratégia para provisionamento elástico de memória no Xen. A proposta consiste em identificar quando o recurso está subprovisionado ou superprovisionado, ajustando a alocação de memória de acordo com o diagnóstico. Para detectar estes dois diagnósticos é

calculada a média móvel exponencial ponderada da utilização de memória de duas escalas temporais diferentes (5 s e 25 s), analisando, para os intervalos de 25 s, quando as médias de 5 s cruzam as médias de 25 s para cima (aumento da demanda de memória) ou para baixo (diminuição da demanda de memória). Resultados experimentais mostram que esta abordagem é suficiente para cargas de trabalho *I/O-bound*, mas é menos eficaz nas cargas de trabalho dependentes de CPU. O modelo de diagnóstico proposto por este trabalho poderia aprimorar o mecanismo baseado na média móvel para detectar alocações sub- ou superprovisionadas. Uma diferença importante é que a proposta utilizada no artigo aparentemente incorpora os valores utilizados por *buffers* de E/S e *cache* de páginas, o que pode indicar uma demanda de memória menos precisa, uma vez que o Linux eventualmente utiliza a memória livre com propósitos de *caching* (GORMAN, 2004).

O uso de técnicas de aprendizado de máquina para provisionamento de recursos em nuvens é proposto por (RAO et al., 2011; KUNDU et al., 2012; VASIĆ et al., 2012). Esses trabalhos associam a utilização de recursos com medidas de desempenho das aplicações (como tempo de resposta) e usam classificadores estatísticos para determinar a classe à qual pertence uma aplicação e, consequentemente, os recursos que devem ser alocados para sua execução. Além do foco em aplicações e não em máquinas virtuais, os resultados demonstram que essa abordagem funciona para cargas de trabalho estáticas, mas tem problemas com cargas variáveis e/ou desconhecidas.

Em (HEO et al., 2009) é proposta uma ferramenta para gerenciar a alocação de processador e memória a MVs sobre a plataforma Xen, possibilitando o *overbooking*. A utilização de ambos recursos é observada periodicamente; a partir dessas observações e de um valor objetivo, calcula-se a quantidade de recursos necessária. No entanto, como é considerada apenas a média de utilização e não a sua variação, o tempo de resposta da aplicação pode ser degradado. Além disso, em processador é necessário observar o tempo de resposta da aplicação, o que difere da proposta deste trabalho.

(SUDEVALAYAM; KULKARNI, 2011) usam monitoração para avaliar o impacto no consumo de processador quando duas máquinas virtuais que se comunicam entre si são alocadas no mesmo *hardware*, e

propõem um modelo analítico para estimar a utilização de CPU quando as MVs são alocadas na mesma máquina física ou em máquinas separadas. A análise se restringe à plataforma Xen, e o único recurso considerado é o processador.

Um algoritmo para provisionar máquinas virtuais levando em conta a correlação na demanda por recursos entre as MVs é proposto em (HALDER; BELLUR; KULKARNI, 2012). MVs correlacionadas, que são aquelas com picos de demanda nos mesmos intervalos de tempo, são alocadas em máquinas físicas distintas, enquanto MVs não correlacionadas podem ser alocadas na mesma máquina. Além de considerar apenas processador, o algoritmo pressupõe cargas bem conhecidas, que se repetem sazonalmente, uma premissa que não é necessária em nossa abordagem.

Uma abordagem para identificação de demanda e provisionamento dinâmico de memória é apresentada em (ZHAO; WANG; LUO, 2009). A proposta é utilizar um histograma de LRU para calcular o WSS (*Working Set Size*) de uma máquina virtual, e assim, provisionar esta quantidade de memória à MV. Nas situações em que o WSS pode ser maior que a memória física disponível, agrega-se o percentual de utilização em *swap*. A construção do histograma de LRU é feita no MMV e a utilização de *swap* é obtida através de um monitor inserido na MV. O artigo mostra, através de experimentação, que a abordagem é eficiente para identificar a demanda e provisionar corretamente a memória. A principal diferença para o nosso trabalho se dá na dependência com o provedor, no nosso caso o modelo é independente de provedor, ou seja, pode ser aplicado sem o conhecimento do mesmo, já na proposta do artigo a ferramenta é inserida no MMV que somente é acessível ao provedor. Além disto, entende-se que a construção do histograma de LRU em máquina virtual pode ocasionar um *overhead* maior sobre o desempenho do que o observado em MMV, uma vez que a memória provisionada à MV também será usada para a construção da lista de referências as páginas.

Em (WOOD et al., 2009) é apresentado o *Sandpiper*, uma ferramenta para automatizar o processo de monitoração e detecção de gargalos de desempenho em ambientes virtualizados, migrando e reconfigurando as MVs quando necessário. O sistema utiliza duas técnicas

para monitoração para identificar os gargalos, a monitoração *black-box* e a monitoração *gray-box*. Na primeira técnica os dados de monitoração são capturados a nível de MMV, enquanto que, na segunda, são obtidos dados em nível de SO convidado. O uso da segunda abordagem é justificado pela necessidade de identificar de forma mais precisa os gargalos de desempenho, principalmente no caso de memória, utilizando valores de utilização de *swap*. A ferramenta monitora e provisiona as capacidades de processador, rede e memória, em uma abordagem bastante similar a este trabalho, pois também são utilizados limiares para caracterizar os recursos. A principal diferença é que a ferramenta proposta por Wood et al. (2009) executa em nível de MMV para monitorar e ajustar as capacidades, utilizando tanto os dados internos da MV quanto do MMV, sendo assim dependente de provedor, o que não é o caso do modelo de diagnóstico proposto neste trabalho. Além disto, o trabalho não define categorias para superprovisionado e provisionamento adequado. Contudo, apesar da diferença, as métricas propostas neste trabalho poderiam auxiliar o *Sandpiper* na identificação de gargalos, aprimorando o provisionamento.

## 3.6 CONSIDERAÇÕES DO CAPÍTULO

Este capítulo apresentou o modelo de diagnóstico de recursos para nuvens computacionais IaaS, este modelo classifica cada recurso em três categorias, subprovisionado, superprovisionado ou adequadamente provisionado. Definiu-se que um recurso está subprovisionado quando o acréscimo de capacidade melhora significativamente o desempenho, já o superprovisionamento caracteriza-se pela situação onde adicionar ou reduzir a capacidade provisionada não gera influência perceptível no desempenho. Por fim, o provisionamento adequado ocorre na região intermediária entre sub e superprovisionamento, ou seja, reduzir a capacidade degrada o desempenho enquanto aumentar não gera melhoras significativas.

O modelo avalia cada recurso individualmente, com métricas associadas a limiares para determinar em qual classificação o recurso se encontra. As métricas observadas foram a utilização de CPU e

*%Steal* para processador, enfileiramento e variação da largura de banda para rede, e memória residente e reservada para memória.

Além da definição de métricas e limiares, este capítulo também apresenta uma sugestão de ajuste de provisionamento. A ideia foi indicar uma ordem de ajuste de recursos nas situações em que há empate nos diagnósticos. Como este trabalho privilegia o desempenho, a proposta foi de ajustar primeiramente os recursos em estado de subprovisionamento, para depois ajustar os recursos superprovisionados. Nos casos de empate, a ordem definida foi, rede, processador e memória, respectivamente.

No próximo capítulo será realizada a verificação do modelo proposto, avaliando em diferentes cenários se os diagnósticos apontados condizem com o desempenho observado nas aplicações.

# 4 AVALIAÇÃO EXPERIMENTAL DO MODELO

A fim de verificar as propostas definidas no Capítulo 3 para diagnóstico do provisionamento de recursos em nuvem, fez-se uma avaliação experimental. Os recursos foram monitorados a partir da execução de diferentes cargas de trabalho que representam cenários variados de utilização dos recursos. A metodologia utilizada foi de avaliar o modelo de duas formas, com recursos isolados e com recursos combinados. Quando os recursos foram avaliados isoladamente, as capacidades de alocação foram ajustadas um recurso por vez. Na avaliação de recursos combinados, a capacidade destinada à MV foi uma combinação entre os recursos disponíveis, visando observar a influência de um recurso sobre o outro no diagnóstico.

## 4.1 AMBIENTES DE TESTES

Foram utilizados dois ambientes de testes cada um composto por dois computadores idênticos, *Cenário-1* e *Cenário-2*. No *Cenário-1* a característica dos computadores é: processador AMD Phenom II X4 B93 2,8 GHz (*quad-core*), 4 GB de memória RAM e disco 500 GB SATA (7200 rpm). A conexão entre as máquinas é realizada por uma rede local de 100 Mbps. A capacidade de processador, rede e memória providos à MV variou de acordo com experimento realizado.

No *Cenário-2*, utilizou-se os equipamentos disponíveis no *cluster edel* sob a infraestrutura da GRID5000[1], composto por 72 nós com processador Intel Xeon E5520 2,27 GHz (com dois processadores *quad-core*) e 24 GB de memória RAM cada, interligados por uma rede 1Gbps.

Em ambos os cenários a característica de *software* utilizada foi idêntica, foi instalado o sistema operacional Debian Squeeze 64 bits, com *kernel* 2.6.32-5, e sobre este instalou-se o *hypervisor* Xen na versão 4.0.1. As máquinas virtuais utilizam sistema operacional paravirtualizado Debian Squeeze 64 bits, com *kernel* 2.6.32-5.

[1] https://www.grid5000.fr/

## 4.2 FERRAMENTAS DE *BENCHMARK* E MONITORAÇÃO

Inspirado por trabalhos relacionados ((WOOD et al., 2007), (VASIĆ et al., 2012), (KUNDU et al., 2012), (MUKHERJEE et al., 2013)) foram utilizadas diferentes ferramentas de *benchmark* e monitoração. Cada ferramenta foi aplicada no diagnóstico do recurso alvo da mesma (processador, rede ou memória), ou seja, aquele que deva ser o gargalo de seu funcionamento. As ferramentas de *benchmark* utilizadas foram:

- *stress*[2]: A ferramenta permite que sejam criados processos do tipo *CPU-Bound* e *IO-Bound*. O usuário pode determinar a proporção entre esses dois tipos de processos, bem como o tempo total de execução. Essas características permitem que diversos níveis de carga sejam aplicados ao processador, sendo, portanto, usado para diagnosticar o impacto junto ao mesmo.

- *httperf*[3] versão 0.9.0-1: Esta ferramenta gera cargas de trabalho a servidores *web*, o princípio é requisitar páginas de um servidor em intervalos de tempo regulares parametrizáveis, permitindo a obtenção de métricas de desempenho no atendimento destas requisições, tais como tempo de resposta médio e utilização dos recursos do cliente. Tendo em vista essas características, a ferramenta foi usada para gerar cargas e observar o comportamento do recurso de rede.

- RUBiS: *benchmark* que simula cargas de requisições realizadas a um site de busca e compra de produtos (similar ao Ebay). Os parâmetros de configuração permitem especificar o tamanho da base de dados e caracterizar as requisições em número, tipo e distribuição (CECCHET et al., 2003). Esta ferramenta não tem um recurso específico como foco de saturação. O seu objetivo é o uso controlado (simulação) de uma aplicação real.

