Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pós-Graduação em Engenharia de Teleinformática

Thiago Teixeira Sá

Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado

Fortaleza – Ceará
Agosto 2013

Autor:
Thiago Teixeira Sá

Orientador:
Prof. Danielo Gonçalves Gomes, Dr.

Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado

Dissertação submetida à coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como requisito parcial para a obtenção do grau de Mestre em Engenharia de Teleinformática. Área de concentração: Sinais e Sistemas.

Fortaleza – Ceará
Agosto 2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós-Graduação em Engenharia - BPGE

S115e Sá, Thiago Teixeira
Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado / Thiago Teixeira Sá. – 2013
72 f. : il. color., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Departamento de Engenharia de Teleinformática, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2013.
Concentração: Sinais e Sistemas.
Orientação: Prof. Dr. Danielo Gonçalves Gomes.

1. Teleinformática. 2. Sistemas operacionais Distribuídos (Computadores). I. Título.

CDD 621.38

THIAGO TEIXEIRA SÁ

UMA ESTRATÉGIA DE GERENCIAMENTO DE INFRAESTRUTURA DE DATACENTERS BASEADA EM TÉCNICAS DE MONITORAMENTO DISTRIBUÍDO E CONTROLE CENTRALIZADO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Teleinformática.
Área de concentração: Sinais e Sistemas.

Aprovada em: 29/08/2013.

BANCA EXAMINADORA

Prof. Dr. Daniel Gonçalves Gomes (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. José Marques Soares
Universidade Federal do Ceará - UFC

Prof. Dr. Cesar Augusto Fonticielha De Rose
Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS

Resumo

A Computação em Nuvem desponta como um paradigma de utilização de recursos computacionais segundo o qual infraestrutura de hardware, software e plataformas para o desenvolvimento de novas aplicações são oferecidas como serviços disponíveis em escala global. Tais serviços são disponibilizados por meio de *datacenters* de larga escala onde é recorrente o emprego de tecnologias de virtualização para o uso compartilhado de recursos. Neste contexto, a gerência eficiente da infraestrutura do *datacenter* pode levar a reduções significativas de seus custos de operação. Este trabalho apresenta uma estratégia de gerenciamento de infraestrutura de *datacenters* virtualizados que aplica técnicas de monitoramento distribuído combinadas a ações centralizadas de controle. Tal estratégia busca diminuir os efeitos de sobrecarga observados em modelos tradicionais de gerência baseados em um nó controlador que acumula responsabilidades de controle e monitoramento. Por conseguinte, busca-se elevar o poder de escalabilidade da infraestrutura e melhorar sua eficiência energética sem comprometer a Qualidade de Serviço (QoS) oferecida ao usuário final. A análise de desempenho da estratégia proposta é realizada através de múltiplos experimentos de simulação realizados com ferramentas voltadas especificamente para a modelagem de nuvens computacionais e com suporte à representação do consumo energético da infraestrutura simulada.

Palavras-chave: Computação em Nuvem, Sistemas Distribuídos, Gerência de *datacenters* virtualizados.

Abstract

Cloud Computing emerges as a paradigm of computing resources utilization where hardware infrastructure, software and development platforms for new applications are provided as services available worldwide. Such services are offered using large scale datacenters where virtualization technologies are frequently used so that a multitenancy-enabled architecture can be deployed. In this context, the efficient management of the datacenter infrastructure can lead to significant reduction of its operational costs. This work presents a management strategy for virtualized datacenters infrastructures that applies both distributed monitoring techniques and centralized control actions. Such strategy aims to attenuate overloading problems observed in traditional management models based on a controller that is responsible for monitoring and controlling the infrastructure. Thus, it intends to increase the datacenter's scalability and improve its power efficiency without compromising the Quality of Service offered to the end user. The strategy's performance evaluation is made through multiple simulation experiments which use tools aimed specifically at modeling cloud infrastructures with support to power consumption representation.

Keywords: Cloud Computing, Distributed Systems, Management of virtualized datacenters.

Este trabalho é dedicado integralmente à minha família, a quem tudo devo.

Agradecimentos

Esta dissertação de mestrado é um resultado direto de muitos meses de esforço contínuo, inúmeras noites em claro e incontáveis xícaras de café. Tê-la finalizada evoca um sentimento de realização único e traz de imediato à memória todas as pessoas responsáveis pelo sucesso desse trabalho. Em primeiro lugar, quero agradecer a confiança em mim depositada pelo meu professor e orientador Danielo Gonçalves Gomes. Graças ao seu apoio e sua sensibilidade ao perceber minhas necessidades em períodos difíceis dessa caminhada, consegui conciliar trabalho e vida acadêmica até a conclusão do curso. Agradeço ainda ao professor José Marques Soares, que ajudou a viabilizar minha entrada no mestrado ao ser meu orientador durante alguns meses e, ao fim, aceitar o convite para participar da banca de minha defesa. Agradeço também ao caro amigo Rodrigo Calheiros, que desempenhou um papel imprescindível durante toda a produção do texto final, e cujas sugestões provenientes de suas revisões ajudaram a enriquecer sobremaneira esta dissertação. Ademais, agradeço ao professor César Augusto De Rose pela participação na banca examinadora e pelas críticas construtivas aplicadas ao meu trabalho.

Contudo, nada disso seria possível sem o suporte oferecido por minha família e o apoio incessante e incondicional de minha então namorada e agora noiva Ingrid. Sua compreensão durante todos os dias em que tive que me dedicar integralmente ao mestrado sempre me serviu de combustível para não desistir e sempre continuar trabalhando. Portanto, embora este documento me tenha em sua capa como único autor, muitos coautores colaboraram para sua existência oferecendo-me confiança, compreensão, apoio e amor. Deixo aqui meus sinceros agradecimentos a todos.

Simplicity and elegance are unpopular because they require hard work and discipline to achieve and education to be appreciated.

Edsger Wybe Dijkstra

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

SLA *Servive Level Agreement* (Acordo de Nível de Serviço)

QoS *Quality of Service* (Qualidade de Serviço)

IaaS *Infrastructure as a Service* (Infraestrutura como Serviço)

PaaS *Platform as a Service* (Plataforma como Serviço)

SaaS *Software as a Service* (*Software* como Serviço)

ERP *Enterprise Resource Planning* (Planejamento de Recursos Empresariais)

DCIM *Datacenter Infrastructure Management* (Gerenciamento de Infraestrutura de *Datacenter*)

ORM *Object-relational Mapping* (Mapeamento de Objetos Relacionais)

SST *Single Static Threshold* (Limiar Estático Único)

DST *Double Static Threshold* (Limiar Estático Duplo)

MAD *Median Absolute Deviation* (Desvio Absoluto de Mediana)

MMT *Minimum Migration Time* (Tempo Mínimo de Migração)

LR *Local Regression* (Regressão Local)

PUE *Power Usage Effectiveness* (Eficácia do Uso de Potência)

VM *Virtual Machine* (Máquina Virtual)

TEC Tempo de Estabilização de Consumo

API *Application Programming Interface* (Interface de Programação de Aplicativos)

POSIX *Portable Operating System Interface* (Interface Portável entre Sistemas Operacionais)

Lista de Símbolos

$\bar{\gamma}_j$	Limite máximo de utilização de memória para uma máquina virtual v_j
$\bar{\omega}_j$	Limite máximo de utilização de processamento para uma máquina virtual v_j
$\bar{c}_i$	Capacidade de processamento do nó computacional n_i
$\bar{m}_i$	Quantidade de memória disponível no nó computacional n_i
μ	Limite mínimo de utilização de recursos em um nó computacional
$\omega_j(t)$	Demanda por processamento de uma máquina virtual v_j no instamte t
τ	Limite máximo de utilização de recursos em um nó computacional
$\theta(t)$	Quantidade de máquinas virtuais pertencentes a V no instante t
A	Número de nós computacionais que não alocam nenhuma máquina virtual
C	Capacidade total de processamento do *datacenter*
$c_i(t)$	Consumo de processamento do nó computacional n_i no instante t
M	Quantidade total de memória do *datacenter*
$m_i(t)$	Consumo de memória do nó computacional n_i no instante t
N	Conjunto de nós computacionais do *datacenter*
n	Nó computacional percentence a N
p	Quantidade de nós computacionais pertencentes a N
V	Conjunto de máquinas virtuais alocadas pelo *datacenter*
v	Máquina virtual pertencente a V

Capítulo 1

Introdução

Neste capítulo, a Seção 1.1 situa o trabalho no contexto da Computação em Nuvem, introduzindo uma terminologia básica e discutindo suas características fundamentais. A Seção 1.2 identifica os problemas que motivaram o desenvolvimento da solução proposta. Na Seção 1.3, são apresentadas hipóteses sobre os efeitos da arquitetura de gerenciamento em um *datacenter* virtualizado. A Seção 1.4 lista os objetivos gerais e específicos do trabalho, seguidos da descrição de todas as etapas da metodologia empregada em sua execução, apresentada na Seção 1.5. O capítulo é concluído com uma visão geral da estrutura da dissertação na Seção 1.6.

1.1 Contextualização

A Computação em Nuvem propõe a integração de modelos tecnológicos para o provimento de infraestrutura de hardware, plataformas de desenvolvimento e aplicações na forma de serviços disponíveis remotamente e em escala global. Neste novo paradigma de utilização de recursos computacionais, clientes abrem mão da administração de uma infraestrutura própria e dispõem de serviços oferecidos por terceiros, delegando responsabilidades e assumindo custos proporcionais à quantidade de recursos que utilizam. Os ambientes computacionais em nuvem são caracteristicamente compostos por infraestruturas distribuídas e com recursos heterogêneos e virtualizados, além de servirem concomitantemente a uma grande diversidade de clientes cujo Acordo de Nível de Serviço (SLA) pode exigir níveis de Qualidade de Serviço (QoS) distintos. Ademais, os *datacenters* que compõem uma nuvem de recursos devem suportar um número virtualmente ilimitado de aplicações,

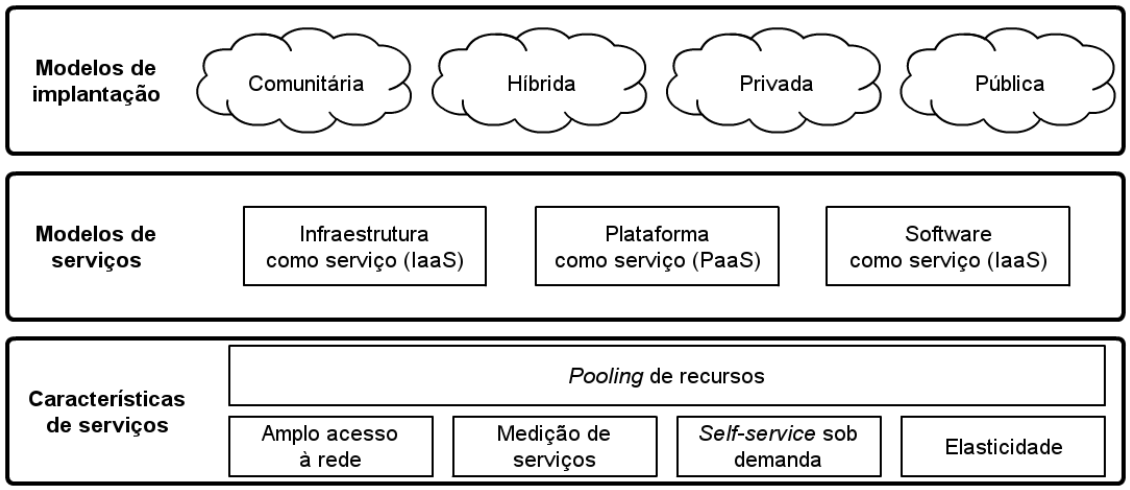


Figura 1.1: Representação da definição oficial do NIST para modelos da Computação em Nuvem (adaptada de (SOSINSKY, 2011)).

o que os obriga a lidar com demandas variantes por processamento, armazenamento de dados e utilização de banda.

O NIST (*U.S. National Institute of Standarts and Technology*) dispõe de um conjunto de definições que qualifica a Computação em Nuvem de acordo com modelos distintos de implantação e serviço. Tal divisão, assim como algumas das principais características dos serviços em nuvem, é mostrada na Figura 1.1.

Os modelos de implantação dizem respeito ao tipo de gerenciamento e nível de acesso a serviços presente em uma nuvem computacional. Enquanto as nuvens públicas estão acessíveis ao público geral e são gerenciadas por um provedor de serviços, as nuvens privadas são voltadas para o uso exclusivo de uma organização, sendo normalmente gerenciada pela mesma. Nuvens comunitárias são infraestruturas criadas para servir a interesses comuns de diferentes grupos (e.g. nuvens acadêmicas que dão suporte a pesquisas de múltiplas universidades). O modelo de implantação híbrido consiste em uma combinação de múltiplas nuvens (públicas, privadas ou comunitárias) cujas características são mantidas, mas que se apresentam como uma única grande infraestrutura. Os modelos de serviço especificam que tipo de recurso será oferecido ao usuário final. Na Infraestrutura como Serviço (IaaS), são oferecidos capacidade de processamento, armazenamento de dados, estrutura de rede e uma vasta série de recursos básicos de *hardware* de forma virtualizada seguindo características de *self-service* sob demanda, elasticidade e multi-arrendamento (obtido por meio de *pooling* de recursos). Provedores de

IaaS utilizam técnicas de medição de serviços para aplicar regras de taxação de acordo com o nível de utilização de sua infraestrutura, em uma modalidade de cobrança comumente denominada de *pay-as-you-go*. O modelo de Plataforma como Serviço (PaaS) tem como público alvo programadores e engenheiros de *software*, oferecendo-os ambientes de desenvolvimento autoconfiguráveis e livrando-os de preocupações com o gerenciamento de toda a pilha de aplicações e *hardware* exigida por esses ambientes. Por fim, o modelo de *Software* como Serviço (SaaS) disponibiliza via rede uma quantidade virtualmente ilimitada de aplicações, que vão desde jogos infantis a complexos sistemas de *Enterprise Resource Planning* (ERP).