- NAS: Conjunto de ferramentas de *benchmark* para avaliação de desempenho de aplicações paralelas (BAILEY et al., 1991). O

[2]http://weather.ou.edu/ apw/projects/stress/

[3]http://www.hpl.hp.com/research/linux/httperf/

recurso influenciável depende da ferramenta escolhida. Em geral os testes são impactados pela capacidade de processador disponível, mas também podem ser afetadas pela capacidade de memória. Do conjunto de aplicações disponíveis (EP, CG, MG, IS, etc) escolheu-se IS, por se tratar de uma aplicação que possui comportamento de uso de recursos equilibrado. A aplicação IS não tem característica de saturação constante de um único recurso, como ocorre com EP, que satura o processador praticamente em todo o tempo de execução.

- DaCapo: é um conjunto de ferramentas de *benchmark* que visam avaliar o desempenho do recurso de memória, com base em cargas de trabalho de aplicações reais (BLACKBURN et al., 2006). Para este trabalho foram utilizados os testes H2 e Tradesoap, uma vez que apresentaram os três possíveis diagnósticos de acordo com a quantidade de memória provisionada.

A monitoração dos recursos foi realizada a nível de sistema operacional nas máquinas virtuais. Para capturar as métricas de processador utilizou-se as ferramenta *sar*, do pacote *sysstat*[4]. Para monitorar o consumo de largura de banda, foram extraídas estatísticas de `/proc/net/dev`, que discriminam a banda consumida por interface. Para monitorar a fila da interface de rede, utilizou-se a ferramenta de configuração e gerenciamento de interface de rede *tc*[5], que fornece, entre outras estatísticas, a quantidade de pacotes na fila da interface. Nos experimentos envolvendo memória a monitoração foi realizada com base nas estatísticas extraídas de `/proc/stat/meminfo`.

As métricas foram observadas a cada 1 s, o que permite acompanhar a variabilidade de comportamento das MVs; além disso, as ferramentas *sar* e *stress* têm granularidade mínima de 1 s. Como o *overhead* de monitoração das métricas escolhidas é baixo (*sar* consumiu 0,06% de CPU nos testes), esse mesmo intervalo pode ser usado em ambientes de produção, uma vez que é pouco intrusivo sobre o desempenho das aplicações. O diagnóstico foi realizado com base no período de execução das ferramentas, o que é suficiente para os propó-

[4]http://sebastien.godard.pagesperso-orange.fr/

[5]http://lartc.org/manpages/tc.txt

sitos da verificação da validade do diagnóstico para a carga de trabalho executada. Em ambientes de produção, esse período de diagnóstico deve ser compatível com o período de tarifação do provedor IaaS, que tipicamente é de 1 h (SULEIMAN et al., 2012).

Para realizar a validação cruzada do diagnóstico e avaliar o comportamento das ferramentas com a evolução das capacidades dos recursos, foram capturadas métricas de desempenho das aplicações. Para as ferramentas *httperf* e RUBiS utilizou-se a média dos tempos de resposta, enquanto nos casos de NAS e DaCapo foram observados os tempos totais de execução. É importante reforçar que essas métricas de desempenho de aplicações não são necessárias no modelo, sendo usadas apenas para fins de validação de resultados.

## 4.3 DIAGNÓSTICO DE RECURSOS ISOLADOS

### 4.3.1 Metodologia do experimento

No diagnóstico de recursos isolados a metodologia utilizada consiste em variar a capacidade provisionada a cada um dos recursos (processador, rede e memória), executando a carga de trabalho e monitorando as métricas definidas no modelo. Depois de realizados os experimentos, o modelo de diagnóstico foi aplicado com base nos dados obtidos na monitoração em intervalos definidos. O resultado do diagnóstico foi comparado com as métricas de aplicação.

Para que o diagnóstico de cada recurso não fosse prejudicado, destinou-se capacidade de recurso suficiente aos demais para que eles não se tornassem gargalos de desempenho. Neste caso, fez-se o uso das capacidades de *hardware* do *Cenário-1*.

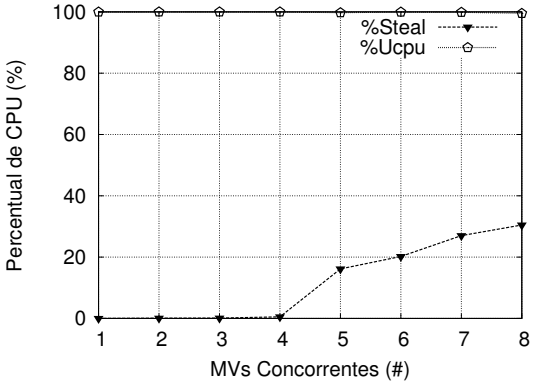
### 4.3.2 Diagnóstico do Provisionamento de Processador

Na avaliação de provisionamento do processador, foram definidos três cenários de carga sobre o recurso: *CPU-Bound*, *Misto* e *IO-Bound*. No cenário *CPU-Bound*, a ferramenta *stress* executa carga

intensiva de processador sobre cada MV. No cenário *IO-Bound*, os processos criados por *stress* executam essencialmente operações em disco. No cenário *Misto*, os processos alternam entre execuções no processador e requisições a disco. O tempo total de execução da ferramenta *stress* foi dividido em intervalos de 10 segundos, sendo que os tempos para execução em processador e disco definidos são 2, 5 e 8 s. Portanto, são avaliados três cenários *Mistos*: carga de processador maior que a de disco (8 s processador/2 s disco), cargas equivalentes (5 s processador/5 s disco), e carga de disco maior que a de processador (2 s processador/8 s disco).

Em todos os cenários (*CPU-Bound*, *Misto* e *IO-Bound*) variou-se o número de máquinas virtuais com execução da carga e monitorou-se as métricas *%Steal* e *%Idle* (de onde deriva-se $U_{cpu}$). Os gráficos das Figuras 8 a 12 mostram a média dos pontos observados nos 60 s de monitoração.

Figura 8 – Média de $U_{cpu}$ e *%Steal* (período de 60 s, 1–8 MVs concorrentes) para um cenário *CPU-Bound*

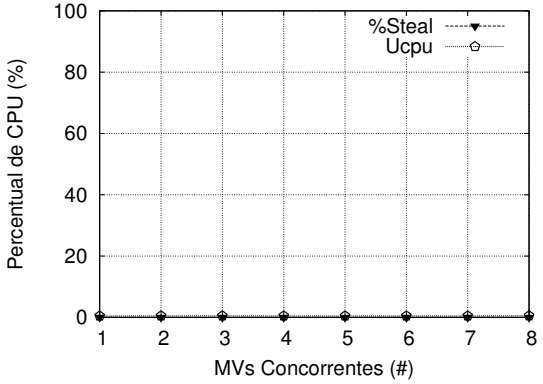



Fonte: o autor

Ao observar o gráfico da Figura 8, é possível perceber que, na execução de cargas *CPU-Bound*, o processador físico se torna saturado a partir de cinco máquinas virtuais, pois possui quatro núcleos de processamento. Isso pode ser verificado pelo surgimento de *%Steal* na monitoração do consumo de processador das MVs. Como $\overline{U_{cpu}}$ é igual

a 100%, o processador está subprovisionado.

Figura 9 – Média de $U_{cpu}$ e *%Steal* (período de 60 s, 1–8 MVs concorrentes) para um cenário *IO-Bound*




Fonte: o autor

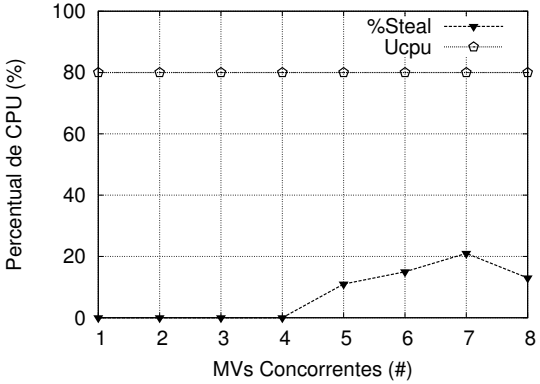
Os dados relacionados à carga *IO-Bound* (Figura 9) apresentam comportamento oposto da carga *CPU-Bound*. Para todos os pontos, o percentual de utilização manteve-se em 0% e não há presença de *%Steal*, caracterizando superprovisionamento de CPU.

Na execução de carga mista com maioria do tempo em processador (Figura 10), a média de $U_{cpu}$ se mantém em 80% para qualquer número de máquinas virtuais, o que indica subprovisionamento. Além disto, observa-se que a partir de cinco máquinas virtuais em execução simultânea há ocorrência de *%Steal*, indicando que não há CPU física disponível para todas as CPUs virtuais, isto reforça o subprovisionamento.

Nos casos de 50% e 20% de utilização de CPU (Figuras 11 e 12), diagnostica-se subprovisionamento a partir de 5 MVs em execução simultânea, pois, existe a ocorrência de *%Steal* nestes cenários. Entretanto, onde não há ocorrência de *%Steal* o diagnóstico sobre o provisionamento não pode ser obtido pela simples avaliação da média geral, pois a utilização de CPU pode não ser constante.

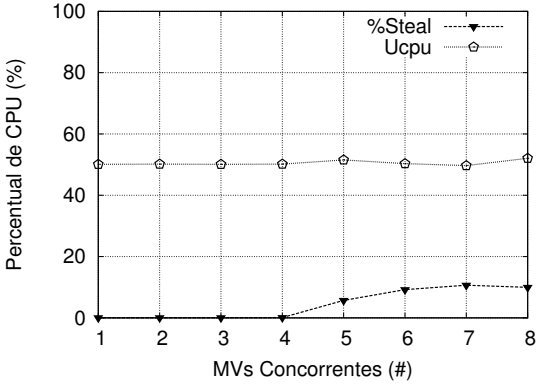
Por exemplo, o cenário em que observa-se 20% de utilização (Figura 12) poderia ser gerado por cargas diferentes, uma com utiliza-

Figura 10 – Média de $U_{cpu}$ e *%Steal* (período de 60 s, 1–8 MVs concorrentes) para um cenário de carga mista com 80% em CPU




Fonte: o autor

Figura 11 – Média de $U_{cpu}$ e *%Steal* (período de 60 s, 1–8 MVs concorrentes) para um cenário de carga mista com 50% em CPU

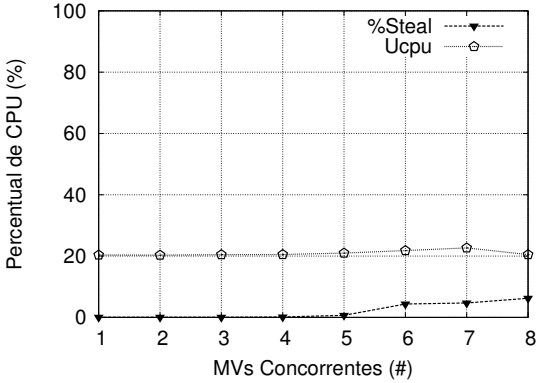



Fonte: o autor

ção de CPU constante em 20% e outra com variações entre alta e baixa utilização, cuja média também resulta em 20%. Utilizar o mesmo diagnóstico nestes dois casos (de superprovisionamento) poderia prejudicar a carga que tem picos de utilização. Sendo assim, de acordo com o modelo, quando a utilização é inferior à 50% deve-se avaliar a variação na

utilização do recurso.