O problema tratado neste trabalho, definido em detalhes na Seção 2.1, limita-se ao contexto local de gerenciamento de um único *datacenter* cujos recursos de *hardware* são virtualizados e que possui as características fundamentais de uma nuvem computacional (vide características de serviços na Figura 1.1). Portanto, a aplicação dos métodos de gerenciamento a múltiplos *datacenters* não é abordada diretamente, sendo apontada como possibilidade de trabalho futuro. A infraestrutura e os serviços oferecidos por uma nuvem computacional estão inseridos em um ambiente dotado de uma rede local de alta velocidade e que pode ser formado por centenas de milhares de nós computacionais (IDC, 2012). Ambientes computacionais dessa natureza não podem ser gerenciados através de técnicas tradicionais de gerenciamento centralizado devido ao grande volume de estados que precisam ser monitorados concomitantemente (WUHIB; STADLER; SPREITZER, 2010). Dessa forma, cria-se a necessidade de técnicas que solucionem não só questões como alocação de recursos e tratamento de falhas, mas que também ofereçam a escalabilidade necessária para que esses ambientes operem de forma eficiente. Este trabalho propõe uma estratégia de Gerenciamento de Infraestrutura de *Datacenter* (DCIM) voltada a nuvens computacionais de larga escala que busca dissociar as responsabilidades de monitoramento e controle através da integração de técnicas de monitoramento distribuído e ações de controle centralizado.

1.2 Justificativa

A Seção 1.1 apresentou o contexto no qual identifica-se a necessidade de técnicas de DCIM que combinem a capacidade de gerenciar de forma eficiente os recursos de um *datacenter*, a possibilidade de identificar e tratar falhas de infraestrutura e a escalabilidade necessária para garantir sua viabilidade de operação em ambientes

que podem crescer de forma constante.

Na Seção 2.2, são apresentadas diversas soluções de DCIM, de acordo com as quais pode-se perceber que as estratégias de gerenciamento centralizado são preponderantes, o que se deve, em grande parte, ao fato desse ser o *modus operandi* das ferramentas atualmente disponíveis para a implantação de nuvens computacionas em infraestruturas reais. Tais abordagens têm como problema comum o baixo poder de escalabilidade do ambiente, visto que a existência de um nó controlador que acumula uma grande quantidade de responsabilidades relacionadas à gerência da infraestrutura indica uma forte tendência de sobrecarga do mesmo à medida que o número de nós computacionais que compõem o *datacenter* cresce.

Este trabalho busca amenizar esse problema distribuindo a atividade de monitoramento entre todos os nós que compõem a nuvem e restringindo as responsabilidades do nó controlador às ações de controle do ambiente. Essa abordagem é fundamentada em bons resultados obtidos em trabalhos que lançaram mão de estratégias distribuídas para o monitoramento de ambientes similares (WUHIB; STADLER; SPREITZER, 2010). Contudo, as soluções que apresentam resultados satisfatórios quanto à escalabilidade do sistema não abordam de forma integrada outros problemas relevantes na gerência de nuvens computacionais, a saber: eficiência na utilização de recursos (e.g. consumo energético da infraestrutura) e avaliação do nível de Qualidade de Serviço (QoS) oferecido.

1.3 Hipóteses

O panorama descrito na Seção 1.2 indica que ganhos significativos podem ser obtidos com a existência de uma arquitetura de gerenciamento que possibilite o emprego integrado de políticas de controle centralizado existentes na literatura e técnicas de monitoramento distribuído que permitam o crescimento do ambiente sem impactos relevantes em seu funcionamento. Baseando-se nesse indício, as seguintes hipóteses são elaboradas:

i. O monitoramento distribuído do *datacenter* diminuirá o volume de informações que o nó controlador necessita processar, o que indica uma tendência menor de sobrecarga nesse nó à medida que a quantidade de máquinas que compõem a nuvem computacional se eleva;

ii. Com sua responsabilidade pelo monitoramento do ambiente eliminada, o nó controlador passará a apresentar um tempo de resposta menor na aplicação das políticas de controle. Tendo em vista o caráter dinâmico de uma nuvem computacional, essa melhora poderá ter um impacto positivo direto na eficiência do *datacenter* e no nível de Qualidade de Serviço (QoS);

iii. A cada nó hospedeiro do *datacenter*, será atribuída a responsabilidade de monitoramento de um conjunto limitado de n nós vizinhos. Um valor de n pequeno tornará o *datacenter* demasiadamente estratificado, o que poderá ter um impacto negativo em sua eficiência. Por outro lado, um valor elevado de n aproximará a estratégia de gerenciamento proposta ao modelo centralizado, posto que cada nó hospedeiro passará a ter uma visão mais global do ambiente, o que ocasionará impactos negativos relacionados à escalabilidade. Portanto, poderão existir valores de n apropriados para que eficiência e escalabilidade sejam equilibradas.

1.4 Objetivos

1.4.1 Objetivo geral

O objetivo geral deste trabalho consiste em validar as hipóteses apresentadas na Seção 1.3 através do emprego de uma arquitetura de DCIM que combine técnicas de monitoramento distribuído e ações de controle centralizado com vistas à eficiência na utilização de seus recursos aliada ao provimento de níveis aceitáveis de QoS.

1.4.2 Objetivos específicos

Especificamente, este trabalho busca alcançar os objetivos abaixo:

i. Distribuir a responsabilidade de monitoramento de um ambiente computacional em nuvem entre todos os nós que o compõem e restringir a atuação do nó controlador a ações de controle;

ii. Associar políticas de distribuição e consolidação de máquinas virtuais às ações de controle do nó controlador para que o impacto de uma eventual diminuição em seu tempo de resposta seja analisado;

iii. Determinar a relação entre a quantidade de vizinhos que cada nó do *datacenter* deve monitorar e características operacionais do ambiente, como eficiência energética e QoS.

1.5 Metodologia

A metodologia empregada para a concepção da arquitetura de DCIM proposta e sua avaliação pode ser descrita de acordo com as etapas abaixo:

i. Revisão Bibliográfica (Seção 2.2): Políticas de DCIM são extraídas da literatura relacionada e empregadas como base de conhecimento para a concepção da arquitetura proposta e como referência de comparação para a avaliação dos resultados do trabalho. A revisão bibliográfica também explora artigos relacionados a *frameworks* e ferramentas de simulação para que se identifique a alternativa mais apropriada (Seção 2.2.2);

ii. Definição do problema (Seção 2.1): São realizadas a caraterização analítica do problema tratado e a especificação do sistema alvo do trabalho por meio das variáveis consideradas durante o desenvolvimento da arquitetura proposta;

iii. Definição da proposta (Capítulo 3): Após a revisão dos modelos de DCIM comumente adotados na indústria e estudados nas referência bibliográficas, é apresentado o modelo que consiste no núcleo deste trabalho;

iv. Adaptação das ferramentas e criação dos ambientes de simulação (Seção 4.1.5): Para tornar viável a reprodução do sistema alvo do trabalho em um ambiente de simulação, é abordada uma série de modificações nas ferramentas empregadas. Ademais, dados de um *trace* de utilização de recursos em um *cluster* real disponibilizado pela multinacional Google são utilizados para a representação dos *workloads* empregados nas simulações.

1.6 Estrutura do documento

O restante desta dissertação está organizado como segue: o Capítulo 2 realiza a caracterização do problema tratado, seguida de uma revisão bibliográfica que explora trabalhos relacionados a modelos de DCIM e ferramentas de simulação voltadas a ambientes computacionais em nuvem. O Capítulo 3 apresenta a proposta do

trabalho, analisando inicialmente as estratégias mais comuns de gerenciamento de infraestruturas virtualizadas e definindo formalmente o modelo da arquitetura de DCIM. O Capítulo 4 descreve todo o processo de criação dos ambientes de simulação, assim como os critérios adotados para a análise de desempenho apresentada nas Seções 4.2 e 4.3. No Capítulo 5, são feitas considerações finais acerca do trabalho com uma síntese de contribuições, limitações e perspectivas de trabalhos futuros. Por fim, o Apêndice A traz detalhes sobre a ferramenta CloudReports, que foi continuamente aperfeiçoada durante a execução deste trabalho e consiste em uma das principais contribuições do mesmo.

Capítulo 2

Caracterização do problema e revisão bibliográfica

Neste capítulo, o problema tratado neste trabalho é apresentado através da representação analítica do ambiente gerenciado e das condições atreladas à função objetivo (Seção 2.1). Em seguida, a Seção 2.2 explora uma série de trabalhos relacionados ao problema definido e aos campos de gerência de recursos em *datacenters* virtualizados (Seção 2.2.1) e simulação de ambientes computacionais em nuvem (Seção 2.2.2).

2.1 Definição do problema

Este trabalho aborda o problema do gerenciamento de máquinas virtuais em nuvens computacionais de larga escala com foco em duas necessidades básicas de tais ambientes: eficiência na utilização de recursos e manutenção da Qualidade de Serviço (QoS) oferecida.

O ambiente tratado consiste em um *datacenter* virtualizado composto por um conjunto N de p nós computacionais n interconectados tal que $N = [n_1, n_2, \ldots, n_p]$. Cada nó n_i possui capacidade de processamento $\bar{c}_i$ e quantidade memória de sistema $\bar{m}_i$. Assim, a capacidade total de processamento C do *datacenter* é dada por $\sum_{1 \leq i \leq p} \bar{c}_i$ e a quantidade total de memória M é dada por $\sum_{1 \leq i \leq p} \bar{m}_i$. Ademais, cada nó computacional possui limites máximo e mínimo de utilização de recursos expressos, respectivamente, por τ e μ, onde as três condições abaixo devem ser obedecidas:

$$0 < \tau \leq 1; \tag{2.1}$$

$$0 \leq \mu < 1; \tag{2.2}$$

$$\tau > \mu. \tag{2.3}$$

Em um dado instante t, o *datacenter* aloca um conjunto V composto por uma quantidade $\theta(t)$ de máquinas virtuais v. Uma dada máquina virtual v_j possui limites máximos de utilização de processamento ($\bar{\omega}_j$) e memória ($\bar{\gamma}_j$). A demanda por processamento é variante no tempo e expressa por $\omega_j(t)$, enquanto a demanda por memória é considerada constante e equivalente a seu limite máximo de utilização $\bar{\gamma}_j$ (CARRERA *et al.*, 2008).

Para que uma máquina virtual v_j possa ser alocada em um nó $n \in N$, considerando que $c_i(t)$ e $m_i(t)$ denotam o consumo de CPU e memória do nó n_i no instante t, todas as condições seguintes devem ser obedecidas para um único valor de i:

$$\exists\ n_i \mid \bar{\omega}_j < \bar{c}_i - c_i(t); \tag{2.4}$$

$$\exists\ n_i \mid \frac{\bar{\omega}_j + c_i(t)}{\bar{c}_i} < \tau; \tag{2.5}$$

$$\exists\ n_i \mid \bar{\gamma}_j < \bar{m}_i - m_i(t); \tag{2.6}$$

$$\exists\ n_i \mid \frac{\bar{\gamma}_j + m_i(t)}{\bar{m}_i} < \tau. \tag{2.7}$$

As equações 2.4 e 2.6 determinam que os limites máximos de utilização de processamento e memória de uma máquina virtual devem sempre ser inferiores à quantidade atual de recursos disponíveis no nó hospedeiro em que a mesma será alocada. Tal condição é complementada pelas equações 2.5 e 2.7, que estabelecem que uma alocação está sempre condicionada ao limiar τ de utilização de recursos no nó hospedeiro.

O estado de sobrecarga de um nó n_i é caraterizado por um consumo de processamento ou memória acima do limiar máximo τ em um instante t, como mostrado nas equações 2.8 e 2.9. De forma análoga, as equações 2.10 e 2.11 estabelecem que o estado de ociosidade de um nó n_i é caracterizado por um consumo de recursos inferior ao limiar mínimo μ.

$$\frac{c_i(t)}{\bar{c}_i} > \tau; \tag{2.8}$$

$$\frac{m_i(t)}{\bar{m}_i} > \tau; \tag{2.9}$$

$$\frac{c_i(t)}{\bar{c}_i} < \mu; \tag{2.10}$$

$$\frac{m_i(t)}{\bar{m}_i} < \mu. \tag{2.11}$$

O problema da alocação de recursos em um *datacenter* virtualizado consiste em atender à demanda aplicada ao sistema utilizando a menor parcela possível da infraestrutura disponível sem violar os limites máximos de utilização estabelecidos, ou seja, todas as máquinas virtuais devem ser alocadas na menor quantidade possível de nós computacionais sem que os limites individuais de utilização de cada nó sejam ultrapassados. Esse problema pode ser modelado como uma variação do problema NP-Completo conhecido como *knapsack problem* (SHACHNAI; TAMIR, 2001). Dessa forma, sendo A o número de nós computacionais que não alocam nenhuma máquina virtual e, por consequência, podem ser desativados sem comprometer o atendimento à demanda recebida pelo sistema, o problema de alocação de recursos supracitado passa a ser representado pela busca de heurísticas que consigam maximizar A de maneira sub-ótima sem violar nenhum limite de utilização, como descrito na Equação 2.12 e obedecendo às condições 2.13 a 2.16.

$$\max\left(A = \sum_{1 \leq i \leq p} \sigma(i)\right), \quad \sigma(i) = \begin{cases} 1 & : \quad c_i(t) = 0, \quad m_i(t) = 0 \\ 0 & : \quad \text{caso contrário} \end{cases}; \tag{2.12}$$

$$\forall n_i \in N: \quad \frac{c_i(t)}{\bar{c}_i} < \tau, \quad \frac{m_i(t)}{\bar{m}_i} < \tau; \tag{2.13}$$

$$\forall v_{j,i} \in V, \quad \forall t: \quad \sum_j \omega_j(t) \leq \bar{c}_i, \quad \omega_j(t) \leq \bar{\omega}_j; \tag{2.14}$$

$$\forall v_{j,i} \in V: \quad \sum_j \bar{\gamma}_j \leq \bar{m}_i; \tag{2.15}$$

$$\forall v_j \in V, \quad \forall t: \quad \sum_{1 \leq j \leq \theta(t)} \omega_j(t) \leq C, \quad \sum_{1 \leq j \leq \theta(t)} \bar{\gamma}_j \leq M. \tag{2.16}$$

A Equação 2.12 representa a função objetivo: maximizar a quantidade de nós computacionais desativados, onde ambos os consumos de processamento e memória são nulos, para que o consumo energético total do *datacenter* seja minimizado ainda que toda a demanda por recursos aplicada à infraestrutura continue sendo atendida. Contudo, a desativação de nós através de técnicas de consolidação de alocação de máquinas virtuais não pode ser realizada de forma demasiadamente agressiva, pois tal medida leva a altas taxas de sobrecarga nos nós hospedeiros, o que tem um impacto negativo direto em métricas de QoS. Portanto, a maximização de A deve ser realizada em consonância com as condições estabelecidas nas equações de 2.13 a 2.16. A Equação 2.13 determina que, para qualquer nó pertencente ao *datacenter*, as taxas de utilização de processamento e memória devem ser inferiores ao limiar máximo de utilização τ. A Equação 2.14 estabelece que, para qualquer máquina virtual alocada no *datacenter* em qualquer instante, seu consumo de processamento deve ser simultaneamente inferior à quantidade de recursos disponíveis em seu nó hospedeiro e à sua demanda total de processamento. Na Equação 2.15, é estabelecido que cada máquina virtual alocada deve possuir uma demanda por memória menor que quantidade de memória disponível em seu nó hospedeiro. Por fim, a Equação 2.16 define a premissa de que um *datacenter* está limitado a atender a uma demanda igual ou inferior à quantidade total de recursos presentes em sua infraestrutura. Portanto, considerando qualquer máquina virtual alocada no *datacenter* em qualquer instante, o somatório do consumo de processamento de todos os elementos de V deve ser sempre igual ou inferior à capacidade total de processamento C, assim como o somatório das demandas por memória deve ser sempre igual ou inferior à quantidade total de memória M.