Figura 12 – Média de $U_{cpu}$ e *%Steal* (período de 60 s, 1–8 MVs concorrentes) para um cenário de carga mista com 20% em CPU



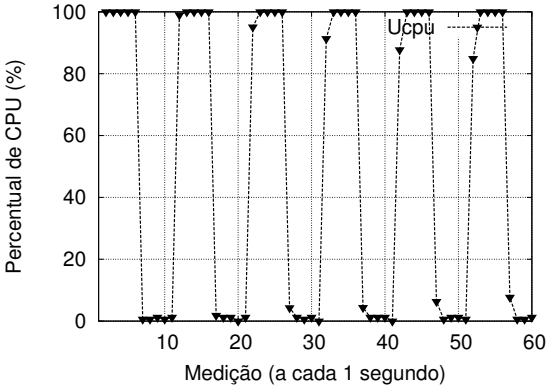

Fonte: o autor

Sendo assim, para os dois casos com utilização média de CPU inferior a 50% e sem *%Steal*, plotou-se os gráficos das Figuras 13(a) e 13(b), onde todos os pontos medidos para uma MV em execução são apresentados.

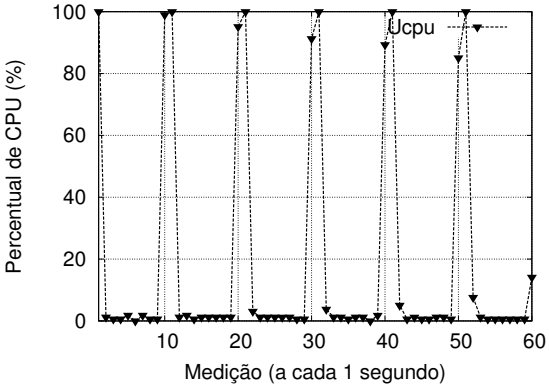
O cenário de carga mista equilibrada (Figura 13(a)), com $\overline{U_{cpu}} = 49{,}95\%$, apresenta $N' = 26$ (Equação (3.2)) e $\varphi = 43\%$ (Equação (3.3)); adotando $\omega = 20\%$ na Tabela 1, o provisionamento é considerado adequado. No cenário com maior carga de disco (Figura 13(b)), $\overline{U_{cpu}} = 20\%$, $N' = 8$ e $\varphi = 13\%$, e o diagnóstico é de superprovisionamento.

Entende-se que os comportamentos observados nos experimentos realizados com a ferramenta *stress* são condizentes com os diagnósticos propostos. Entretanto, como o *benchmark* executa por períodos de tempo definidos pelo usuário, o diagnóstico não pôde ser verificado com métricas de usuário (tal como tempo de resposta), uma vez que o tempo de execução será o mesmo independentemente da capacidade de CPU disponível. Sendo assim, decidiu-se por avaliar o comportamento de aplicações onde o tempo de execução varia de acordo com a capacidade do recurso. Para tanto, avaliou-se o tempo de execução do *benchmark* NAS-IS (Figura 14), de acordo com a variação da capaci-

Figura 13 – Utilização de CPU no período de 60 s com 1 MV em execução




(a) Misto, 50% CPU



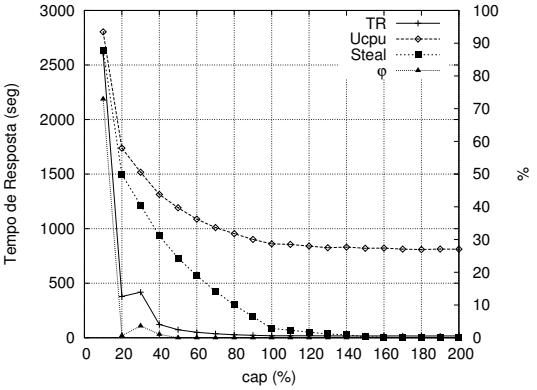

(b) Misto, 20% CPU

Fonte: o autor

dade de CPU disponível entre 10 e 200% (10% é equivalente a 10% de uma VCPU e 200% é equivalente a duas VCPUs).

Ao observar os resultados apresentados pela Figura 14 e associados com o diagnóstico proposto pela Tabela 3, pode-se dizer que o comportamento segue de acordo com o esperado, uma vez que é fidedigno com o tempo de resposta observado para a aplicação. Observa-se no gráfico que *%Steal* é superior ao limiar (1%) para valores de cap in-

Figura 14 – Tempo de execução da ferramenta NAS-IS de acordo com a capacidade de CPU disponível à MV com monitoração das métricas *%Steal*, $U_{cpu}$ e $\varphi$




Fonte: o autor

feriores a 100%, o que caracteriza o recurso como subprovisionado, ou seja, adicionar mais capacidade de processador à MV deve melhorar o desempenho, o que é acompanhado pelo tempo de resposta.

Visualmente a variação do tempo de resposta não é perceptível, mas a análise sobre os dados mostra que há uma queda de 17% dos pontos com cap 80 para cap 90 e de 18% de 90 para 100. A partir deste ponto o percentual de CPU em *Steal* é inferior ao limiar, que aliado ao percentual de utilização inferior a 50% e $\varphi$ inferior a 20%, diagnostica o recurso como superprovisionado, ou seja, adicionar mais capacidade de processador não influencia o desempenho da aplicação, o que é verificado pelo comportamento do tempo de resposta.

### 4.3.3 Diagnóstico do Provisionamento de Rede

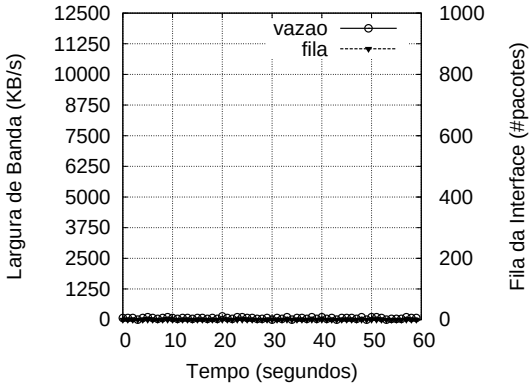
Para validar o diagnóstico do provisionamento de rede, foi usada uma máquina virtual com capacidade de largura de banda limitada no MMV através do arquivo de configuração da MV. A banda alocada variou entre 10 Mbps e 100 Mbps, em intervalos de 10 Mbps. Definiram-se três cargas de trabalho, de acordo com o tamanho do ar-

quivo requisitado ao servidor *web* (32 KB, 4 MB, 50 MB). Em todos os casos, havia cinco clientes simultâneos, com intervalo de chegada de requisições entre 1 e 5 s, de acordo com uma distribuição uniforme.

O arquivo de 32 KB foi escolhido para manter um comportamento de subutilização para qualquer largura de banda provisionada e número de clientes simultâneos. O arquivo de 50 MB foi utilizado porque, no intervalo de requisições realizadas, e para qualquer largura de banda utilizada, a rede ficará saturada. Por fim, em um cenário intermediário, foram usados arquivos de 4 MB, que deveriam saturar a rede quando a banda provisionada fosse pequena, mas não quando ela fosse aumentada além de certo ponto no intervalo entre 10 e 100 Mbps.

As requisições de 32 KB e 50 MB exibem comportamentos semelhantes, independente da largura de banda provisionada; assim, são mostrados os resultados somente para os extremos de cada cenário (10 Mbps e 100 Mbps). Os gráficos plotados nas Figuras 15 e 16 mostram os dados de consumo de largura de banda para requisições ao arquivo de 32 KB.

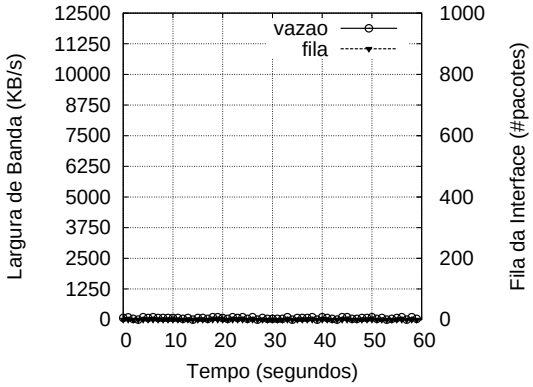
Figura 15 – Consumo de largura de banda e enfileiramento no ambiente com requisições a um arquivo de 32KB com provisionamento de 10 Mbps. A média de largura de banda é 57,8 KB/s e o CV é 53,3%




Fonte: o autor

Nos resultados para o arquivo de 32 KB observa-se um baixo consumo de largura de banda, aproximadamente 57 KB/s nas duas bandas provisionadas, com comportamento não constante (coeficiente de

Figura 16 – Consumo de largura de banda e enfileiramento no ambiente com requisições a um arquivo de 32KB com provisionamento de 100 Mbps. A média de largura de banda é 57,7 KB/s e o CV é 57,1%




Fonte: o autor

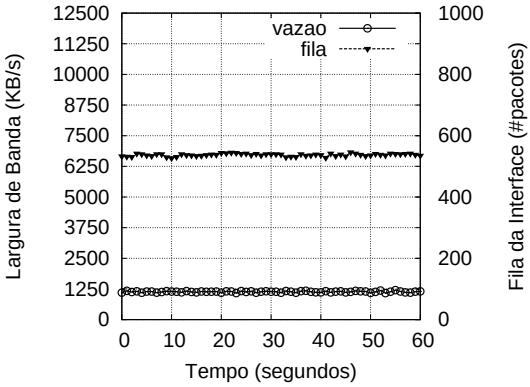
variação entre 50% e 60%) e sem enfileiramento na interface de rede. Portanto, o diagnóstico deste caso é definido como superprovisionado, conforme expresso pela Tabela 2 (Seção 3.2).

O comportamento oposto ao dos testes com 32 KB é observado nas requisições ao arquivo de 50 MB, onde os dados de consumo de largura de banda e enfileiramento, plotados nas Figuras 17 e 18, indicam o subprovisionamento. Em ambos os casos, verifica-se a ocorrência de fila e o consumo de banda mantém-se constante ($CV_{lb} < 5\%$) sobre a largura de banda provisionada. No gráfico da Figura 18, a queda abrupta no tamanho da fila em t=13 s é irrelevante para o modelo, uma vez que avalia-se apenas a existência de fila, não a sua variabilidade.

Para o cenário com arquivos de 4 MB, foram obtidos três diagnósticos diferentes, dependendo da largura de banda alocada: subprovisionamento entre 10 e 50 Mbps, provisionamento adequado com 60 Mbps e superprovisionamento entre 70 e 100 Mbps. Os gráficos da Figura 19 mostram o intervalo em que há a mudança no diagnóstico.

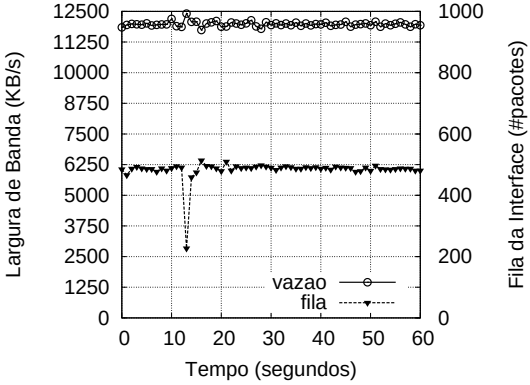
Em termos do modelo proposto na Seção 3.2, verifica-se que, com largura de banda igual a 50 Mbps o ambiente está subprovisionado (Figura 19(a)), ou seja, é necessário mais largura de banda, uma vez que o consumo de banda é constante ($CV_{lb} < 5\%$) e ocorre enfilei-

Figura 17 – Consumo de largura de banda e enfileiramento no ambiente com requisições a um arquivo de 50MB com provisionamento de 10Mbps. A média de largura de banda é 1136 KB/s e o CV é 2,4%




Fonte: o autor

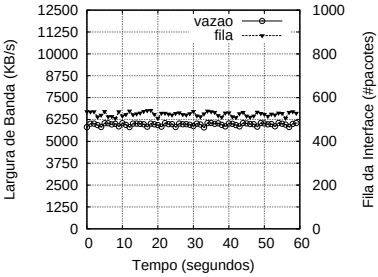
Figura 18 – Consumo de largura de banda e enfileiramento no ambiente com requisições a um arquivo de 50MB com provisionamento de 100Mbps. A média de largura de banda é 11979 KB/s e o CV é 0,8%



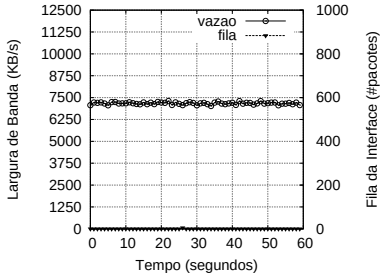

Fonte: o autor

ramento. Quando o limite estabelecido é de 60 Mbps (Figura 19(b)), a Tabela 2 indica que o provisionamento realizado é adequado, pois o comportamento é constante e não há enfileiramento. A partir de

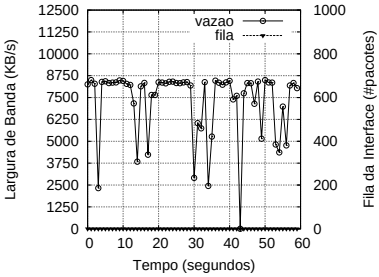
Figura 19 – Representação de cenário com busca por largura de banda ideal, com requisições a arquivos de 4 MB. M é a média do consumo de banda, e CV o coeficiente de variação. O consumo é constante com banda de 50 Mbps e 60 Mbps, e inconstante com 70 Mbps. A partir de 60 Mbps, a fila na interface desaparece.




(a) 4MB-50Mbps (M 5958 KB/s, CV 1,3%)




(b) 4MB-60Mbps (M 7173 KB/s, CV 1,9%)




(c) 4MB-70Mbps (M 7276 KB/s, CV 26,5%)

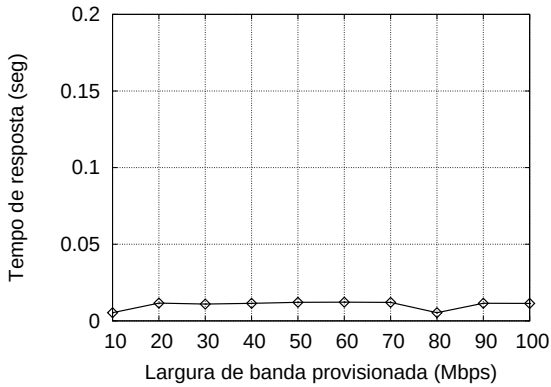
Fonte: o autor

70 Mbps (Figura 19(c)), a análise da Tabela 2 indica que o recurso foi superprovisionado e a largura de banda pode ser reduzida, uma vez que o comportamento não é constante ($CV_{lb} > 5\%$). Portanto, observa-se que o raciocínio subjacente às equações da Seção 3.2 é consistente com os comportamentos verificados experimentalmente.

Para reforçar essa conclusão, observou-se o tempo de resposta da aplicação (no caso o *httperf*) em cada cenário (tamanho de arquivo e largura de banda alocada). As Figuras 20 a 22 apresentam a média

dos tempos de resposta entre os cinco clientes. Para requisições de 32 KB (Figura 20), o tempo de resposta praticamente não é influenciado pela variação da largura de banda alocada, com ganhos na ordem de milissegundos.

Figura 20 – Média do tempo de resposta das requisições a um arquivo de 32 KB de acordo com a largura de banda provisionada.

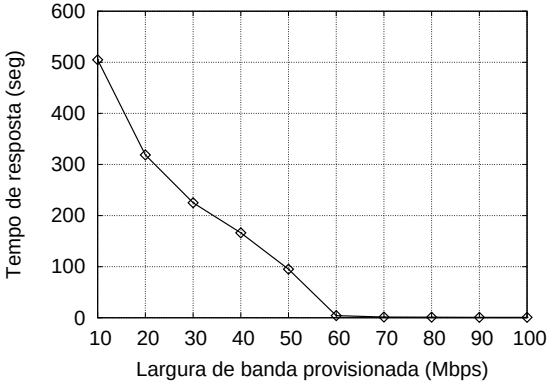



Fonte: o autor

No cenário da Figura 21, onde os arquivos requisitados são de 4 MB, verifica-se que o tempo de resposta cai rapidamente à medida em que a largura de banda disponível aumenta de 10 Mbps para 60 Mbps, e que os ganhos a partir de 60 Mbps são sensivelmente menores. Isso reforça a fidedignidade do modelo de diagnóstico quanto ao comportamento das aplicações.

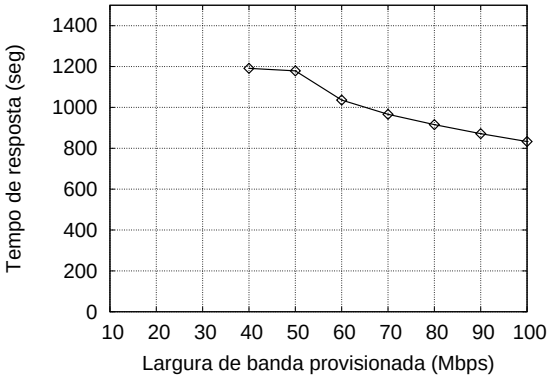
Por fim, as requisições de 50 MB (Figura 22) obtêm uma redução perceptível no tempo de resposta com o aumento da banda; neste gráfico, quando a largura de banda era inferior a 40 Mbps os tempos de resposta foram maiores do que o intervalo de monitoração, e por isso os dados correspondentes não estão mostrados. Os resultados apresentados corroboram o diagnóstico do provisionamento de rede dado pela Tabela 2. Em particular, o gráfico da Figura 21 demonstra que, dependendo do valor tarifado por Mbps, o ganho de desempenho pode não justificar um aumento de banda além de 60 Mbps.

Figura 21 – Média do tempo de resposta das requisições a um arquivo de 4 MB de acordo com a largura de banda provisionada.




Fonte: o autor

Figura 22 – Média do tempo de resposta das requisições a um arquivo de 50 MB de acordo com a largura de banda provisionada.




Fonte: o autor

### 4.3.4 Diagnóstico de Provisionamento de Memória

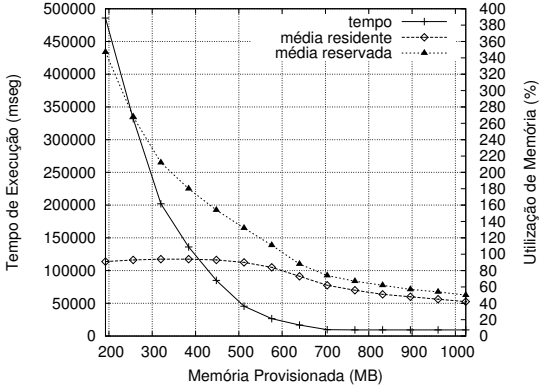
De acordo com o apresentado na Seção 3.3, o modelo proposto para diagnóstico do provisionamento de memória baseia-se em duas métricas, a memória residente ($\overline{m_r}$) e memória reservada ($\overline{m_c}$), que

comparadas a limiares ($\alpha$, $\beta$ e $\gamma$) definem o diagnóstico. O subprovisionamento é observado quando $\overline{m_r}$ é alto, ou quando é confortável e $\overline{m_c}$ é relevante. Por sua vez, o provisionamento adequado é diagnosticado quando há memória residente confortável e memória reservada irrelevante, ou ainda, quando há baixa memória residente e memória reservada relevante. No último caso identifica-se o superprovisionamento, quando observa-se baixo $\overline{m_r}$ e $\overline{m_c}$ irrelevante.

Nos experimentos para diagnosticar a memória os recursos processador e rede foram superprovisionados e a capacidade de memória alocada para MV foi variada. Nas aplicações DaCapo H2, DaCapo Tradesoap e NAS IS o intervalo de memória provisionado foi entre 192 MB e 1024 MB a passos de 64 MB e para a ferramenta RUBiS provisionou-se memória entre 80 MB e 192 MB a passos de 16 MB.

Os resultados obtidos com DaCapo H2 são apresentados na Figura 23. O diagnóstico de subprovisionamento é obtido entre 192 MB e 640 MB, pois tem-se memória residente alta, com o menor valor igual a 73% em 640 MB. Entre 704 MB e 832 MB tem-se provisionamento adequado: $\overline{m_r}$ se encontra entre 62% e 51% (região confortável), e $\overline{m_c}$ apresenta-se inferior a 80% (irrelevante). A partir de 896 MB o diagnóstico é de superprovisionamento, uma vez que $\overline{m_r}$ é inferior a 50% (baixo) e $\overline{m_c}$ é inferior a 60% (irrelevante).

Figura 23 – Tempo de execução (eixo *y* à esquerda) e utilização de memória (eixo *y* à direita) para DaCapo H2.




Fonte: o autor

A Tabela 5 apresenta a evolução do tempo de execução de acordo com a memória alocada. No intervalo entre 192 MB e 640 MB, cada incremento de 64 MB resulta em queda significativa do tempo de execução (entre 31% e 46%); todas as alocações de memória neste intervalo foram diagnosticadas como subprovisionadas. O tempo de execução apresenta queda de 3% no intervalo de provisionamento adequado, de 9,6 s com 704 MB para 9,3 s com 832 MB. O tempo de execução com 1024 MB é o mesmo que o de 896 MB, 9,3 s, mostrando que não há ganho de desempenho a partir de 832 MB de memória alocada à MV (faixa com diagnóstico superprovisionado). Sendo assim, pode-se perceber que o diagnóstico obtido pelo modelo é consistente com o desempenho observado.

Tabela 5 – Tempos de execução para DaCapo H2 com variação de alocação de memória. "Diferença" é a variação comparada com a alocação anterior, e "'Diagnóstico" é a saída do modelo.

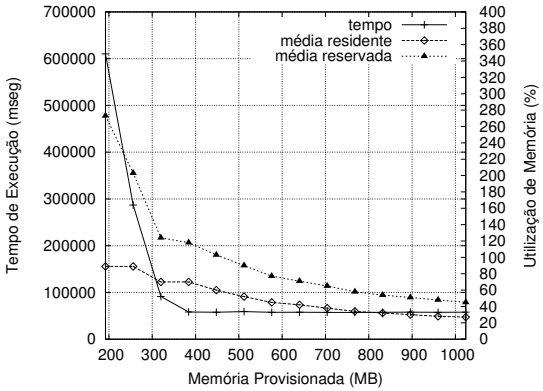
| Memoria (MB) | Tempo de Execução (ms) | Diferença (%) | Diagnóstico |
|---|---|---|---|
| 192 | 485.800,8 | — | sub |
| 256 | 332.105,4 | −31,63 | sub |
| 320 | 202.146,2 | −39,13 | sub |
| 384 | 136.093,8 | −32,67 | sub |
| 448 | 84.950,0 | −37,57 | sub |
| 512 | 45.581,6 | −46,34 | sub |
| 576 | 26.625,2 | −41,58 | sub |
| 640 | 16.891,4 | −36,55 | sub |
| 704 | 9.681,6 | −42,83 | adequado |
| 768 | 9.334,6 | −3,58 | adequado |
| 832 | 9.354,0 | +0,20 | adequado |
| 896 | 9.374,4 | +0,21 | super |
| 960 | 9.437,4 | +0,67 | super |
| 1024 | 9.361,2 | −0,80 | super |

Fonte: o autor

Os resultados obtidos com DaCapo Tradesoap são apresentados pela Figura 24. O diagnóstico de subprovisionamento é obtido entre 192 MB e 384 MB, pois tem-se memória residente alta. Entre

448 MB e 512 MB tem-se provisionamento adequado ($\overline{m_r}$ é confortável, $\overline{m_c}$ é irrelevante). As alocações de memória a partir de 576 MB estão superprovisionadas, pois memória residente e reservada situam-se, respectivamente, abaixo de 50% (baixo) e 80% (irrelevante). O tempo de execução na faixa de subprovisionamento cai de 610,0 s (192 MB) para 91,3 s (384 MB).