2.2 Trabalhos relacionados

Para o levantamento do estado da arte na área em que este trabalho está inserido, foram realizadas revisões bibliográficas de forma sistemática através de bases de busca como Scopus, IEEE Xplore, ACM Digital Library e Google Scholar. Como o foco do problema abordado envolve técnicas de gerenciamento e análise de performance e escalabilidade de sistemas distribuídos virtualizados na forma de nuvens de recursos de larga escala, os termos utilizados para a formação das *strings* de busca dos artigos aqui referidos foram: "*cloud computing*", "*distributed computing*", "*distributed systems*", "*performance analysis*",

"*performance evaluation*", "*cloud management*", "*virtual machines management*", "*system virtualization*", "*resource virtualization*", "*resource management*" e "*resource allocation*". Inicialmente, foram considerados apenas artigos publicados em *journals* com classificação iguais ou superiores a B2 (Qualis Capes) nas áreas de Engenharias IV ou Ciências da Computação. Em seguida, o fator de impacto das publicações foi empregado como filtro de prioridade na seleção dos trabalhos. A última etapa de revisão consistiu em selecionar artigos de conferência com classificação iguais ou superiores a B2 (Qualis Capes) na área de Ciências da Computação que se enquadrassem diretamente no assunto tratado, embora poucos figurem no conjunto final das referências exploradas nesta seção.

2.2.1 Gerência de recursos em *datacenters* virtualizados

O surgimento de nuvens computacionais de larga escala e a necessidade de técnicas para o gerenciamento eficiente de seus recursos trazem consigo um extenso grupo de novos desafios e oportunidades para pesquisa, levantando também discussões sobre a necessidade de padronização das tecnologias envolvidas no processo de gerência de uma nuvem computacional. Carlson (2011) apresenta perspectivas para novas ferramentas de gerência de infraestruturas virtualizadas, analisando a possibilidade de que tais trabalhos convirjam para um padrão. Ferramentas e *frameworks* de gerência para nuvens computacionais também são discutidos sob o ponto de vista acadêmico por Gallard *et al.* (2012) e Chen, Zhang e Li (2011), e com foco em soluções para o mercado por Xu e Yang (2011) e Yan *et al.* (2011).

O problema formalizado na Seção 2.1 trata da alocação de recursos em *datacenters* virtualizados. Embora soluções sejam encontradas com frequência na literatura (Beloglazov e Buyya (2010a) e Islam *et al.* (2011)), poucas tratam as questões relacionadas à eficiência na utilização de recursos, métricas de QoS e escalabilidade do sistema de forma integrada. Carrera *et al.* (2008), Famaey *et al.* (2008) e Tang *et al.* (2007) confirmam que grande parte das soluções encontradas adota uma abordagem de gerenciamento centralizada e tem, por conseguinte, a escalabilidade do sistema como problema comum.

Um mecanismo de gerenciamento de sistemas distribuídos envolve técnicas de monitoramento e controle. Como as políticas de gerenciamento centralizado têm em comum o problema de escalabilidade, este trabalho tem foco em soluções

de monitoramento distribuído. Wuhib, Stadler e Spreitzer (2010) propõem uma arquitetura de gerenciamento inteiramente distribuída para nuvens computacionais de larga escala, cujos objetivos são alocação justa de recursos, satisfação de demanda, custo de reconfiguração do ambiente e escalabilidade. Embora todos os objetivos sejam satisfeitos no ambiente abordado pelo trabalho, o conceito de eficiência na utilização de recursos não leva em conta a quantidade de nós computacionais ativos na nuvem. Ademais, a solução não aborda o tratamento de eventuais falhas no ambiente, sendo esta questão indicada como oportunidade para trabalhos futuros.

Chaves, Uriarte e Westphall (2011) apresentam uma discussão sobre o reaproveitamento de alternativas de monitoramento já existentes e a definição de uma arquitetura de monitoramento de propósito geral (PCMONS) voltada para nuvens privadas são apresentadas. A arquitetura é composta por três camadas (Infraestrutura, Integração e Visão), sendo a camada de integração o principal ponto de ações dos módulos que monitoram a nuvem. O monitoramento é feito tanto localmente em cada nó computacional, por meio de injeções de scripts nas máquinas virtuais alocadas, quanto em módulos centrais, que obtêm dados em uma base unificada.

A arquitetura proposta neste trabalho trata as máquinas virtuais alocadas como caixas pretas de propriedade exclusiva dos usuários da infraestrutura e, por conseguinte, invioláveis. Portanto, o artifício de injeção de scripts para monitoramento será evitado. Ademais, o protocolo utilizado para o monitoramento de recursos nos nós será inteiramente distribuído, centralizando-se apenas as ações de controle aplicadas no ambiente.

No que diz respeito especificamente à natureza do problema definido na Equação 2.12, torna-se clara a necessidade de mecanismos de monitoramento contínuo dos nós que compõem o *datacenter* virtualizado, posto que o valor de A é dinâmico, alterando-se continuamente com a variação de t. Mouratidis, Bakiras e Papadias (2006), Prieto e Stadler (2007) e Wuhib *et al.* (2009) apresentam vários protocolos para monitoramento contínuo que permitem o acompanhamento da evolução do estado de consumo de recursos em um sistema distribuído.

A Tabela 2.1 mostra uma visão comparativa dos trabalhos que apresentam estratégias de DCIM abordados nessa seção de acordo com o ambiente gerenciado, modelos de monitoramento e controle e tipos de recursos gerenciados.

Tabela 2.1: Tabela comparativa dos trabalhos relacionados à gerência de recursos em *datacenters* virtualizados.

Trabalho	**Ambiente gerenciado**	**Modelo de monitoramento**	**Modelo de controle**	**Recursos gerenciados**
Gallard *et al.* (2012)	Nuvem privada	Centralizado	Centralizado	Sistema operacional, CPU e memória
Chen, Zhang e Li (2011)	Nuvem pública	Centralizado	Centralizado	CPU, memória, armazenamento e rede
Xu e Yang (2011)	Nuvem privada	Centralizado	Centralizado	CPU, memória e armazenamento
Yan *et al.* (2011)	Nuvem híbrida	Centralizado	Centralizado	CPU, memória, armazenamento e rede
Beloglazov e Buyya (2010)	Nuvem pública	Híbrido	Centralizado	CPU, memória, armazenamento e rede
Islam *et al.* (2011)	Nuvem pública	Centralizado	Centralizado	CPU
Fu (2010)	Nuvem privada	Centralizado	Centralizado	CPU e memória
Chaves, Uriarte e Westphall (2011)	Nuvem privada	Híbrido	Centralizado	CPU e memória
Prieto e Stadler (2007)	N/A	Distribuído	Híbrido	Rede
Wuhib *et al.* (2009)	N/A	Distribuído	Centralizado	Rede
Wuhib, Stadler e Spreitzer (2010)	Nuvem privada	Distribuído	Distribuído	CPU e memória

2.2.2 *Frameworks* e ferramentas de simulação

Este trabalho busca analisar a performance da estratégia de gerenciamento proposta em *datacenters* de larga escala que representam infraestruturas de provedores de IaaS. Contudo, a realização de experimentos que possam ser seguidamente repetidos de forma consistente em uma infraestrutura real tem alto grau de complexidade e custos elevados (BELOGLAZOV; BUYYA, 2012). Portanto, para que a avaliação da proposta fosse viável e para garantir a execução repetida e consistente dos experimentos, este trabalho faz uso de simulações. Esta seção apresenta um levantamento de *frameworks* e ferramentas de simulação extraídos da literatura relacionada e voltados especificamente para a reprodução de ambientes computacionais em nuvem.

Como descrito em maiores detalhes na Seção 4.1.6, uma das métricas utilizadas durante a análise de desempenho é o consumo energético total do *datacenter*. Embora uma quantidade razoável de ferramentas voltadas para a simulação de sistemas distribuídos e grades computacionais possam ser encontrada, ainda existem poucas alternativas para a simulação de *datacenters* virtualizados com suporte à análise de consumo energético. Como exemplos, pode-se citar o *framework* SimGrid (CASANOVA; LEGRAND; QUINSON, 2008), que simula ambientes parelalizados e distribuídos de larga escala, e o *toolkit* GridSim (BUYYA; MURSHED, 2002), que possibilita a modelagem de infraestruturas distribuídas, aplicações, recursos e algoritmos de escalonamento. Contudo, essas ferramentas não reproduzem a virtualização dos recursos, criando-se a necessidade de criação de extensões ou simuladores inteiramente novos. Sulistio *et al.* (2008) apresenta um *toolkit* que exemplifica tais extensões, embora não seja focado especificamente em ambientes computacionais em nuvem.

No que diz respeito a ferramentas de simulação para infraestruturas virtualizadas, a plataforma iCanCloud (NÚÑEZ *et al.*, 2012) trata-se de um projeto de código aberto que busca modelar e simular nuvens computacionais. É baseada no simulador de redes OMNET e oferece uma API baseada no padrão POSIX para a modelagem de aplicações. Entretanto, não oferece a possibilidade de representação do consumo energético da infraestrutura. O simulador GreenCloud (KLIAZOVICH *et al.*, 2010) é uma extensão do simulador NS-2 com funcionalidades adicionais para a análise de ambientes computacionais em nuvem. Oferece a possibilidade de modelagem do consumo energético de nós computacionais e elementos de

infraestrutura de rede como *switches* e enlaces. Contudo, não oferece suporte à representação de máquinas virtuais e aspectos relacionados à simulação de aplicações, como algoritmos de escalonamento de tarefas.

Este trabalho optou pelo uso do *framework* CloudSim (CALHEIROS *et al.*, 2011) combinado à ferramenta CloudReports (SÁ; SOARES; GOMES, 2011). O CloudSim trata-se de uma *engine* de simulação com suporte à representação de máquinas virtuais, algoritmos de escalonamento de tarefas e modelagem de consumo energético. O CloudReports é uma ferramenta que emprega o CloudSim e gerencia os dados produzidos pelos experimentos, além de oferecer uma interface gráfica para a modelagem da infraestrutura simulada. Ambos são descritos em detalhes no Apêndice A.

Aksanli, Venkatesh e Rosing (2012) fazem um estudo comparativo que analisa as diferenças entre algumas ferramentas de simulação de *datacenters* voltadas para a avaliação de consumo energético. Kocaoglu, Malak e Akan (2012) exploram alguns dos aspectos chave do campo de *green computing* e ratifica a importância das ferramentas de simulação para que se avaliem novas estratégias de gerenciamento, arquiteturas e protocolos. Por fim, Buyya, Ranjan e Calheiros (2009) discutem oportunidades de pesquisa e desafios que surgem com o advento de simuladores para ambientes de computação em nuvem com suporte à modelagem do consumo energético e Calheiros *et al.* (2011), Beloglazov e Buyya (2010b) e Kim, Beloglazov e Buyya (2009) apresentam resultados provenientes do emprego dessas ferramentas.

2.3 Resumo

Este capítulo apresentou uma formulação analítica do problema tratado neste trabalho na Seção 2.1 e, em seguida, explorou a pesquisa bibliográfica realizada através de uma discussão de trabalhos relacionados à gerência de recursos em *datacenters* virtualizados e às principais ferramentas voltadas para a simulação de sistemas distribuídos e ambientes computacionais em nuvem (Seção 2.2).

Capítulo 3

Definição da proposta

Este capítulo aborda os principais conceitos relacionados à arquitetura de gerenciamento de infraestrutura de *datacenter* (DCIM) proposta nesta dissertação. Inicialmente, é realizada uma discussão acerca dos modelos de gerenciamento mais utilizados atualmente de acordo com a literatura relacionada e ferramentas de código aberto voltadas para a criação e gerência de nuvens computacionais. Em seguida, a Seção 3.2 caracteriza a proposta do trabalho fornecendo uma visão geral de seu funcionamento. As subseções seguintes realizam um detalhamento da arquitetura, onde são exploradas características dos mecanismos de controle e monitoramento do *datacenter*.

3.1 Modelo centralizado de DCIM

Uma nuvem computacional de larga escala é composta por uma infraestrutura virtualizada que serve simultaneamente como plataforma para um elevado número de aplicações. Essas aplicações são executadas em servidores representados por máquinas virtuais, cuja utilização de recursos é monitorada e controlada pelo detentor da infraestrutura. Como a demanda por recursos muda constantemente com o tempo, cria-se a necessidade de mecanismos de gerência capazes de operar de forma dinâmica, monitorando a evolução do uso geral da infraestrutura e executando ações de controle que busquem maximizar a eficiência na utilização da mesma. Um dos principais objetivos de uma estratégia de DCIM consiste essencialmente em utilizar o mínimo possível da infraestrutura disponível para alocar toda a demanda de aplicações e máquinas virtuais aplicada ao sistema sob requisitos

mínimos de performance pré-estabelecidos. Contudo, a solução ótima para atender a uma demanda de recursos flutuante por meio de uma quantidade limitada de nós computacionais tem alto custo computacional (SHACHNAI; TAMIR, 2001), o que justifica o emprego de heurísticas com desempenho sub-ótimo para solucionar problemas de tal natureza.