Figura 24 – Tempo de execução (eixo *y* à esquerda) e utilização de memória (eixo *y* à direita) para DaCapo Tradesoap.



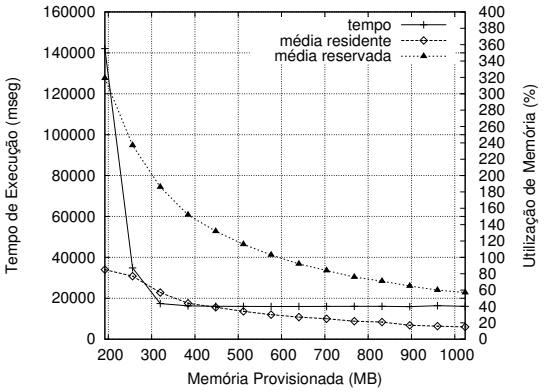

Fonte: o autor

Ainda sobre os resultados da ferramenta Tradesoap, o comportamento associado ao ponto de 384 MB (quarto ponto na Figura 24), poderia ser considerado adequado, uma vez que ele apresenta o mesmo desempenho que alocações maiores. Isto poderia ser considerado um erro de diagnóstico, todavia, este seria o único ponto incorreto nos experimentos realizados, e entende-se que qualquer mecanismo de diagnóstico baseado em limiares é suscetível a erros quando os dados se encontram muito próximos aos limiares. Uma análise sobre o tempo de execução mostra que alocações maiores que 576 MB apresentam mínimo impacto sobre o desempenho, corroborando o diagnóstico de superprovisionamento de memória.

Nos testes com NAS IS (Figura 25), o subprovisionamento é diagnosticado para o intervalo entre 192 MB e 320 MB; para 192 MB e 256 MB porque a memória residente é alta, enquanto para 320 MB

$\overline{m_r}$ é confortável e $\overline{m_c}$ é relevante. A memória está adequadamente provisionada com 384 MB ($\overline{m_r}$ baixo, $< 50\%$, e $\overline{m_c}$ relevante, $> 150\%$), e superprovisionada para alocações maiores que 384 MB ($\overline{m_c}$ reduz para valores irrelevantes). O tempo de execução reduz 87% de 192 MB para 320 MB, e mantem-se estável a partir de 320 MB, assim, os resultados do modelo estão de acordo com o comportamento observado.

Figura 25 – Tempo de execução (eixo *y* à esquerda) e utilização de memória (eixo *y* à direita) para NAS-IS.

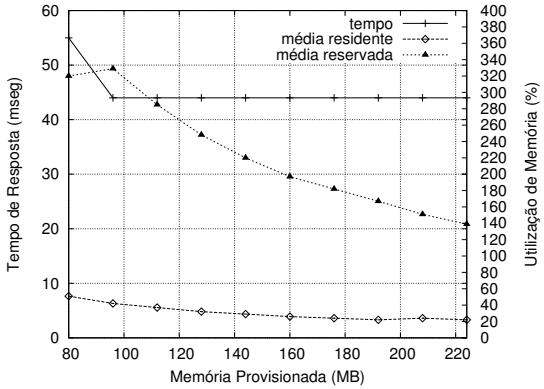



Fonte: o autor

Tendo em vista que o comportamento de RUBiS se estabiliza mais rapidamente que os outros *benchmarks*, esta aplicação foi avaliada sobre um intervalo mais curto de provisionamento (80–224 MB em vez de de 192–1024 MB), com um incremento menor (16 MB em vez de 64 MB). Os resultados para a ferramenta RUBiS são apresentados na Figura 26.

A Figura 26 mostra que a memória residente esteve baixa (valor abaixo de 50%) para todos os casos de alocação de memória com exceção de 80 MB. Por outro lado, a memória reservada manteve-se consistentemente alta, entre 140% e 340%. Assim, somente a primeira alocação (80 MB) é diagnosticada como subprovisionada, com $\overline{m_r} = 52\%$ (confortável) e $\overline{m_c} = 300\%$ (relevante), e somente o último caso (224 MB) é diagnosticado superprovisionado, com $\overline{m_r} = 22\%$ (baixa) e $\overline{m_c} = 139\%$ (irrelevante). As alocações de memória na faixa

Figura 26 – Tempo de resposta (eixo *y* à esquerda) e utilização de memória (eixo *y* à direita) para RUBiS.




Fonte: o autor

96–208 MB são diagnosticadas como adequadas, com $\overline{m_r}$ baixa e $\overline{m_c}$ relevante. Destaca-se que, apesar de apresentar vários pontos, a amplitude de alocação não é grande (112 MB), sendo uma faixa comparável com aquelas dos outros experimentos. O tempo de resposta cai de 55,2 ms com 80 MB para 44,8 ms com 96 MB (uma queda de 18%), mantendo-se em torno de 44,4 ms para os valores superiores a 112 MB. A Figura 26 também mostra que a memória reservada pode ser muito maior que a memória residente, reforçando a ideia de que basear-se exclusivamente em *Committed_AS* para diagnosticar o provisionamento de memória pode ser uma abordagem muito conservadora (Seção 3.3).

## 4.4 DIAGNÓSTICO DE RECURSOS COMBINADOS

Os experimentos realizados na Seção 4.3 para avaliar o diagnóstico do provisionamento dos recursos às máquinas virtuais tomaram como premissa que não haveria influência em um diagnóstico por parte dos demais recursos. Essa abordagem é suficiente para validar o modelo proposto de acordo com cada recurso, mas pode não representar a necessidade de um usuário que adquire recursos de uma nuvem computacional. Em termos de provisionamento em nuvem, é possível

que as capacidades provisionadas variem individualmente para cada recurso, ocasionando interferência de umas sobre as outras. Ou seja, um cliente utilizará o modelo para diagnosticar uma alocação combinada e não simplesmente os recursos isolados.

Diante desse comportamento, o objetivo desta seção é verificar o modelo de diagnóstico de provisionamento considerando os três recursos simultaneamente, permitindo avaliar se o modelo proposto irá conduzir a alocação dos recursos para um estado de provisionamento de bom desempenho com o menor custo possível.

### 4.4.1 Metodologia do experimento

Nos testes de provisionamento de recursos combinados fez-se o uso da ferramenta RUBiS, uma vez que ela não satura nenhum dos três recursos, e não é possível prever qual a menor relação de capacidade que apresente desempenho comparável com o ideal. Isso simula um cliente de nuvem computacional que não conhece as necessidades de recursos de suas aplicações.

Foi utilizada a carga padrão da ferramenta, cujo cliente envia requisições ao servidor durante um período aproximado de 10 minutos. Neste espaço de tempo a taxa de requisições não é constante, e apresenta uma rampa de subida (aumento gradual da taxa) com duração de 3 minutos, um período com taxa constante de 5 minutos e uma rampa de descida com tempo de 2 minutos.

Para que a avaliação do modelo de diagnóstico fosse possível, foram provisionadas diferentes relações de alocação entre os recursos, com as seguintes capacidades, e executada a carga de trabalho da ferramenta sobre cada uma delas:

- Rede: entre 10 e 100 Mbps, com incrementos de 10 Mbps;
- CPU: uma VCPU de capacidade entre 10 e 100%, com incrementos de 10%;
- Memória: entre 200 e 1000 MB, incrementos de 100 MB.

Com estas alocações são encontradas 900 possíveis combinações, o que ressalta a dificuldade de se encontrar uma relação ótima de

alocação. A carga de trabalho foi executada sobre cada uma das possíveis combinações, e as métricas necessárias ao modelo foram capturadas a cada rodada do experimento. Esta abordagem foi aplicada aos dois ambientes de teste disponíveis, *Cenário-1* e *Cenário-2*, e foram três rodadas para cada combinação; com isso, o tempo de resposta teve coeficiente de variação inferior a 5%.

Com base nestes dados, o algoritmo de ajuste de provisionamento (Figura 7) foi utilizado para simular a ação de um cliente de nuvem. O ajuste de provisionamento é feito da seguinte forma:

1. Uma combinação entre todas as disponíveis é escolhida;

2. O modelo de diagnóstico é aplicado sobre os recursos;

3. Com base nos resultados do modelo e na sugestão do algoritmo de ajuste o próximo ponto é avaliado (simulando o ajuste). Neste caso é feito o incremento/decremento de uma unidade de granularidade sobre o recurso ajustado.

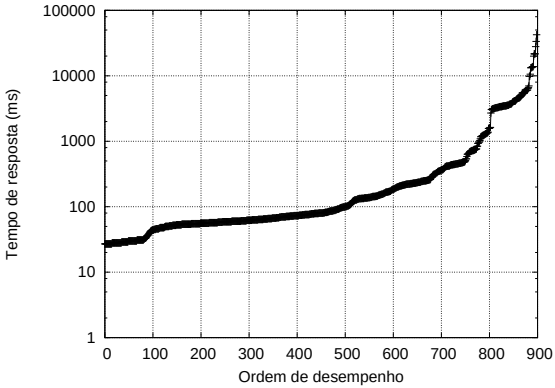
Assim, após algumas iterações, o modelo tende a convergir para uma combinação de alocação. Para avaliar a sensibilidade da convergência à alocação inicial de recursos, foram adotados como pontos de início da simulação foram a menor combinação (10 Mbps, 10% de cap e 200 MB), a maior combinação (100 Mbps, 100% de cap e 1000 MB) e uma combinação intermediária (50 Mbps, 50% de cap e 500 MB).

### 4.4.2 Diagnóstico de provisionamento no *Cenário-1*

O gráfico da Figura 27 apresenta os tempos de resposta em relação aos experimentos realizados. Para gerá-lo, os tempos de resposta foram ordenados de forma crescente e plotados em escala logarítmica no eixo y. Neste gráfico, quanto mais à esquerda no eixo x, menor é o tempo de resposta, e maior tende a ser a alocação combinada. Ao observar o gráfico pode-se perceber que a partir da 500ªcombinação (tempo de resposta igual a 100 ms) os tempos de resposta apresentam

maior taxa de crescimento, enquanto que, para os valores inferiores a este ponto, os ganhos de desempenho são marginais.

Figura 27 – Distribuição dos tempos de resposta em ordem crescente para o *Cenário-1*




Fonte: o autor

Aplicando-se o algoritmo de ajuste percebeu-se que, independentemente do ponto inicial, o modelo conduz para a mesma combinação, com 20 Mbps de rede, 60% de uma VCPU e 200 MB de memória, para o qual o tempo de resposta médio foi de 66 ms. Foram necessárias 12 iterações a partir da menor alocação, 39 iterações iniciando pela maior alocação e 17 iniciando pela combinação intermediária. Sendo assim, decidiu-se por apresentar com maior detalhes os passos realizados a partir da menor alocação. Estes passos são expostos pela Tabela 6, que apresenta as iterações de ajuste, o diagnóstico para cada recurso e o tempo de resposta de cada combinação.

Observa-se na Tabela 6 que, a partir do passo 8, o algoritmo continua a pesquisa por estados não visitados e converge para uma mesma combinação (passos 8, 10 e 12). Este tempo de resposta pode ser considerado um valor de equilíbrio entre custo e desempenho, isto justifica-se pelo fato de que existem alocações com maior capacidade de processador e memória com tempos de resposta maiores ou iguais ao apresentado (por exemplo, uma MV com 20 Mbps, 100% de uma VCPU e 500 MB possui tempo de resposta igual a 96 ms). Além disto,

Tabela 6 – Iterações de alocação para ajuste de provisionamento de recursos no cenário 1, em destaque o recurso ajustado em cada iteração (#).

| # | Rede (Mbps) | CPU (%) | Memória (MB) | Diagnósticos (rede/CPU/mem) | TR (ms) |
|---|---|---|---|---|---|
| 0 | 10 | 10 | 200 | sub/sub/adeq. | 4665 |
| 1 | 20 | 10 | 200 | sub/sub/adeq. | 3293 |
| 2 | 30 | 10 | 200 | super/sub/adeq. | 3385 |
| 3 | 30 | 20 | 200 | super/sub/adeq. | 485 |
| 4 | 30 | 30 | 200 | super/sub/adeq. | 218 |
| 5 | 30 | 40 | 200 | super/sub/adeq. | 88 |
| 6 | 30 | 50 | 200 | super/sub/adeq. | 63 |
| 7 | 30 | 60 | 200 | super/super/super | 80 |
| 8 | 20 | 60 | 200 | super/super/adeq. | 66 |
| 9 | 20 | 50 | 200 | super/sub/adeq. | 86 |
| 10 | 20 | 60 | 200 | super/super/adeq. | 66 |
| 11 | 10 | 60 | 200 | sub/super/adeq. | 145 |
| 12 | 20 | 60 | 200 | super/super/adeq. | 66 |

Fonte: o autor

o valor situa-se abaixo de 100 ms na distribuição apresentada pela Figura 27, na área em que os ganhos de desempenho são marginais frente às melhorias de capacidade alocadas.

Para que o diagnóstico encontre tempos de resposta melhores seria necessário provisionar mais capacidade de rede. Por exemplo, uma MV com 40 Mbps, 60% de uma VCPU e 200 MB apresenta tempo de resposta igual a 62 ms; ao elevar a rede para 80 Mbps (mantendo processador e memória) o tempo de resposta cai para 44 ms. Todavia, entende-se que este ganho não justifica o incremento de recursos, uma vez que, no primeiro caso, dobrou-se a capacidade em relação ao ponto sugerido pelo modelo (20 Mbps) e a melhoria foi de 4 ms, e no segundo caso, o recurso foi quadruplicado e o ganho foi de 22 ms.

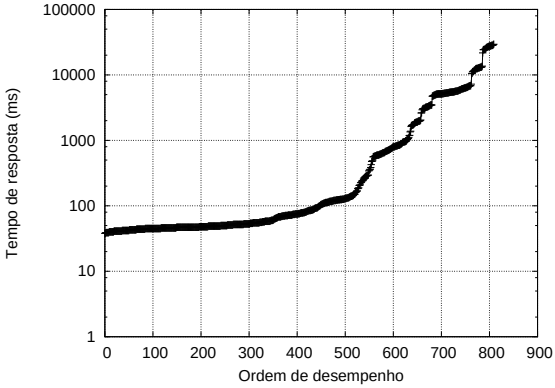
Com base nestes resultados, pode-se dizer que o modelo apresenta um diagnóstico correto sobre o provisionamento de cada um dos recursos, mesmo havendo interferência de um sobre o outro em uma situação de combinação. Portanto, este modelo pode ser utilizado como

base em um algoritmo de ajuste de capacidade.

### 4.4.3 Diagnóstico de provisionamento no *Cenário-2*

O comportamento da aplicação no *cenário-2* é similar ao comportamento observado no *cenário-1*, conforme pode ser observado pelo gráfico da Figura 28, que apresenta a distribuição ordenada dos tempos de resposta.

Figura 28 – Distribuição dos tempos de resposta em ordem crescente para o *Cenário-2*




Fonte: o autor

Os tempos de resposta distribuem-se em uma curva semelhante à da Figura 27, onde o aumento de TR é significativo para os valores superiores a 100 ms e a queda é marginal nos valores abaixo de 100 ms. Tendo em vista esta similaridade de comportamentos entre os cenários, era esperado que o modelo de diagnóstico/alocação convergisse também de forma similar. Isso foi comprovado pelo número idêntico de iterações necessárias e pela combinação de alocação para o qual o modelo convergiu. A Tabela 7 apresenta as iterações de ajuste, o diagnóstico para cada recurso e o tempo de resposta de cada combinação no *cenário-2* partindo da combinação de menor capacidade.

A observação da Tabela 7 permite concluir que a ordem das alocações é a mesma daquela apresentada para o *Cenário-1*, diferindo

Tabela 7 – Iterações de alocação para ajuste de provisionamento de recursos no *cenário-2*, em destaque o recurso ajustado em cada iteração (#).

| # | Rede (Mbps) | CPU (%) | Memória (MB) | Diagnósticos (rede/CPU/mem) | TR (ms) |
|---|---|---|---|---|---|
| 0 | 10 | 10 | 200 | sub/sub/adeq. | 4872 |
| 1 | 20 | 10 | 200 | sub/sub/adeq. | 5781 |
| 2 | 30 | 10 | 200 | super/sub/adeq. | 5127 |
| 3 | 30 | 20 | 200 | super/sub/adeq. | 649 |
| 4 | 30 | 30 | 200 | super/sub/adeq. | 75 |
| 5 | 30 | 40 | 200 | super/sub/adeq. | 59 |
| 6 | 30 | 50 | 200 | super/sub/adeq. | 52 |
| 7 | 30 | 60 | 200 | super/super/super | 50 |
| 8 | 20 | 60 | 200 | super/super/adeq. | 55 |
| 9 | 20 | 50 | 200 | super/sub/adeq. | 57 |
| 10 | 20 | 60 | 200 | super/super/adeq. | 55 |
| 11 | 10 | 60 | 200 | sub/super/adeq. | 124 |
| 12 | 20 | 60 | 200 | super/super/adeq. | 55 |

Fonte: o autor

apenas quanto ao tempo de resposta. Enquanto no *Cenário-1* o TR converge para 66 ms, neste outro ambiente a mesma alocação (20 Mbps, 60 % de VCPU e 200 MB) obteve 55 ms.

Portanto, é possível afirmar que a aplicação RUBiS apresenta comportamento similar mesmo executando em ambientes distintos, com a mesma necessidade de capacidade de recursos. Além disto, os resultados obtidos permitem concluir que o modelo proposto é capaz de diagnosticar corretamente o provisionamento, independente da plataforma de *hardware*, e, com o auxílio de um algoritmo de ajuste, pode conduzir a uma combinação de capacidade de recursos que equilibra o custo e a qualidade de serviço.

## 4.5 CONSIDERAÇÕES SOBRE LIMIARES

Os modelos para diagnosticar os níveis de utilização de recursos propostos neste trabalho baseiam-se na definição de limiares para caracterizar as categorias especificadas: subprovisionado, adequado e superprovisionado. A escolha dos limiares entre as categorias deve equilibrar as características conflitantes de custo e qualidade de serviço. Quanto mais recursos são alocados, a tendência é que melhor seja o desempenho, mas os custos também aumentam. Assim, são os limiares que definem o "tamanho" de cada categoria. Neste trabalho priorizou-se a qualidade de serviço em detrimento do custo quando estes dois comportamentos estão em conflito de diagnóstico. Todavia, os limiares são parametrizáveis e podem ser definidos pelo usuário.

Espera-se que os diagnósticos sugeridos sejam base para o provisionamento de recursos. Assim, o mecanismo de tomada de decisão (seja manual ou automático) deve lidar com algumas situações não controladas. Por exemplo, quando houver sequência intercalada entre diagnósticos de sub- e superprovisionamento, sem indicação de recurso adequado, o mecanismo deveria manter a alocação do recurso com uma capacidade equivalente ao super, ou se possível um meio termo entre os dois, privilegiando a qualidade do serviço. O algoritmo da Figura 7 na Seção 3.4 é uma abordagem simples de uso do diagnóstico para provisionar recursos à MVs. Todavia, seja qual for a abordagem adotada, um usuário do modelo pode refinar os limiares para priorizar custo ou qualidade de serviço.

Desse modo, esta seção objetiva discutir os impactos que os ajustes sobre os limiares podem incorrer sobre cada recurso. A metodologia consiste em mostrar, para cada ajuste de limiar, seja aumentando ou diminuindo o seu valor, se o resultado do diagnóstico irá privilegiar o custo ou a qualidade de serviço.

### 4.5.1 Ajustes sobre o modelo de processador

Uma análise sobre os limiares definidos para diagnóstico de processador (Seção 3.1) mostra que estes irão priorizar a qualidade

do serviço se forem utilizados valores menores que os propostos. Por exemplo, utilizar 70% como limiar superior para $\overline{U_{cpu}}$ (em vez de 80%) fará com que aumente a faixa de valores diagnosticados como subprovisionados, privilegiando o desempenho em detrimento ao custo. O mesmo ocorre para os limiares $\theta$, $\omega$ e *%Steal*.

De forma oposta, aumentar os valores admitidos para estes limiares tende a proporcionar faixas maiores de superprovisionamento, privilegiando o custo em relação ao desempenho. Por exemplo, utilizar 95% para $\overline{U_{cpu}}$ determina que serão considerados subprovisionados os pontos com alta taxa de utilização de processador, diminuindo a probabilidade deste diagnóstico ocorrer.

O impacto destes ajustes pode ser avaliado no diagnóstico realizado com recursos combinados no *Cenário-1* (Seção 4.4.2). Adotar valores 5% maiores em todos os limiares de processador (6% para *%Steal*, 55% e 85% para $\overline{U_{cpu}}$) conduzirá o diagnóstico para uma combinação com 20 Mbps, 50% de VCPU e 200 MB, onde o tempo de resposta é de 86 ms. Ao reduzir os valores dos limiares em 5% (0% para *%Steal*, 45% e 75% para $\overline{U_{cpu}}$) a combinação encontrada será de 20 Mbps, 100% de VCPU e 200 MB, com TR de 65 ms.

Percebe-se que, no caso deste cenário, aumentar os valores dos limiares de processador degrada o desempenho a um favorecimento pequeno de custo, economizando 10% da capacidade de CPU e degradando o desempenho em 20 ms. Reduzir os limiares prejudica sensivelmente o custo a um ganho mínimo de desempenho, sendo necessário o dobro de capacidade de processador para um ganho de 1 ms.

### 4.5.2 Ajustes sobre o modelo de rede

O diagnóstico proposto para o recurso de rede leva em consideração dois limiares, um sobre o coeficiente de variação (CV) e outro sobre o número de pacotes na fila da interface de rede (*backlog*). O enfileiramento é utilizado para diagnosticar o subprovisionamento, e presume-se que qualquer ocorrência é um sinal de necessidade de recurso. Nos casos onde não há enfileiramento, o coeficiente de variação é utilizado para diferenciar as situações de superprovisionamento

e adequação. Quando o recurso está superprovisionado o consumo de largura de banda não é constante ($CV_{lb} > 5\%$), quando o provisionamento está adequado o consumo é constante ($CV_{lb} \leq 5\%$).