Para que se compreenda de forma mais ampla o contexto no qual este trabalho está inserido, é necessário que se apresente uma visão geral da arquitetura de gerenciamento predominante das chamadas nuvens computacionais. Como pode-se encontrar de forma recorrente na literatura (e.g. (BELOGLAZOV; ABAWAJY; BUYYA, 2012) e (LEE; ZOMAYA, 2011)), o modelo padrão de implementação de DCIM em *datacenters* virtualizados é baseado em um paradigma de gerenciamento centralizado, onde um nó controlador, também denominado estação de controle[1], é responsável pela coordenação geral da infraestrutura e comunicação com clientes. Embora esse modelo tenha sido vastamente explorado na literatura relacionada, como já descrito na Seção 2.2.1, poucos trabalhos levam em consideração a capacidade de escalabilidade do ambiente. Tal característica é de suma importância em *datacenters* de larga escala, posto que, à medida que o *datacenter* em questão torna-se maior, a quantidade de informação com a qual o nó controlador deve lidar cresce a ponto de inviabilizar a operação do mesmo sob exigências mínimas de performance, expondo assim a deficiência da arquitetura no que concerne à escalabilidade do sistema.

A Figura 3.1 mostra uma visão simplificada da arquitetura centralizada de DCIM em uma nuvem de recursos. A camada de controle é representada por uma estação dotada de uma pilha de *software* que normalmente é composta por um sistema operacional com suporte a um *middleware* de nuvem. O restante da infraestrutura é composta por nós que hospedam máquinas virtuais e que possuem uma pilha de *software* semelhante, embora os componentes presentes na camada de *middleware* sejam distintos dos observados na estação de controle. Tais características podem ser constatadas na documentação de dois dos principais projetos de código aberto voltados para o gerenciamento de infraestruturas de nuvens computacionais, a saber: Eucalyptus e OpenNebula.

Por meio de uma visão global do ambiente, a estação de controle orquestra

[1]Nesta dissertação, as expressões "nó controlador" e "estação de controle" são utilizadas indistintamente.

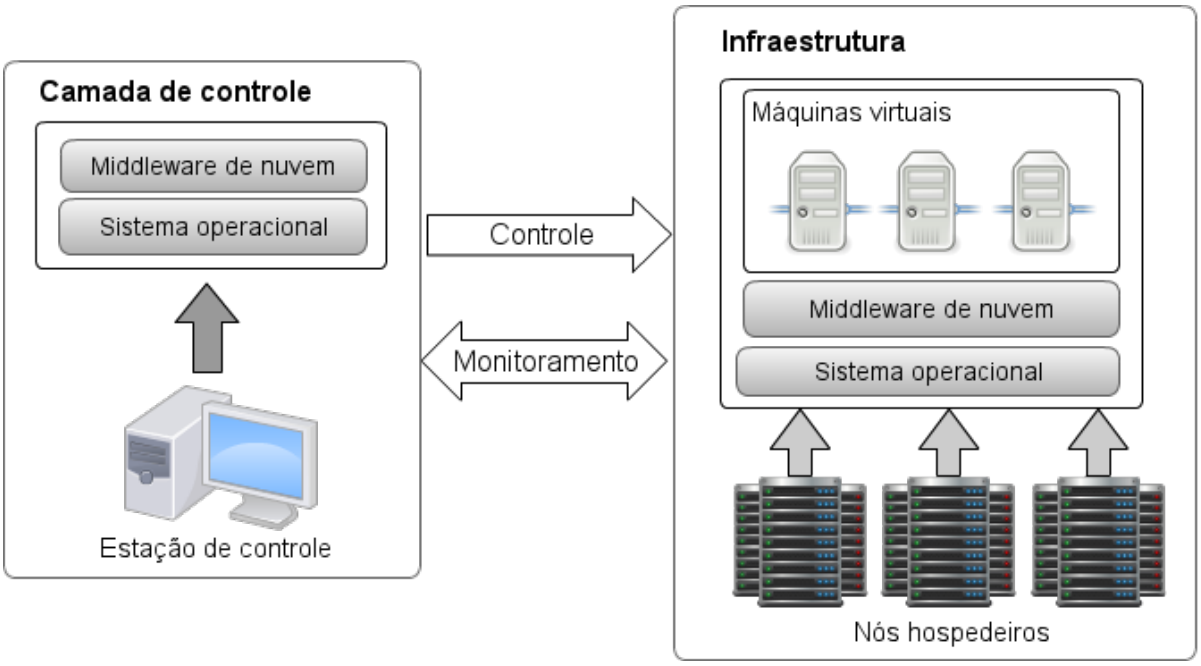


Figura 3.1: Arquitetura de DCIM centralizada em uma nuvem computacional.

todas as atividades realizadas pelos chamados nós hospedeiros, que atuam de forma passiva no ambiente. Para tal, uma abordagem *pull* de gerenciamento é empregada para extrair regularmente dados de utilização de recursos e *status* de operação de cada nó. O termo *pull* remete ao estado passivo dos nós hospedeiros, posto que as informações são sempre "puxadas" pela estação de controle, sendo essa a única responsável pelos mecanismos de gerência do ambiente. Para que essa extração de dados seja viável, é comum que o *middleware* de nuvem presente em cada nó hospedeiro possua um módulo de monitoramento com funcionamento baseado em múltiplos sensores (também denominados *probes*) para coleta de dados como uso de CPU, consumo de memória e atividade das interfaces de rede disponíveis.

Alternativamente ao modelo apresentado na Figura 3.1, uma variante da estratégia centralizada de DCIM pode ser apresentada com o nó controlador sendo responsável unicamente pelas ações de controle, enquanto outros nós também presentes na camada de controle realizam o monitoramento do ambiente. É importante observar que, embora tal variante delegue o monitoramento a outros nós e diminua a possibilidade de sobrecarga no nó controlador, a camada de controle permanece com as responsabilidades de monitorar e controlar o ambiente, o que não elimina problemas de sobrecarga nessa camada em *datacenters* de larga escala.

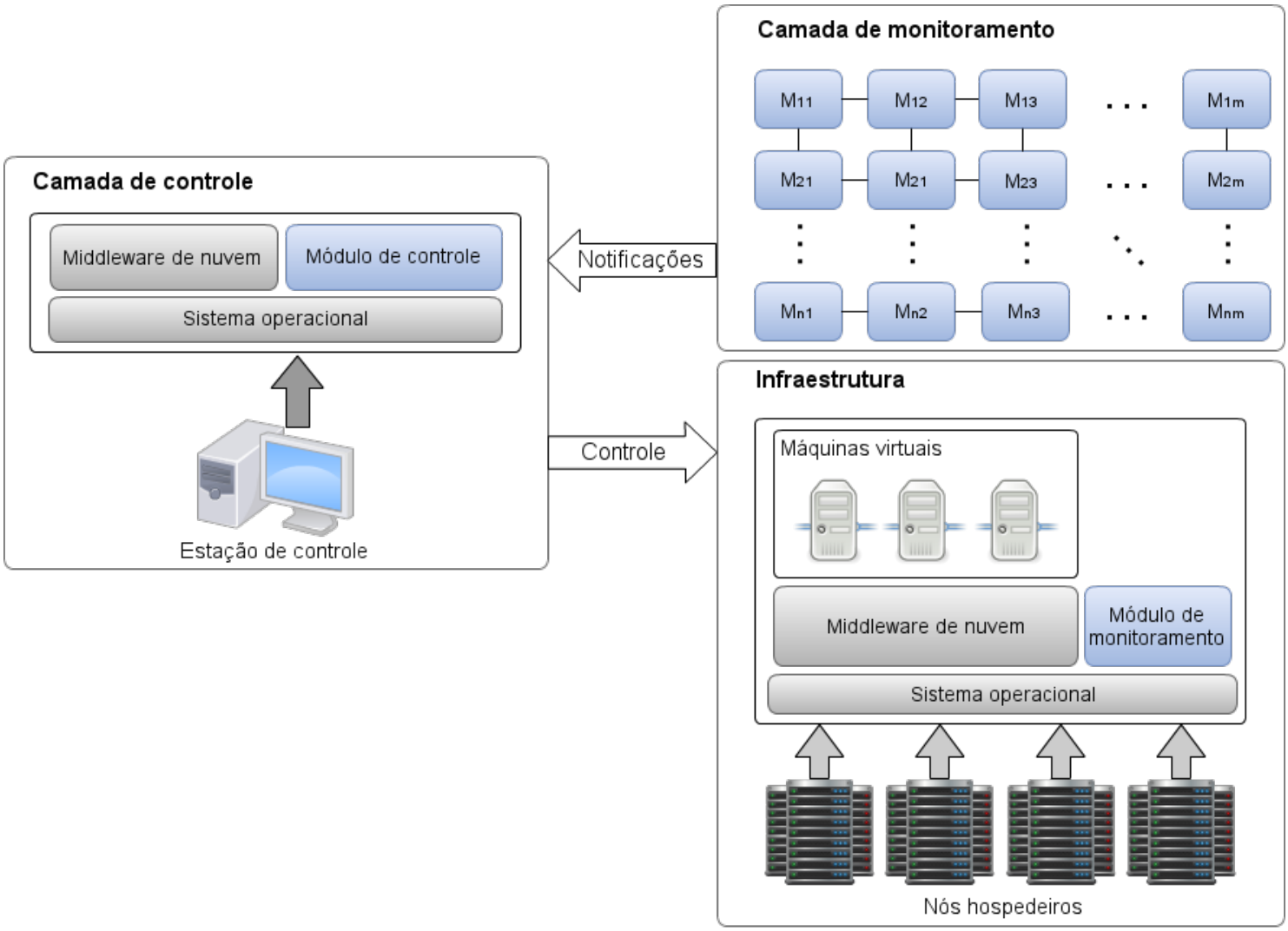


Figura 3.2: Arquitetura de DCIM com monitoramento distribuído e controle centralizado.

3.2 Caracterização do modelo proposto

Em contrapartida ao modelo *pull*, este trabalho propõe um protocolo de gerenciamento onde os nós hospedeiros abandonam a postura passiva descrita na seção anterior e passam a desempenhar um papel ativo no monitoramento do ambiente. Como ilustrado na Figura 3.2, adiciona-se à arquitetura uma camada específica de monitoramento. Nessa nova camada, cada nó hospedeiro possui uma quantidade limitada de n nós vizinhos, com os quais estabelece uma troca periódica de informações. A estação de controle, apesar de continuar detendo o poder de comunicação com qualquer nó presente na nuvem, não é mais responsável pelo monitoramento, limitando-se apenas a ações de controle. Dessa forma, a arquitetura

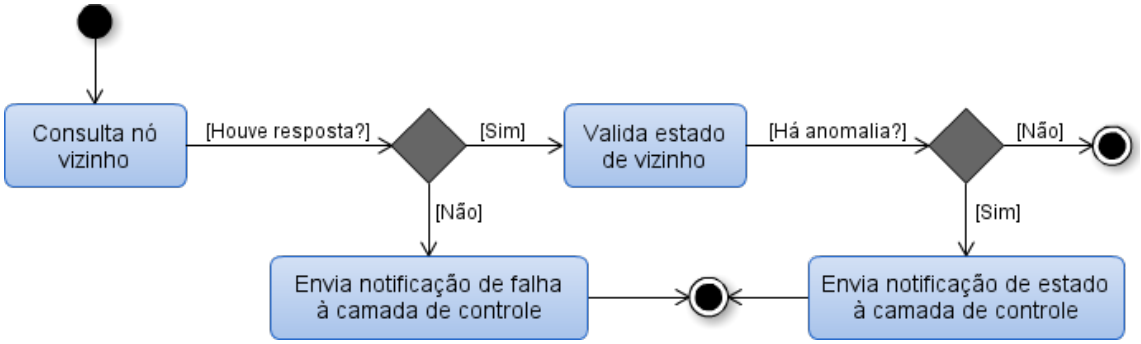


Figura 3.3: Fluxograma de operação do mecanismo de monitoramento distribuído para cada nó hospedeiro.

proposta adota uma abordagem de monitoramento onde as informações, no formato de notificações, são enviadas à estação de controle apenas quando necessário. Enquanto o monitoramento é feito de forma distribuída e em um contexto limitado, posto que cada nó hospedeiro processa informações apenas de seus nós vizinhos, o controle é mantido centralizado, recebendo notificações e tomando as ações correspondentes. Dessa forma, espera-se obter um maior poder de escalabilidade para o *datacenter*.

Os blocos em azul na Figura 3.2 representam os diferenciais da arquitetura proposta quando comparada ao modelo descrito na Figura 3.1. No nó hospedeiro, o módulo de monitoramento não se limita mais à coleta de informações por meio de *probes*. Além dessa atividade, tal módulo estabelece uma troca de relatórios de estado com nós vizinhos, criando uma rede de comunicação representada na figura como uma camada de monitoramento. A disposição desse módulo externamente ao *middleware* de nuvem não é exigida, existindo a possibilidade integração entre ambos. No nó contolador, o módulo de controle é responsável pelo recebimento de notificações provindas da camada de monitoramento e pela tomada direta de ações de controle no ambiente. Assim como no módulo de monitoramento, existe também a possibilidade real de integração entre o *middleware* de nuvem e o módulo de controle.