O ajuste sobre estes limiares influencia o diagnóstico da seguinte forma: admitir maiores valores para o CV (por exemplo usar 10% em vez de 5%) aumenta a probabilidade do provisionamento ser diagnosticado como adequado, uma vez que admite-se maior oscilação do consumo de largura de banda. Isto implica em melhorar a qualidade de serviço em detrimento ao custo. De forma oposta, considerar um valor menor para o CV (por exemplo, usar 1% em vez de 5%) reduz a probabilidade de um recurso ser diagnosticado como adequado. Nestes casos seria necessário um consumo quase constante de largura de banda para caracterizar a adequação, resultando em um diagnóstico direcionado à redução de custos em detrimento ao desempenho.

Os impactos destes ajustes podem ser observados nos resultados do diagnóstico sobre o recurso de rede (Seção 4.3.3). Nos testes com requisições a arquivos de 4 MB a capacidade diagnosticada como adequada foi de 60 Mbps, uma vez que não havia enfileiramento e o CV era inferior a 5%. A partir de 70 Mbps o diagnóstico era de superprovisionamento. Caso fosse utilizado um limiar de CV igual a 30%, a alocação de 70 Mbps seria incluída na faixa de diagnóstico adequado. Neste caso, se o usuário iniciasse o ajuste de provisionamento a partir de uma maior alocação (por exemplo 100 Mbps) ele seria conduzido ao ponto de 70 Mbps, com maior custo e desempenho ligeiramente superior. Utilizando-se um limiar de CV igual a 1%, faria com que a capacidade de 60 Mbps deixasse de ser considerada adequada, conduzindo o diagnóstico para um estado de alocação entre 50 Mbps e 60 Mbps, o que degradaria o desempenho em prol da redução de custos.

Nos resultados obtidos nos experimentos com RUBiS e recursos combinados (Seção 4.4) não foram identificados pontos com diagnóstico "adequado" em rede. Isto se deve à inconstância da carga de trabalho da ferramenta (rampas de subida e descida). Contudo, mantém-se a ideia de que o recurso deve ser diagnosticado como adequado quando o consumo de largura de banda for constante e próximo à capacidade provisionada, de outro modo, ou ele será diagnosticado como subprovisionado (com enfileiramento) ou ele será diagnosticado

como superprovisionado (sem enfileiramento e com variação de consumo).

Por fim, destaca-se que o limiar de enfileiramento não deve ser ajustado, pois entende-se que qualquer ocorrência de enfileiramento na interface de rede é um indicativo de saturação do meio de transmissão.

### 4.5.3 Ajustes sobre o modelo de memória

Os valores adotados para os limiares de memória – $\alpha = 70\%$, $\beta = 50\%$, e $\gamma = 150\%$ – mostraram-se apropriados para diagnosticar o provisionamento deste recurso nos cenários avaliados. Entretanto, para ajustar esses limiares é preciso entender como eles podem afetar o diagnóstico. Primeiro, deve-se ter em mente que a memória residente nunca poderá alcançar o total de memória disponível (seja porque o gerenciador de memória do SO irá forçar a liberação de memória quando a utilização estiver alta, ou porque operando com alta utilização o sistema não conseguirá absorver flutuações de carga, sem copiar páginas para o disco). Assim, os limiares $\alpha$ e $\beta$ definem margens de segurança para a quantidade de memória não utilizada: com margens de segurança maiores tem-se grande chance do desempenho ser aceitável. Sendo assim, valores baixos para $\alpha$ e $\beta$ resultam em margens maiores, e valores altos resultam em margens menores. Uma explicação similar se aplica ao limiar $\gamma$, que pondera o risco da porção de memória reservada que não é residente ser utilizada (ou seja, se tornar memória residente): diminuir o valor $\gamma$ aumenta a margem de segurança e vice-versa.

A relação com o custo tende a ser diretamente proporcional a essas margens de segurança (maiores margens implicam em aumento de custos, e pequenas margens implicam em menores custos), uma vez que, conforme cresce a margem de segurança, mais memória é necessária para passar da faixa de subprovisionamento para a faixa de provisionamento adequado.

Ajustar os limiares de memória não traz grande impacto sobre os experimentos do *Cenário-1*. Isso se deve à pouca necessidade de memória da aplicação na carga de trabalho testada. Percebeu-se que o

resultado mais significativo seria utilizar $\gamma = 100\%$, aumentando o intervalo de capacidade de memória diagnosticada como adequada. Isto faz com que o algoritmo, quando iniciado pelo ponto de maior capacidade, conduza à combinação com 20 Mbps, 60% de VCPU e 300 MB de memória, com tempo de resposta de 66 ms, ou seja, aumento de custo sem melhoria significativa de desempenho.

Contudo, em aplicações mais influenciáveis pela capacidade de memória o ajuste dos limiares pode ter um maior impacto, como é o caso da aplicação DaCapo H2, que foi avaliada na Seção 4.3.4. Neste caso, aumentar todos os limiares em 5% ($\alpha = 75\%$, $\beta = 55\%$ e $\gamma = 155\%$) faz com que a faixa de provisionamento adequado seja entre 640 MB (TR=16,8 s) e 768 MB (TR=9,3 s) em vez de ser entre 704 MB (TR=9,6 s) e 832 MB (TR=9,3 s). Isto claramente prejudica o desempenho em benefício do custo. Ainda em H2, reduzir todos os limiares em 5% ($\alpha = 65\%$, $\beta = 45\%$ e $\gamma = 145\%$) adiciona o ponto com 896 MB (TR=9,3 s) à faixa de adequado, o que implica maior custo se o ajuste de provisionamento iniciar pela maior alocação.

## 4.6 CONSIDERAÇÕES DO CAPÍTULO

Este capítulo apresentou uma avaliação experimental sobre o modelo de diagnóstico proposto no Capítulo 3. Esta avaliação foi realizada para verificar o comportamento do modelo em diferentes cenários de teste.

Primeiramente verificou-se o comportamento do modelo sob a perspectiva de cada recurso isoladamente, ou seja, sem a interferência dos demais sobre o diagnóstico. Para cada recurso variou-se a capacidade provisionada e executou-se o modelo de diagnóstico, e o resultado do modelo foi comparado com os tempos de resposta das aplicações avaliadas. Deste modo, foi possível constatar que o comportamento observado nas aplicações é condizente com o resultado apresentado pelo modelo.

Depois de constatada a eficácia do modelo no diagnóstico isolado dos recursos, buscou-se avaliar o comportamento do modelo quando a capacidade de um recurso pode interferir sobre os demais. Assim,

fez-se a avaliação experimental sobre cada uma das combinações de recursos, identificando a situação do provisionamento de cada recurso. Para constatar a eficácia do mesmo, decidiu-se por realizar ajustes sobre o provisionamento seguindo o algoritmo proposto na Seção 3.4. O tempo de resposta da combinação resultante foi comparado com os valores de todas as possíveis combinações de recursos, e constatou-se que o modelo de diagnóstico em conjunto com o mecanismo de ajuste conduz para uma combinação com tempo de resposta próximo ao ideal. Portanto, o modelo pode ser aplicado e seu resultado é uma combinação de alocação que equilibra custo e desempenho.

# 5 CONCLUSÃO

Em nuvens IaaS os provedores vendem recursos de infraestrutura como serviços. Nestes ambientes, os clientes adquirem capacidade computacional de acordo com sua necessidade. A principal vantagem é a possibilidade do cliente ajustar os recursos provisionados conforme a variação da demanda de seus serviços, enquanto o provedor minimiza os custos com o compartilhamento da infraestrutura por entre múltiplos clientes. Normalmente os recursos computacionais são oferecidos através de instâncias de máquinas virtuais (processador, memória e disco) e largura de banda de rede.

A definição da capacidade fornecida a um cliente é expressa por acordos de nível de serviço (SLAs – *Service Level Agreements*), que definem métricas (SLIs – *Service Level Indicators*) e respectivos valores-objetivo (SLOs – *Service Level Objectives*) de modo que o desempenho dos recursos alocados seja considerado satisfatório. Neste modelo, é implícito que o cliente saiba identificar as suas necessidades, e descreve-las em SLIs e SLOs. Entretanto, a definição dos valores objetivos não é uma atividade trivial, uma vez que nem sempre se tem pleno conhecimento das necessidades das aplicações que irão executar na nuvem. Sendo assim, os clientes têm grande chance de sub- ou superdimensionar seus objetivos e seus recursos.

Quando os recursos estiverem subdimensionados o desempenho das aplicações ficará aquém do esperado. Um usuário de um serviço executando em nuvem pode perceber que os recursos estão subdimensionados através de métricas de aplicação (por exemplo, tempo de resposta), mas a definição de qual dos recursos que influencia implica em tal comportamento não é óbvia. Em caso de superdimensionamento a consequência é o desperdício financeiro, pois a capacidade provisionada supera a necessidade das aplicações, ou seja, com um custo menor um desempenho similar poderia ser obtido. Diferentemente do subdimensionamento, um usuário dificilmente poderá identificar o superdimensionamento com métricas de aplicação.

Nesse contexto, o objetivo deste trabalho foi apresentar um modelo para diagnóstico do provisionamento de processador, rede e

memória para máquinas virtuais. Com este modelo um cliente de nuvem IaaS é capaz de identificar a situação de cada um dos recursos quanto ao provisionamento, classificando-os em subprovisionado, adequado e superprovisionado. Com base nos resultados apresentados pelo modelo um cliente pode adequar as suas alocações de recursos ou seus acordos de nível de serviço (SLAs/SLOs). Além disto, as métricas expostas podem ser utilizadas em mecanismos de ajuste dinâmico de capacidade.

A abordagem proposta se baseia na análise de métricas disponíveis nas máquinas virtuais para determinar se esses recursos estão adequados, sub- ou superprovisionados diante da demanda existente. A avaliação de processador leva em conta a média e a variação da utilização do recurso, sendo a utilização expressa por duas métricas, o *%Steal* e a utilização total. A avaliação de rede se baseia na variação do consumo de largura de banda e no enfileiramento observado nas interfaces virtuais. A avaliação de memória leva em consideração duas métricas, a memória residente e a memória reservada. Resultados experimentais com Linux e Xen demonstram que a abordagem proposta é capaz de diagnosticar corretamente o provisionamento em diferentes cenários de carga.

Destaca-se que este trabalho de pesquisa proporcionou alguns resultados importantes em termos de publicação, os modelos de diagnóstico para os recursos processador e rede foram publicados no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) na edição de 2013 (PFITSCHER; PILLON; OBELHEIRO, 2013b), o modelo para diagnóstico do provisionamento de memória foi publicado no Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC), na trilha de Sistemas Operacionais (WSO), na edição de 2013 (PFITSCHER; PILLON; OBELHEIRO, 2013a), e por este ter sido escolhido um dos melhores artigos da trilha, uma expansão do artigo foi aceita para publicação em revista e pode ser encontrado em (PFITSCHER; PILLON; OBELHEIRO, 2014).

Alguns trabalhos futuros podem ser elencados para continuidade desta pesquisa. A primeira proposta é a implementação de um provisionador automático de recursos, a ideia é que esta aplicação ajuste dinamicamente a capacidade dos recursos de MVs com base

nos resultados do modelo. Em uma segunda proposta, sugere-se testar este modelo em ambientes de produção (por exemplo o ambiente utilizado na caracterização realizada em (BIRKE et al., 2013)), utilizando-se cargas de trabalhos que não sejam provenientes de ferramentas de *benchmark*, o que proporcionaria dados ainda mais consistentes para o modelo. Por fim, uma terceira proposta seria avaliar o comportamento do modelo sob a perspectiva de outros monitores de máquinas virtuais (por exemplo VMware, KVM[1], HyperV[2]) e sistemas operacionais convidados (por exemplo Windows e Solaris).