O fluxograma da Figura 3.3 descreve o *modus operandi* do mecanismo de monitoramento proposto. Inicialmente, cada nó hospedeiro consulta seus nós vizinhos acerca de seu estado de operação. As informações recebidas podem incluir taxa de consumo de recursos, quantidade e identificação das máquinas virtuais atualmente alocadas e *uptime* de operação. Caso não haja resposta à consulta, uma falha é notificada à estação de controle e o nó aguarda pelo

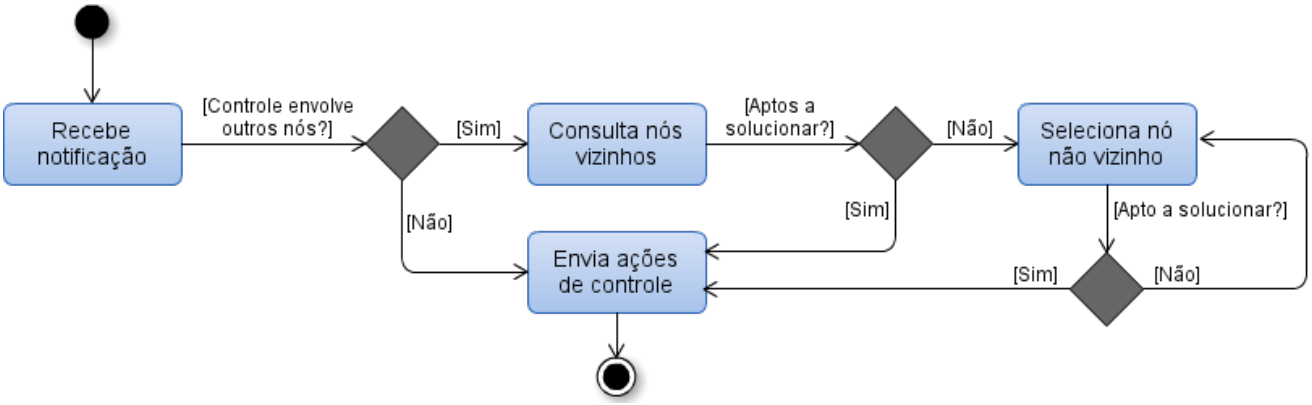


Figura 3.4: Fluxograma de operação do mecanismo de controle centralizado para o nó controlador.

próximo ciclo de monitoramento. Uma vez recebida a resposta à consulta, cada nó realiza uma validação do estado de operação de seus vinhos. Essa ação consiste basicamente na verificação de possíveis violações ao SLA adotado no gerenciamento da infraestrutura. Caso uma violação seja detectada, é caracterizada uma anomalia, que é prontamente reportada à estação de controle. Após essa fase, os nós aguardam o próximo ciclo de monitoramento. É importante ressaltar que, embora o mecanismo retratado na Figura 3.3 possibilite a notificação de falhas à camada de controle, este trabalho tem foco especificamente no impacto do monitoramento por meio de notificações de estado. A possibilidade de aplicar o modelo proposto em uma estratégia voltada à detecção e reparação de falhas em nós hospedeiros é indicada como possibilidade de trabalho futuro na Seção 5.2.

Na Figura 3.4, é apresentado o fluxograma de operação do mecanismo de controle centralizado. Nesse ponto, torna-se mais clara a proposta deste trabalho por uma mescla entre uma abordagem *pull* distribuída de monitoramento, onde cada nó hospedeiro consulta o estado de operação de seus vizinhos, e uma abordagem *push* centralizada de controle, onde informações sobre estados falhos e quebras de SLA são repassadas (ou "empurradas") diretamente ao nó controlador.

O início do fluxo consiste na recepção de uma notificação de estado anômalo ou falho pela estação de controle. Em seguida, é verificado se a ação de controle correspondente ao estado reportado envolve outros nós hospedeiros além do nó alvo da notificação. Caso envolva, o nó controlador consulta os nós vizinhos ao ponto onde o problema foi identificado para verificar se os mesmos são capazes de solucioná-lo. Se tal alternativa for viável, ações de controle correspondentes são enviadas aos

nós vizinhos. Caso contrário, o nó controlador consulta outros nós com o intuito de encontrar um apto a solucionar o problema e enviar as ações de controle. As técnicas de seleção de nós não vizinhos não serão exploradas detalhadamente, pois dependem de implementações específicas e fogem ao escopo deste trabalho. A estratégia de dar prioriade aos nós vizinhos para a solução do problema advém da necessidade de reduzir o escopo no qual a estação de controle age para suas tomadas de decisão. Tal medida busca diminuir a quantidade de informação a ser processada e, por conseguinte, melhorar o tempo de resposta do nó controlador. A eficácia dessa estratégia será explorada em detalhes no Capítulo 4. Por fim, caso a ação de controle correspondente ao estado notificado não envolva a utilização de outros nós além do nó alvo da notificação, a mesma é enviada de imediato.

Como cada nó hospedeiro é monitorado por n nós vizinhos, a estação de controle poderá receber até $(n-1)$ notificações de estado referentes a um único problema. Portanto, implementações da arquitetura devem considerar tal fato e traçar estratégias para tratamento e utilização de múltiplas notificações. Uma possível aplicação envolve a dimuição de falsos positivos através da criação de regras de tolerância baseadas na quantidade de notificações recebidas para um estado específico. No momento em que o nó controlador recebe a primeira notificação relacionada a um problema para o qual não tenha sido realizada nenhuma ação anterior de controle, a mesma é armazenada em registro e nenhuma ação subsequente é tomada. Dessa forma, ações de controle só seriam executadas se dois ou mais nós reportarem o mesmo problema ao nó controlador.

Dentre as possíveis ações que podem ser tomadas pela estação de controle, este trabalho considera as três alternativas abaixo:

i. **Migração de máquinas virtuais:** consiste em transferir uma máquina virtual alocada em um nó n_i a outro nó n_j da nuvem. As migrações surgem como possível solução para casos de violação de limites de utilização de recursos, dimiuição do intervalo Δt em que as máquinas virtuais ficam inoperantes quando os nós em que estão alocadas entram em estado falho e implementações de políticas que visem aumentar a eficiência de utilização de recursos e consumo energético da nuvem.

ii. **Alocação e desalocação de máquinas virtuais:** a inserção de novas máquinas virtuais no ambiente ou a desativação temporária ou permanente

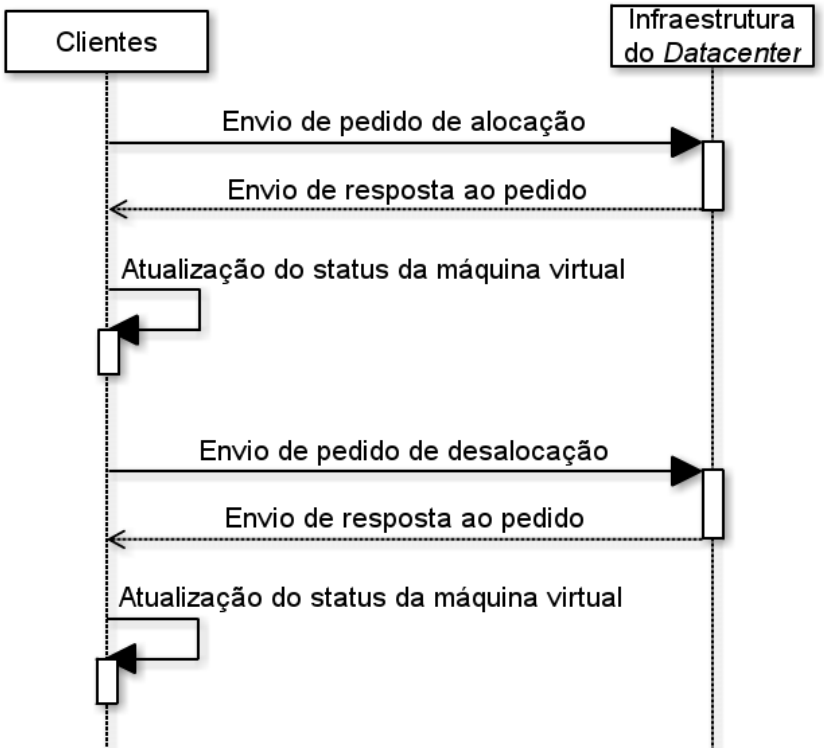


Figura 3.5: Visão geral do fluxo de requisições de alocação e desalocação de máquinas virtuais no *datacenter*.

das mesmas podem ser úteis para evitar quebras de SLA.

iii. **Ativação e desativação de nós:** o desligamento completo ou ativação de modo *standby* em nós ociosos podem aumentar sobremaneira a eficiência energética da nuvem, reduzindo seus custos de operação. Por outro lado, a reativação de nós desligados pode ser necessária para atender à demanda aplicada ao *datacenter*.

A seções seguintes decrevem detalhadamente a modelagem da arquitetura proposta nesta dissertação e retratada na Figura 3.2.

3.2.1 Módulo de controle

O diagrama de sequência da Figura 3.5 fornece uma visão geral da modelagem do fluxo de requisições de alocação e desalocação de máquinas virtuais na infraestrutura do *datacenter*. Os clientes atendidos por essa infraestrutura possuem uma quantidade arbitrária de máquinas virtuais, das quais podem enviar pedidos de alocação ou desalocação. Sempre que recebe um pedido, o *datacenter* o processa e envia uma resposta ao cliente, que pode ser uma confirmação ou rejeição. Ao

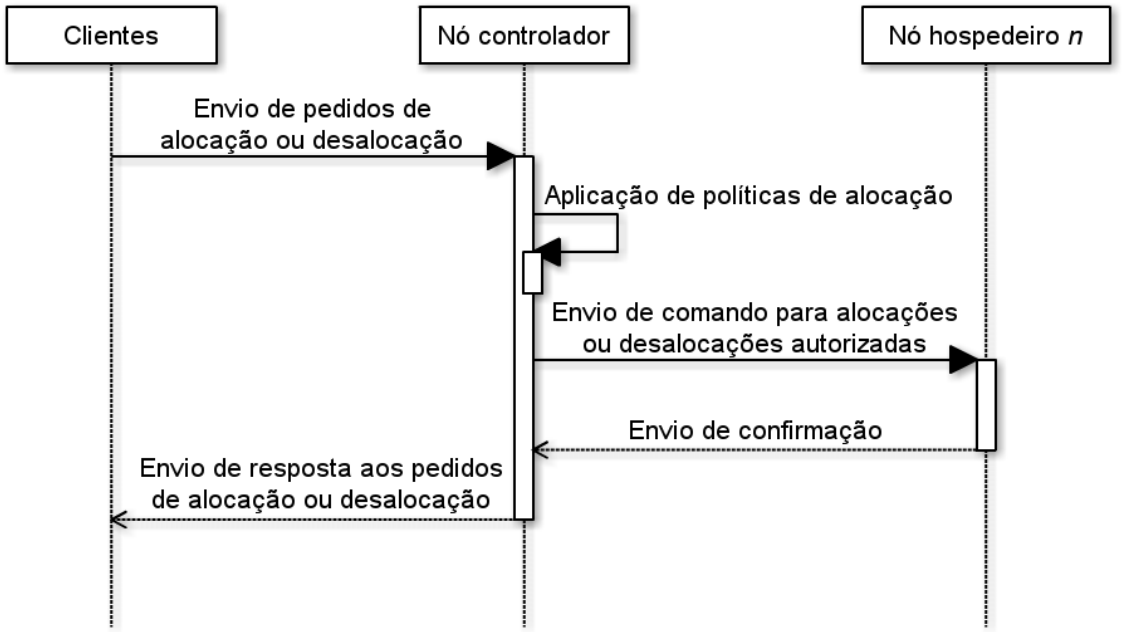


Figura 3.6: Diagrama de sequência representativo da infraestrutura do *datacenter* e dos processos de alocação e desalocação de máquinas virtuais.

receber a resposta, o cliente possui informações suficientes para determinar o status de alocação da máquina virtual (e.g. alocada ou desalocada).

O diagrama de sequência mostrado na Figura 3.6 representa a interação entre o nó controlador e um dado nó hospedeiro n sempre que um pedido de alocação ou desalocação de máquinas virtuais é recebido pelo *datacenter*. Quando um pedido chega ao nó controlador, o mesmo aplica as políticas de alocação vigentes para determinar se a ação requisitada pelo cliente é viável e decidir, nos casos de alocação, em qual nó hospedeiro a máquina virtual será alocada. Após essa decisão, o nó hospedeiro recebe um comando direto do nó controlador e, em seguida, envia uma confirmação de execução da ação solicitada. Por fim, o cliente recebe a resposta ao seu pedido, onde são enviados dados relativos às operações que foram realizadas em decorrência direta do mesmo.

O processo de alocação de desalocação de máquinas virtuais em um nó hospedeiro é representado no diagrama da Figura 3.7. Ao receber um comando de alocação ou desalocação provindo da estação de controle, o nó hospedeiro realiza três ações básicas, a saber: aloca ou desaloca a máquina virtual em questão, atualiza seus registros onde constam os identificadores das máquinas virtuais atualmente alocadas e atualiza seus registros relacionados à quantidade de recursos disponíveis

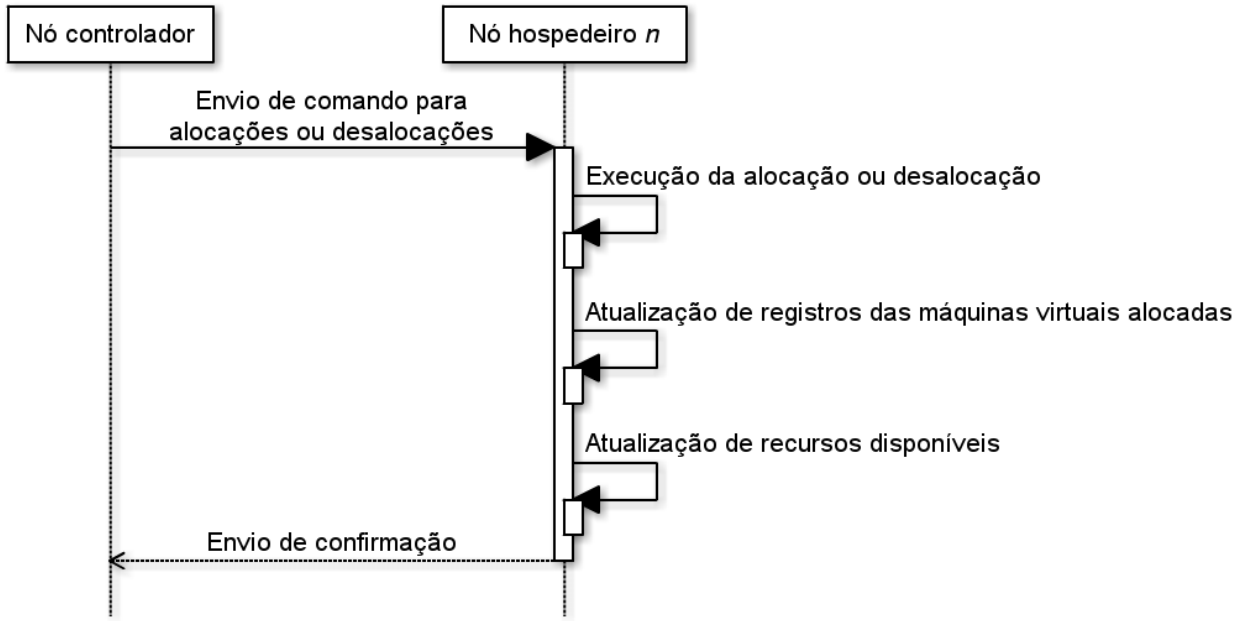


Figura 3.7: Diagrama de sequência representativo do processo de alocação e desalocação de máquinas virtuais em um nó hospedeiro.

para novas alocações. Logo que todas essas etapas são concluídas, uma confirmação é enviada ao nó controlador com informações sobre todas as ações realizadas.

3.2.2 Módulo de monitoramento

O diagrama de sequência da Figura 3.8 apresenta uma visão geral do fluxo de comunicação entre dois nós hospedeiros vizinhos. Para cada um de seus vizinhos, um dado nó hospedeiro segue uma rotina equivalente, que consiste em solicitar periodicamente informações na forma de relatórios de estado. O diagrama mostra que é realizado primeiramente o envio de uma requisição de relatório a um nó vizinho. Este, por sua vez, gera um relatório e o envia como resposta à requisição. Após o recebimento, o nó hospedeiro reinicia a contagem do período de monitoramento do nó vizinho em questão. Esta rotina de comunicação é então repetida de forma contínua.

Na Figura 3.9, é mostrado um diagrama de sequência que representa o processo de geração de relatórios. Ao receber uma requisição de relatório proveniente de um nó vizinho, o nó hospedeiro realiza uma análise de recursos para coletar todas as informações necessárias para a composição do relatório. Uma vez coletados todos os dados, o relatório é gerado e então enviado ao nó vizinho solicitante. Esse processo é realizado por todos os nós computacionais que compõem o *datacenter*

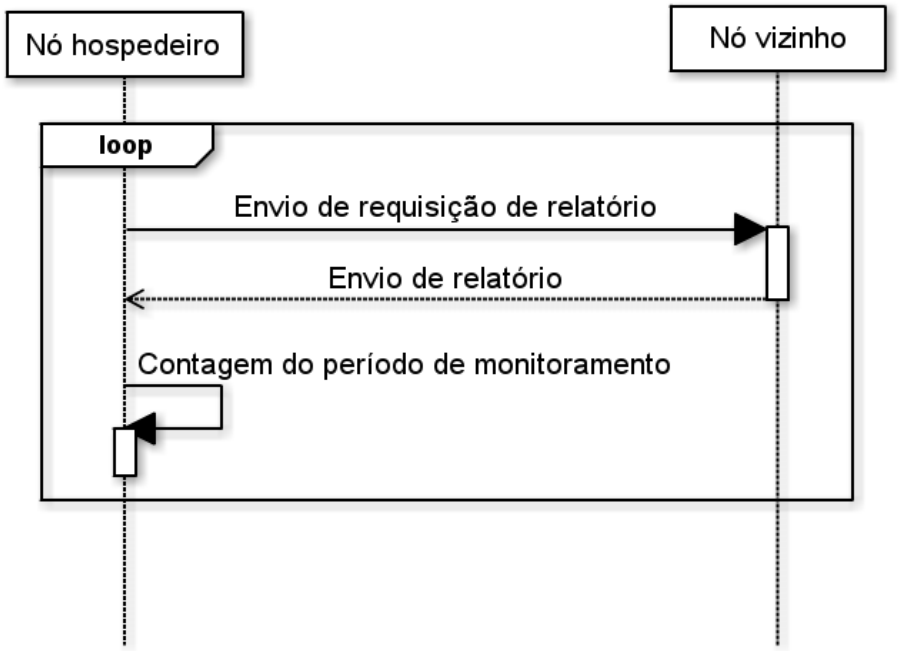


Figura 3.8: Visão geral do fluxo de comunicação entre dois nós hospedeiros vizinhos.

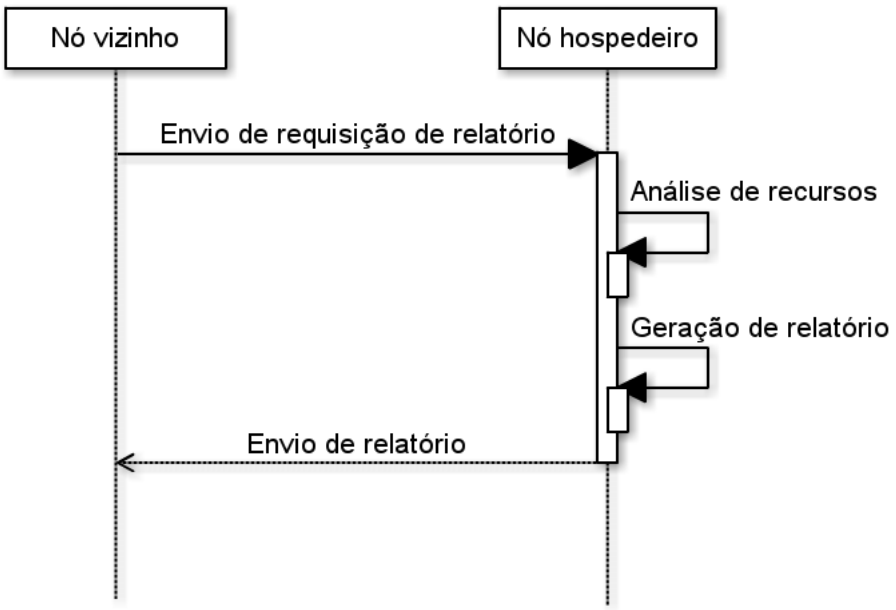


Figura 3.9: Diagrama de sequência representativo do prodesso de geração de relatórios.

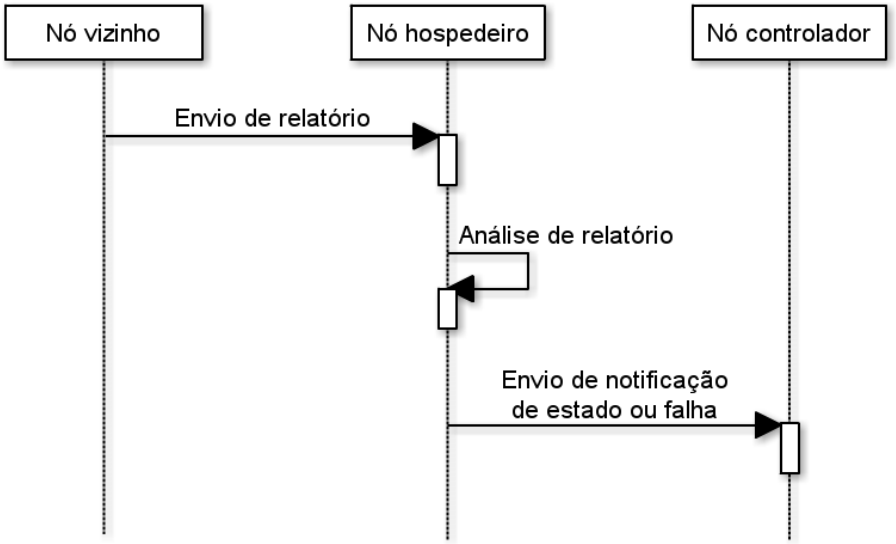


Figura 3.10: Diagrama de sequência representativo do mecanismo de envio de notificações.

com exceção da estação de controle. Cada nó hospedeiro requisita os relatórios de estado periodicamente a todos os seus vizinhos.

O mecanismo de envio de notifições do módulo de monitoramento está descrito na Figura 3.10. Sempre que um relatório de estado é recebido de um nó vizinho, o nó hospedeiro dá início a uma análise de dados baseada em requisitos que envolvem exigências relacionadas a QoS. Caso essa análise aponte que o modo de operação do nó vizinho infringe os requisitos estabelecidos, uma notificação de estado é enviada à estação de controle. Existe ainda a possibilidade de uma requisição de relatório por parte de um nó hospedeiro não obter resposta do nó vizinho correspondente. Nesse caso, a estação de controle pode ser notificada através do envio de uma notificação de falha.

3.3 Resumo

Este capítulo apresentou a proposta de arquitetura de DCIM com controle centralizado e monitoramento distribuído que consiste no ponto central deste trabalho. A Seção 3.1 discutiu a arquitetura de gerenciamento predominante em nuvens computacionais de acordo com referências extraídas da literatura relacionada e abordou os principais problemas relacionados à mesma. A Seção 3.2 forneceu uma visão geral do funcionamento da arquitetura proposta e, em seguida, as Seções 3.2.1 e 3.2.2 apresentaram a modelagem da arquitetura com detalhes específicos sobre as estratégias de controle de monitoramento envolvidas.

Capítulo 4

Resultados

Este capítulo apresenta uma visão detalhada da metodologia aplicada para a realização dos experimentos de simulação (Seção 4.1) e a avaliação dos resultados extraídos dos experimentos por meio de uma análise de desempenho sob os pontos de vista de eficiência energética do *datacenter* (Seção 4.2) e métricas relacionadas à Qualidade de Serviço (Seção 4.3).

4.1 Metodologia de simulação

Como discutido anteriormente na Seção 2.2.2, a realização de experimentos de larga escala que possam ser seguidamente repetidos de forma consistente em uma infraestrutura real tem alto grau de complexidade e custos elevados (BELOGLAZOV; BUYYA, 2012). Portanto, o processo de avaliação da estratégia de gerenciamento proposta neste trabalho é realizado através de experimentos de simulação. Para que o modelo apresentado no capítulo anterior fosse reproduzido em um ambiente de simulação e tivesse sua performance posteriormente avaliada, as ferramentas apresentadas no Apêndice A foram empregadas. Contudo, uma série de modificações no *framework* CloudSim se fizeram necessárias, além de ajustes na ferramenta CloudReports para que se reduzisse o tempo necessário para a execução dos experimentos. Ademais, para que se aplicasse aos ambientes simulados uma carga próxima às existentes em infraestruturas reais, foram extraídos dados de *traces* reais de utilização de recursos disponibilizados pela multinacional Google por meio do projeto *Google Cluster Data* (REISS; WILKES; HELLERSTEIN, 2011). Por fim, a forma como o consumo energético de cada máquina foi modelado baseia-se

em *benchmarks* de máquinas reais disponibilizados pela *Standard Performance Evaluation Corporation*[1]. Esta seção descreve as modificações realizadas nas ferramentas de simulação, o processo de extração de dados para a geração de *workloads*, a modelagem de consumo energético e a criação dos cenários simulados.

4.1.1 CloudSim

O CloudSim é um *framework* de simulação cujo funcionamento se baseia em uma sequência discreta de eventos, que representam a interação entre entidades simuladas. Para representar tais entidades, o CloudSim faz uso da classe abstrata *SimEntity*. Portanto, o uso de eventos para a simulação de interações é feito por meio de implementações concretas dessa classe abstrata. Na versão padrão do *framework*, são oferecidas implementações de *SimEntity* para a representação de *datacenters* e de usuários da infraestrutura (modelados no formato de *brokers*)[2].

Para simular o mecanismo de controle centralizado apresentado na Seção 3.2, as implementações oferecidas pela versão padrão do *framework* são suficientes, pois oferecem uma versão global da infraestrutura através da classe *Datacenter* (implementação concreta de *SimEntity*). Entretanto, a simulação da estratégia de monitoramento distribuído proposta neste trabalho necessita que a interação entre os nós hospedeiros do *datacenter* também seja representada. Os nós componentes da infraestrutura são representados no CloudSim pela classe *Host*. Contudo, tal classe não é uma implementação concreta de *SimEntity* e, por conseguinte, não é possível que objetos do tipo *Host* disparem eventos e estabeleçam uma comunicação direta entre si. Para viabilizar essa comunicação, a classe *Host* foi modificada para se tornar uma implementação concreta de *SimEntity* e as seguintes subclasses foram criadas:

i. *MonitoringWorkerHost*: representa um nó hospedeiro para a simulação da estratégia de monitoramento distribuído. A classe implementa a troca periódica de informações entre uma lista de nós vizinhos e o envio de notificações à estação de controle.

ii. *NonMonitoringControllerHost*: representa a estação de controle para a simulação da estratégia de monitoramento distribuído. O nó controlador é

[1] http://www.spec.org/benchmarks.html

[2] Além de *datacenters* e *brokers*, versões recentes do *framework* oferecem subclasses de *SimEntity* que representam elementos de rede, como *switches*. Por ainda se encontrarem em um estágio inicial de desenvolvimento, tais implementações não foram empregadas nas simulações deste trabalho.

isento das resposabilidades de monitoramento e passa a receber e processar notificações de objetos do tipo *MonitoringWorkerHost*. A visão global do ambiente é mantida através de uma lista completa dos nós hospedeiros que compõem o *datacenter*.

4.1.2 CloudReports

A solução para a persistência de informações na ferramenta CloudReports consiste em uma integração nativa com um banco de dados SQLite. Embora ofereça vantagens como portabilidade e simplicidade de implementação, tal solução apresenta problemas sérios de performance quando a simulação de infraestruturas de grande porte se faz necessária. Como exemplo, ao simular 48 horas de operação de um ambiente composto por 1.000 nós computacionais e 5.000 máquinas virtuais, obteve-se um tempo total médio de execução de 55 horas e 37 minutos para cada repetição do experimento. Ao duplicar o tamanho do ambiente simulado, foi identificado um indício de que sua relação com o tempo total do experimento de simulação seja não-linear, posto que cada repetição do experimento passou a levar um tempo total médio de 134 horas e 16 minutos. Tendo em vista que são necessárias várias repetições para possibilitar uma análise estatística dos resultados e que este trabalho busca simular ambientes com até 10.000 nós computacionais, tornou-se clara a necessidade de buscar alternativas para otimizar a performance do simulador.

Com este objetivo em mente, foram analisadas alternativas de bancos de dados gratuitos e que oferececem a possibilidade de integração com o CloudReports sem demandar um alto custo de desenvolvimento. O PostgreSQL foi selecionado por ser uma opção de código aberto com boa documentação *online*, bons índices de performance e possuir *drivers* maduros para integração com o Hibernate, ferramenta para mapeamento de objetos relacionais (ORM) utilizada pelo CloudReports.

Após a integração com o PostgreSQL, o tempo total médio para a simulação de 48 horas de operação de um ambiente com 1.000 nós computacionais e 5.000 máquinas virtuais foi reduzido para 14 horas e 3 minutos, representando assim um ganho de 74,7% quando comparado ao tempo médio obtido com a solução que emprega SQLite.