[1] http://www.linux-kvm.org/

[2] https://www.microsoft.com/pt-br/server-cloud/

# REFERÊNCIAS

ACETO, G. et al. Cloud monitoring: a survey. **Computer Networks**, Elsevier, v. 57, n. 9, p. 2093–2115, 2013.

AMAZON. **Amazon EC2 Website**. 2014. Disponível em: <http://aws.amazon.com/pt/ec2/>.

APPARAO, P.; MAKINENI, S.; NEWELL, D. Characterization of network processing overheads in Xen. In: **Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing**. Washington, DC, USA: IEEE Computer Society, 2006. (VTDC '06), p. 2–. ISBN 0-7695-2873-1. Disponível em: <http://dx.doi.org/10.1109/VTDC.2006.3>.

ARMBRUST, M. et al. A view of cloud computing. **Commun. ACM**, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>.

ASSUNÇÃO, M. D. de; COSTANZO, A. di; BUYYA, R. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: **Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing**. New York, NY, USA: ACM, 2009. (HPDC '09), p. 141–150. ISBN 978-1-60558-587-1. Disponível em: <http://doi.acm.org/10.1145/1551609.1551635>.

BAILEY, D. H. et al. The NAS parallel benchmarks summary and preliminary results. In: IEEE. **Proceedings of the 1991 ACM/IEEE Conference on Supercomputing**. [S.l.], 1991. p. 158–165.

BARHAM, P. et al. Xen and the art of virtualization. In: **Proceedings of the 19th ACM Symposium on Operating Systems Principles**. New York, NY, USA: ACM, 2003. (SOSP '03), p. 164–177. ISBN 1-58113-757-5. Disponível em: <http://doi.acm.org/10.1145/945445.945462>.

BARUCHI, A.; MIDORIKAWA, E. T. Elasticidade de memória em máquinas virtuais utilizando média móvel exponencial. **Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC)**, 2010.

BELOGLAZOV, A.; BUYYA, R. Energy efficient resource management in virtualized cloud data centers. In: **Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing**. Washington, DC, USA: IEEE Computer Society, 2010. (CCGRID '10), p. 826–831. ISBN 978-0-7695-4039-9. Disponível em: <http://dx.doi.org/10.1109/CCGRID.2010.46>.

BIRKE, R.; CHEN, L. Y.; SMIRNI, E. Data centers in the cloud: A large scale performance study. In: IEEE. **2012 IEEE 5th International Conference on Cloud Computing (CLOUD)**. [S.l.], 2012. p. 336–343.

BIRKE, R. et al. State-of-the-practice in data center virtualization: Toward a better understanding of VM usage. In: IEEE. **2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)**. [S.l.], 2013. p. 1–12.

BLACKBURN, S. M. et al. The DaCapo benchmarks: Java benchmarking development and analysis. **ACM SIGPLAN Notices**, v. 41, n. 10, p. 169–190, 2006.

BUTLER, B. Gartner's IaaS magic quadrant: A who's who of cloud market. **Network World**, 2012. `http://www.networkworld.com/news/2012/110712-gartner-magic-quadrant-264058.html`. Acessado em 17/fev/2014.

BUYYA, R.; GARG, S.; CALHEIROS, R. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In: **2011 International Conference on Cloud and Service Computing (CSC)**. [S.l.: s.n.], 2011. p. 1–10.

CECCHET, E. et al. Performance comparison of middleware architectures for generating dynamic web content. In: SPRINGER. **Middleware 2003**. [S.l.], 2003. p. 242–261.

CHERKASOVA, L.; GUPTA, D.; VAHDAT, A. Comparison of the three CPU schedulers in Xen. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 35, n. 2, p. 42–51, set. 2007. ISSN 0163-5999. Disponível em: <http://doi.acm.org/10.1145/1330555.1330556>.

DM-IOBAND. 2014. Disponível em: <http://sourceforge.net/apps/trac/ioband>.

DU, J.; SEHRAWAT, N.; ZWAENEPOEL, W. Performance profiling of virtual machines. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 46, n. 7, p. 3–14, mar. 2011. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/2007477.1952686>.

FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. In: IEEE. **Grid Computing Environments Workshop, 2008. GCE'08**. [S.l.], 2008. p. 1–10.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 4a. ed. [S.l.]: Atlas, 2002.

GORMAN, M. **Understanding the Linux Virtual Memory Manager**. Upper Saddle River, NJ: Prentice-Hall, 2004.

HALDER, K.; BELLUR, U.; KULKARNI, P. Risk aware provisioning and resource aggregation based consolidation of virtual machines. In: **2012 IEEE 5th International Conference on Cloud Computing (CLOUD)**. [S.l.: s.n.], 2012. p. 598 –605. ISSN 2159-6182.

HEO, J. et al. Memory overbooking and dynamic control of Xen virtual machines in consolidated environments. In: **IFIP/IEEE International Symposium on Integrated Network Management**. [S.l.: s.n.], 2009. p. 630 –637.

HOCH, D. **Linux System and Performance Monitoring**. Redwood City, CA, USA: Darren Hoch - StrongMail Systems, 2010.

HOEFER, C.; KARAGIANNIS, G. Taxonomy of cloud computing services. In: IEEE. **GLOBECOM Workshops (GC Wkshps), 2010 IEEE**. [S.l.], 2010. p. 1345–1350.

JACOBSON, V. Congestion avoidance and control. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 1988. v. 18, n. 4, p. 314–329.

KUNDU, S. et al. Modeling virtualized applications using machine learning techniques. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 47, n. 7, p. 3–14, mar. 2012. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/2365864.2151028>.

LAGAR-CAVILLA, H. A. et al. Snowflock: rapid virtual machine cloning for cloud computing. In: **Proceedings of the 4th ACM European conference on Computer systems**. New York, NY, USA: ACM, 2009. (EuroSys '09), p. 1–12. ISBN 978-1-60558-482-9. Disponível em: <http://doi.acm.org/10.1145/1519065.1519067>.

LO, J. VMware and CPU virtualization technology. **World Wide Web electronic publication**, 2005.

MARCONI, M. A.; LAKATOS, E. M. **Fundamentos da Metodologia Científica**. 7th. ed. [S.l.]: Atlas, 2010.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. 26. ed. [S.l.], jul. 2009. Disponível em: <http://www.csrc.nist.gov/groups/SNS/cloud-computing/>.

MENASCE, D. A.; ALMEIDA, V. A. **Capacity Planning for Web Services: metrics, models, and methods**. [S.l.]: Prentice Hall Upper Saddle River, 2002.

MUKHERJEE, J. et al. Resource contention detection and management for consolidated workloads. In: **Proc. IFIP/IEEE Intl. Symp. on Integrated Network Management (IM)**. Ghent, Belgium: [s.n.], 2013.

NIKOLAEV, R.; BACK, G. Perfctr-Xen: a framework for performance counter virtualization. In: **Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments**. New York, NY, USA: ACM, 2011. (VEE '11), p. 15–26. ISBN 978-1-4503-0687-4. Disponível em: <http://doi.acm.org/10.1145/1952682.1952687>.

PADALA, P. et al. Adaptive control of virtualized resources in utility computing environments. In: **Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007**. New York, NY, USA: ACM, 2007. (EuroSys '07), p. 289–302. ISBN 978-1-59593-636-3. Disponível em: <http://doi.acm.org/10.1145/1272996.1273026>.

PFITSCHER, R. J.; PILLON, M. A.; OBELHEIRO, R. R. Diagnosing memory provisioning in IaaS clouds. In: . [S.l.: s.n.], 2013.

PFITSCHER, R. J.; PILLON, M. A.; OBELHEIRO, R. R. Diagnóstico do provisionamento de recursos para máquinas virtuais em nuvens IaaS. In: . [S.l.: s.n.], 2013.

PFITSCHER, R. J.; PILLON, M. A.; OBELHEIRO, R. R. Customer-oriented diagnosis of memory provisioning for IaaS clouds. **ACM SIGOPS Operating Systems Review**, 2014.

PRAS, A. et al. Dimensioning network links: a new look at equivalent bandwidth. **IEEE Network**, v. 23, n. 2, p. 5 –10, march 2009. ISSN 0890-8044.

RAO, J. et al. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In: **Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on**. [S.l.: s.n.], 2011. p. 45–54. ISSN 1526-7539.

RIEL, R. van. Measuring resource demand on Linux. In: **Linux Symposium, 2006**. [S.l.: s.n.], 2006.

SAUVÉ, J. et al. SLA design from a business perspective. **Ambient Networks**, Springer, p. 72–83, 2005.

SUDEVALAYAM, S.; KULKARNI, P. Affinity-aware modeling of CPU usage for provisioning virtualized applications. In: **2011 IEEE International Conference on Cloud Computing (CLOUD)**. [S.l.: s.n.], 2011. p. 139 –146. ISSN 2159-6182.

SULEIMAN, B. et al. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. **Journal of Internet Services and Applications**, Springer London, v. 3, p. 173–193, 2012. ISSN 1867-4828. 10.1007/s13174-011-0050-y. Disponível em: <http://dx.doi.org/10.1007/s13174-011-0050-y>.

TANENBAUM, A. S. **Modern Operating Systems**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007. ISBN 9780136006633.

VAN, H. N.; TRAN, F.; MENAUD, J.-M. SLA-aware virtual resource management for cloud infrastructures. In: **CIT'09. Ninth IEEE International Conference on Computer and Information Technology, 2009.** [S.l.: s.n.], 2009. v. 1, p. 357–362.

VASIĆ, N. et al. Dejavu: accelerating resource allocation in virtualized environments. **SIGARCH Comput. Archit. News**, ACM, New York, NY, USA, v. 40, n. 1, p. 423–436, mar. 2012. ISSN 0163-5964. Disponível em: <http://doi.acm.org/10.1145/2189750.2151021>.

VIEIRA, J. G. S. **Metodologia de Pesquisa Científica na Prática**. [S.l.]: Editora Fael Ltda., 2010.

WALDSPURGER, C. A. Memory resource management in VMware ESX server. **ACM SIGOPS Operating Systems Review**, ACM, v. 36, n. SI, p. 181–194, 2002.

WANG, L. et al. Autonomic resource management for virtualized database hosting systems. Citeseer, 2009.

WAZLAWICK, R. S. Uma Reflexão sobre a Pesquisa em Ciência da Computação à Luz da Classificação das Ciências e do Método Científico. **Revista de Sistemas de Informação da FSMA**, v. 1, n. 6, p. 3–10, Julho 2010. ISSN 1983-5604.

WOOD, T. et al. Black-box and gray-box strategies for virtual machine migration. In: **Proceedings of the 4th USENIX conference**

**on Networked systems design & implementation**. [S.l.: s.n.], 2007. p. 17–17.

WOOD, T. et al. Sandpiper: Black-box and gray-box resource management for virtual machines. **Computer Networks**, Elsevier, v. 53, n. 17, p. 2923–2938, 2009.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, Springer-Verlag, v. 1, p. 7–18, 2010. ISSN 1867-4828. Disponível em: <http://dx.doi.org/10.1007/s13174-010-0007-6>.

ZHAO, W.; WANG, Z.; LUO, Y. Dynamic memory balancing for virtual machines. **ACM SIGOPS Operating Systems Review**, ACM, v. 43, n. 3, p. 37–47, 2009.