4.1.3 *Workloads*

A Seção 2.2 discutiu através de referências bibliográficas algumas das principais questões a serem consideradas para a modelagem de *workloads* aplicadas a ambientes computacionais. Como tal modelagem exerce um papel determinante nos resultados dos experimentos de simulação, é necessário que se tenha um modelo de carga o mais próximo possível ao observado em ambientes reais. Para isso, foi realizada uma coleta de dados do projeto *Google Cluster Data*, o qual disponibiliza *traces* de utilização de recursos de um *cluster* real com aproximadamente 12.000 máquinas administrado pela multinacional Google.

A carga aplicada ao sistema simulado é caracterizada no CloudSim através de tarefas, representadas pela classe *Cloudlet*, a serem executadas nas máquinas virtuais alocadas nos nós hospedeiros. Por outro lado, os *traces* reais utilizados possuem dados relativos ao consumo de diversos recursos computacionais (e.g. CPU, memória e utilização de disco) e são apresentados na forma de *jobs* executados em máquinas específicas do *cluster* monitorado. Para viabilizar o uso das informações coletadas em *traces* reais nos experimentos de simulação, foi realizado um mapeamento onde as características dos *jobs* executados no *cluster* foram representadas por meio *cloudlets* executadas nos ambientes de simulação. Dessa forma, tornou-se possível simular ambientes com até 10.000 nós computacionais empregando-se uma carga com características similares às de uma infraestrutura real.

4.1.4 Modelo de consumo energético

Esta dissertação trata o consumo energético do *datacenter* como principal métrica de avaliação de desempenho da solução proposta. A Seção 2.2 discutiu trabalhos que abordam diretamente o processo de modelagem desse consumo em sistemas computacionais. Baseando-se no conhecimento extraído da literatura relacionada, as simulações fizeram uso de dados de consumo energético coletados de um *benchmark* de máquinas reais disponibilizado pela *Standard Performance Evaluation Corporation*.

Para que essas informações fossem integradas ao ambiente de simulação do CloudSim, uma nova classe que implementa a interface *PowerModel* foi desenvolvida com características de consumo específicas para a máquina de modelo Dell PowerEdge R820. Portanto, todos os *datacenters* simulados neste trabalho foram compostos por uma infraestrutura homogênea, ou seja, por máquinas de mesmo

modelo. Como os dados do *benchmark* fornecem valores de consumo em Watts (W) correspondentes a níveis discretos de carga no nó computacional, a criação do novo modelo de potência no CloudSim pôde ser realizada de forma direta, pois o *framework* já aborda o consumo energético baseando-se em valores percentuais de carga do nó simulado.

4.1.5 *Cenários simulados*

Os cenários de simulação criados através do CloudReports são compostos pela representação de um provedor de IaaS e um número arbitrário de clientes. O provedor de IaaS pode conter um ou mais *datacenters*, os quais são modelados de forma individual com características como políticas de alocação de máquinas virtuais, custos de operação e limiares de utilização de recursos. Ademais, é possível configurar individualmente cada um de seus nós computacionais. Os clientes são modelados por meio de um conjunto de máquinas virtuais a serem alocadas na infraestrutura gerenciada pelo provedor de IaaS e por um perfil de utilização. Cada máquina virtual pode ser configurada utilizando-se características como demanda de CPU e memória, tipo de *hypervisor* empregado e um escalonador de *cloudlets*. O perfil de utilização do cliente determina a forma como as *cloudlets* irão se comportar no que diz respeito à utilização de recursos enquanto são executadas, além de prover uma política de *brokering* através da qual é possível implementar regras para a seleção do *datacenter* que irá alocar uma determinada máquina virtual.

Como a proposta deste trabalho está situada no contexto de gerência de um único *datacenter*, todos os ambientes simulados possuem um provedor de IaaS modelado com uma infraestrutura única. Os nós componentes dessa infraestrura são modelados de acordo com características descritas nas Seções 4.1.1 e 4.1.4. Ademais, foram simulados quatro tipos distintos de políticas de alocação de máquinas virtuais. Tais políticas determinam como o nó controlador deve distribuir as máquinas virtuais entre os nós hospedeiros disponíveis e desempenham um papel de fundamental importância no consumo energético do *datacenter*. Portanto, para que se avaliasse o desempenho da estratégia de gerenciamento proposta independentemente da política de alocação empregada, as quatro alternativas abaixo foram simuladas:

i. *Single Static Threshold* (SST): possui um único limiar de utilização de recursos estático que determina quando um nó hospedeiro está sobrecarregado. A implementação utilizada nas simulações caracteriza um nó hospedeiro como

sobrecarregado se o mesmo possuir um nível de utilização de CPU ou memória superior a 90% de sua capacidade total. Neste caso, a estação de controle apenas distribui as máquinas virtuais quando um estado de sobrecarga é identificado. Portanto, não há ações de consolidação de alocação quando máquinas virtuais são migradas com o objetivo de diminuir a quantidade de nós hospedeiros sendo utilizados e, por conseguinte, economizar energia.

ii. *Double Static Threshold* (DST): possui dois limiares estáticos de utilização de recursos. Um limiar superior caracteriza o nó hospedeiro como sobrecarregado e um limiar inferior o caracterica como ocioso. A implementação utilizada nas simulações determina um estado de sobrecarga quando o nó hospedeiro está submetido a uma carga superior a 90% de utilização de CPU ou memória, enquanto um estado de ociosidade é determinado por um nível de utilização de recursos abaixo de 10%. Nesta política, a estação de controle distribui as máquinas virtuais sempre que um estado de sobrecarga é identificado, assim como também executa ações de consolidação, migrando máquinas virtuais de nós ociosos com o objetivo de deixá-los em modo *sleep* e diminuir o consumo de energia do *datacenter*.

iii. *Median Absolute Deviation-Minimum Migration Time* (MAD-MMT): política com limiar de utilização variável extraída da literatura relacionada (BELOGLAZOV; BUYYA, 2012). Utiliza uma medida de dispersão estatística (MAD) para a detecção de nós hospedeiros sobrecarregados através do ajuste dinâmico do limiar superior de utilização de recursos. Ademais, emprega uma técnica de tempo mínimo de migração (MMT) para selecionar máquinas virtuais a serem migradas. Para a detecção de nós hospedeiros ociosos, é utilizada uma abordagem simples, onde, após realizada a distribuição de máquinas virtuais, o nó hospedeiro com menor nível de carga é selecionado para que seja realizada a ação de consolidação.

iv. *Local Regression-Minimum Migration Time* (LR-MMT): assim como a política anterior, utiliza limiar de utilização variável e também foi extraída da literatura relacionada (BELOGLAZOV; BUYYA, 2012). Para a detecção de nós hospedeiros sobrecarregados e ajuste do limiar superior, emprega o método de regressão local (LR). A técnica de tempo mínimo de migração para a seleção de máquinas virtuais e a abordagem simples para a detecção de nós hospedeiros

ociosos utilizadas na política MAD-MMT também são utilizadas na política LR-MMT.

No que diz respeito às máquinas virtuais simuladas, foram empregados quatro tipos de configurações baseados em dados de seviços oferecidos por um provedor de IaaS real[3]. A Tabela 4.1 traz um detalhamento da capacidade de processamento e memória disponível em cada um dos tipos de instância utilizados nas simulações. Em todos os experimentos, uma quantidade equivalente de cada um dos tipos foi alocada na infraestrutura simulada.

Tabela 4.1: Tipos de instância das máquinas virtuais simuladas.

Tipo de instância	**CPU**	**RAM**
Extra-pequena	Um núcleo compartilhado de 1GHz	768MB
Pequena	Um núcleo de 1,6GHz	1,75GB
Média	Dois núcleos de 1,6GHz	3,5GB
Grande	Quatro núcleos de 1,6GHz	7GB

A análise de desempenho apresentada nesse capítulo emprega métricas e fatores que levam em consideração o consumo energético do *datacenter*, a qualidade do serviço oferecido pelo provedor de IaaS, a estratégia de DCIM adotada, a política de alocação de máquinas virtuais utilizada, a dimensão da infraestrutura e, para o caso da estratégia de gerenciamento proposta, a quantidade de nós vizinhos que cada nó hospedeiro deve monitorar. As seções seguintes abordam cada uma das métricas e fatores empregados durante a avaliação dos resultados.

4.1.6 Métricas

As métricas abaixo, extraídas dos experimentos de simulação, são utilizadas para a análise de desempenho da estratégia de DCIM proposta neste trabalho:

i. Consumo energético: medido em kilowatt (kilo=10^3, watt=unidade de energia), expressa a quantidade de potência consumida em determinado instante por todos os nós que compõem o *datacenter*. Leva em consideração

[3]Tipos de instância do Windows Azure: http://msdn.microsoft.com/en-us/library/windowsazure/dn197896.aspx

estritamente a potência consumida pelos nós computacionais. Portanto, gastos energéticos relacionados ao sistema de refrigeração e medidas como *Power Usage Effectiveness* (PUE) não são considerados durante a avaliação.

ii. *Tempo de Estabilização de Consumo* (TEC): o processo de *setup* dos experimentos realizados envolve a distribuição de todas as máquinas virtuais uniformemente entre os nós hospedeiros do *datacenter*. Como esse estado de alocação inicial não representa o cenário ideal sob o ponto de vista de consumo energético, as políticas de alocação que possuem técnicas de consolidação passam a agir de modo a alcançar um estado em que todas as máquinas virtuais sejam alocadas na menor quantidade possível de nós hospedeiros obedecendo-se aos critérios mínimos de qualidade de serviço adotados. O TEC consiste no intervalo de tempo levado desde o fim do processo de *setup* até o momento em que o consumo de energia do *datacenter* se estabiliza, dado que o número total de máquinas virtuais alocadas não se altera.

iii. *Número total de migrações*: migrações de máquinas virtuais têm impacto direto na qualidade de serviço oferecida pelo provedor de IaaS. Quanto menor a quantidade de migrações, maior o *uptime* dos serviços virtuais alocados nos nós hospedeiros. Portanto, boas estratégias de DCIM devem balancear a otimização da utilização de recursos com a menor quantidade possível de migrações.

iv. *Operação abaixo de limiares superiores*: manter a carga dos nós hospdeiros abaixo dos limiares superiores de utilização é uma métrica importante para que a qualidade do serviço entregue pelo provedor de IaaS seja garantida. Nós computacionais sobrecarregados tendem a apresentar tempos de resposta maiores, o que gera um impacto negativo direto no QoS observado pelo usuário final da infraestrutura.

4.1.7 Fatores

Para que a estratégia de DCIM proposta fosse validada sob diferentes aspectos e comparada com o modelo discutido na Seção 3.1, a análise de desempenho realizada emprega os seguintes fatores:

i. *Estratégia de DCIM*: em todas as avaliações realizadas, os resultados obtidos

para a estratégia de DCIM proposta são comparados com resultados obtidos para a estratégia de gerenciamento centralizado.

ii. *Política de alocação de máquinas virtuais*: são empregados os quatro tipos apresentados na Seção 4.1.5: *Single Static Threshold*, *Double Static Threshold*, *Median Absolute Deviation-Minimum Migration Time* e *Local Regression-Minimum Migration Time*.

iii. *Quantidade de nós computacionais*: para que o desempenho das estratégias simuladas sejam avaliados sob o ponto de vista de escalabilidade, a quantidade de nós computacionais que compõem o *datacenter* assume valores de $n = \{50, 100, 1.000, 5.000, 10.000\}$.

iv. *Relação quantidade de máquinas virtuais por nó hospedeiro*: a carga aplicada à infraestrutura como um todo é variada tendo como base a relação entre a quantidade de máquinas virtuais que necessitam ser alocadas com a quantidade de nós hospedeiros disponível. Tal relação assume os valores de $n = \{5, 10, 50, 100\}$ para um *datacenter* com 10.000 nós computacionais.

v. *Quantidade de vizinhos por nó hospedeiro*: de acordo com a hipótese *iii* levantada na Seção 1.3, a quantidade de nós vizinhos os quais cada nó hospedeiro deve monitorar pode ter um forte impacto no desempenho geral do sistema. Para que esse impacto seja analisado, o número de vizinhos a serem monitorados assume os valores de $n = \{5, 10, 100, 1.000, 5.000\}$ para um *datacenter* com 10.000 nós computacionais.

Tendo em vista os fatores *i*, *ii* e *iii* listados acima, observa-se uma combinação de duas estratégias de DCIM, quatro políticas de alocação de máquinas virtuais e cinco valores para quantidade de nós computacionais do *datacenter*, o que resulta em 40 ambientes distintos a serem avaliados. Ademais, o fator *iv* combina duas estratégias de DCIM, quatro políticas de alocação de máquinas virtuais e quatro valores para a relação VM/nó, resultando em 32 ambientes adicionais. Por fim, o fator *v* combina uma estratégia de DCIM, quatro políticas de alocação de máquinas virtuais e cinco valores para quantidade de vizinhos a serem monitorados, resultando em 20 ambientes a serem avaliados. Considerando que para cada ambiente são realizadas 30 repetições de experimentos de simulação para que seja aplicada uma análise estatística dos resultados, obtém-se uma quantidade total de experimentos equivalente a $(40 + 32 + 20) \times 30 = 2.760$.

Esta seção forneceu detalhes sobre a metodologia empregada para a criação dos ambientes de simulação e execução dos experimentos realizados. As seções seguintes exploram os resultados obtidos de acordo com as métricas listadas na Seção 4.1.6.

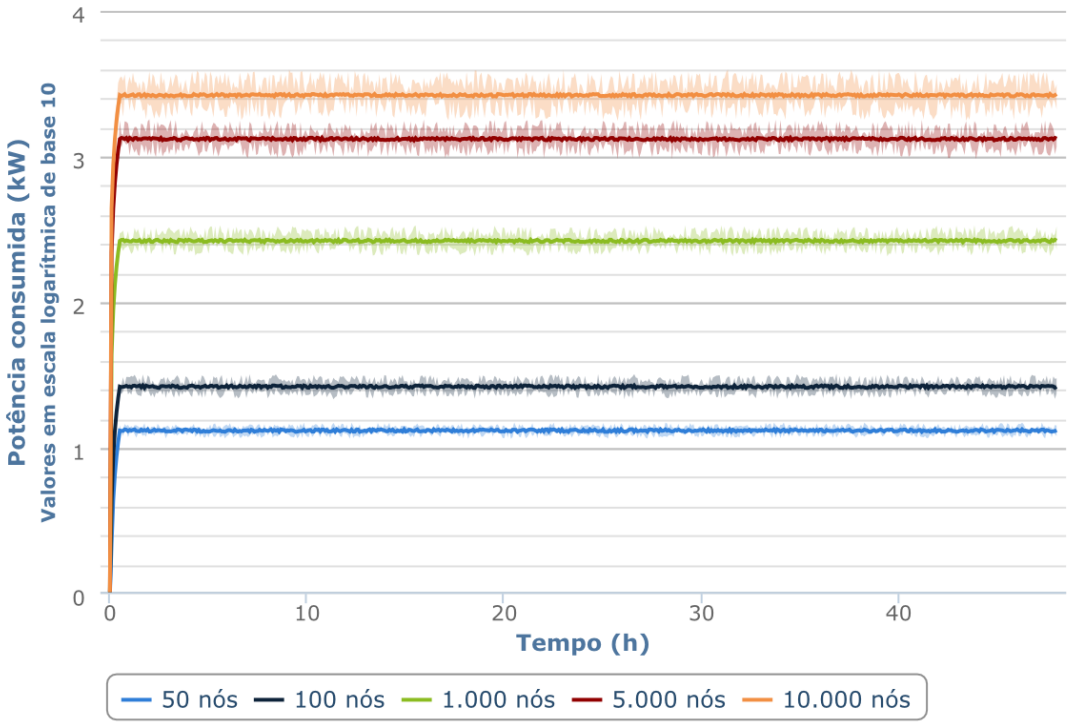
4.2 Eficiência energética

A Figura 4.1 mostra o consumo energético total do *datacenter* durante 48 horas de operação para a política de alocação *Single Static Threshold* (SST) de acordo com as estratégias de DCIM centralizada e distribuída[4]. Foram alocadas 10 máquinas virtuais por nó hospedeiro e, no caso da estratégia distribuída, cada um dos nós monitorou 10 nós vizinhos. Cada linha no gráfico é desenhada de acordo com o número total de nós hospedeiros do *datacenter* e representa a média amostral da métrica, enquanto as áreas sombreadas ao redor das curvas (visíveis na Figura 4.1 sobretudo no gráfico da estratégia centralizada) representam um intervalo de confiança com nível de 90%.

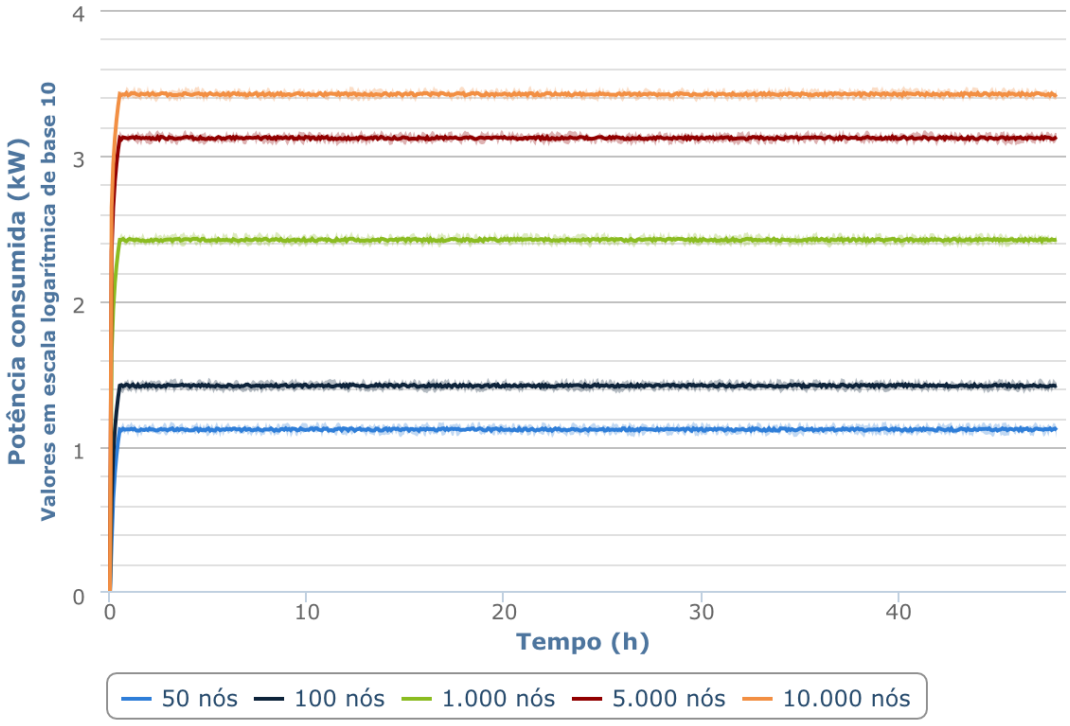
Como a política SST não possui fase de consolidação, apenas migrações com o objetivo de dissipar sobrecargas são realizadas no ambiente. Por consequência, percebe-se nos gráficos que a estabilização do consumo de energia ocorre logo após o período de *setup* (TEC = 0). Observa-se ainda que, para a política SST, os valores de consumo energético de ambas as estratégias de DCIM são muito próximos, independentemente do tamanho do *datacenter*. Isso ocorre sobretudo pelo fato de as ações de controle se limitarem a migrações de distribuição, o que não afeta sobremaneira o funcionamento do nó controlador em ambos os cenários. Contudo, a Seção 4.3 mostra que existe uma diferença sensível quando é realizada a mesma comparação sob o ponto de vista de qualidade de serviço. Outro ponto importante a ser ressaltado na Figura 4.1 é que, quanto maior o número de nós hospedeiros no *datacenter*, maiores são os intervalos de confiança observados nas curvas da estratégia centralizada, o que é resultado de uma maior variância amostral e indica um tempo de resposta mais elevado do nó controlador para a execução das ações de distribuição de carga. Tal fenômeno não acontece nas amostras referentes à estratégia de DCIM distribuída, onde os valores de variância permenecem baixos independetemente da dimensão do *datacenter*.

A Figura 4.2 mostra o consumo energético do *datacenter* para a política

[4]Durante a avaliação de resultados, os termos "centralizado" e "distribuído" se referem, respectivamente, ao modelo padrão de gerenciamento abordado na Seção 3.1 e à estratégia de gerenciamento proposta neste trabalho, descrita na Seção 3.2

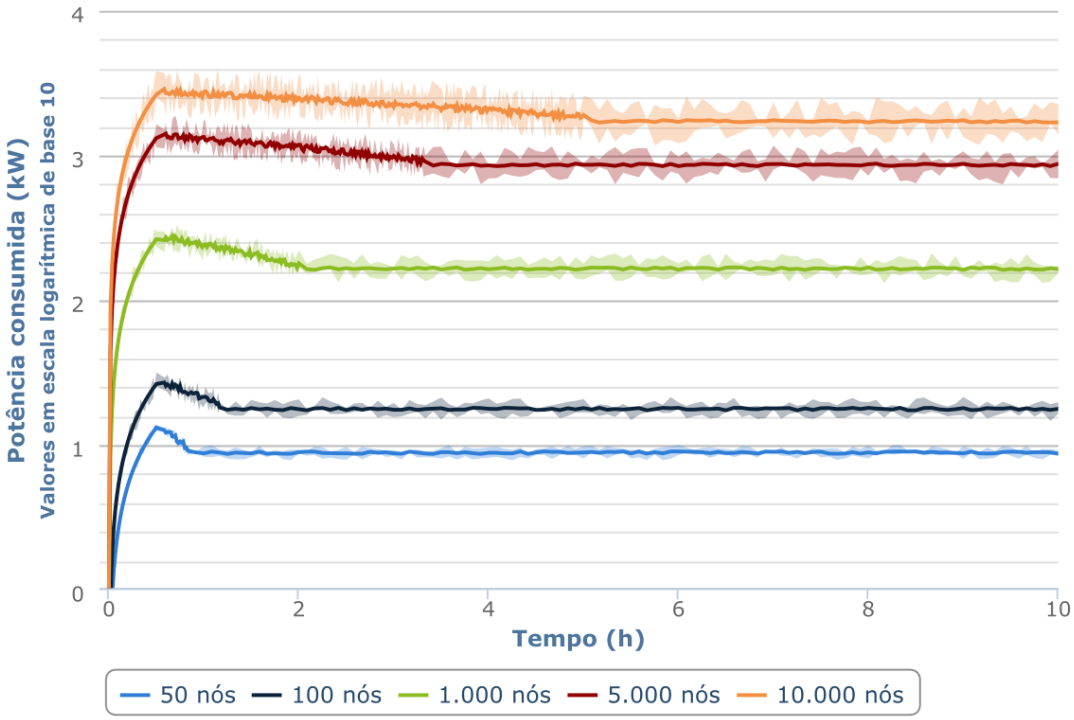


(a)

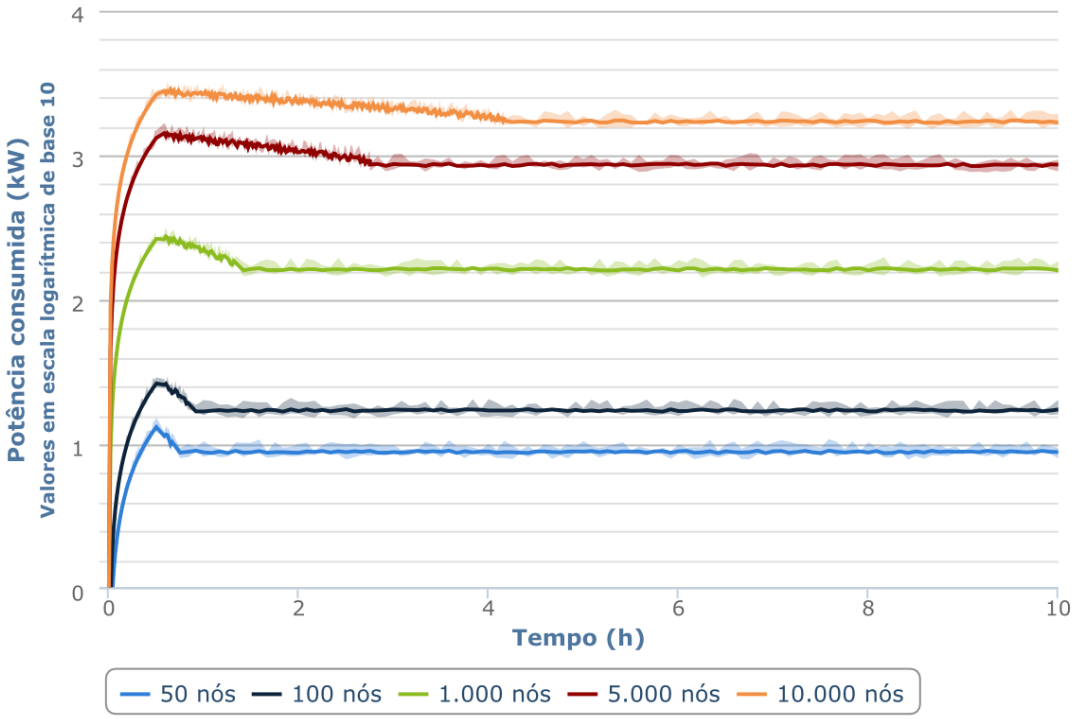


(b)

Figura 4.1: Consumo energético da política de alocação SST de acordo com a quantidade de nós hospedeiros do *datacenter* para estratégia de DCIM (a) centralizada e (b) distribuída.



(a)



(b)

Figura 4.2: Consumo energético da política de alocação DST de acordo com a quantidade de nós hospedeiros do *datacenter* para estratégia de DCIM (a) centralizada e (b) distribuída.

Double Static Threshold (DST) de acordo com as estratégias de DCIM e com a quantidade de nós hospedeiros do *datacenter*. Neste caso, como a política em questão aplica técnicas de consolidação, percebe-se que o TEC é maior que zero e varia proporcionalmente à dimensão da infraestrutura. Os gráficos mostram apenas as dez primeiras horas de operação do *datacenter* para que as diferenças entre os valores de TEC sejam realçadas, embora também tenham sido simuladas 48 horas de operação do *datacenter*. Observa-se que o período de *setup* da simulação é diretamente proporcional à taxa de máquinas virtuais alocadas por nó hospedeiro. Neste caso, manteve-se a taxa de 10 VM/Nó, resultando em um tempo médio de *setup* de 0,32 hora. Pode-se perceber nos gráficos que, para todos os valores de quantidade nós hospedeiros, a estratégia distribuída apresentou valores de TEC menores quando comparados à estratégia centralizada. Tal característica é importante, pois mostra o tempo de resposta da estratégia à mudanças de carga no ambiente até que um nível de consumo estável seja alcançado. A estratégia distribuída foi, em média, 26,1% mais rápida ao alcançar esse nível estável de consumo. Entretanto, ressalta-se que a métrica do TEC deve ser combinada a métricas de qualidade de serviço para que a performance de determinada solução possa ser avaliada de forma plena. Por fim, ainda pode-se perceber um ganho médio de consumo energético da solução distribuída de 6.8% quando comparada à estratégia centralizada. Como o monitoramento distribuído resulta em um tempo menor de resposta do nó controlador a variações de carga, as sobrecargas no ambiente são dissipadas mais rapidamente e as opções de consolidação são aplicadas de forma mais eficiente, o que mantém nos níveis de consumo abaixo dos observados em ambientes com monitoramento centralizado.

Os gráficos mostrados na Figura 4.3 seguem os mesmos parâmetros de simulação já discutidos para as Figuras 4.1 e 4.2 e comparam o consumo energético de duas políticas de alocação extraídas da literatura relacionada de acordo com a dimensão da infraestrutura e a estratégia de DCIM adotada. A política MAD-MMT apresenta consumo energético em média 3% inferior à política LR-MMT para ambas as estratégias de DCIM. Entretanto, os valores de TEC para a LR-MMT são inferiores, o que confirma os resultados apresentados no trabalho do qual ambas as políticas foram extraídas. (BELOGLAZOV; BUYYA, 2012). Ademais, os resultados mostrados na Figura 4.3 confirmam os dados obtidos para as políticas mais simples (SST e DST), a saber: *i.* a estratégia de DCIM centralizada apresenta uma variância amostral crescente de acordo com a dimensão da infraestrutura, enquanto

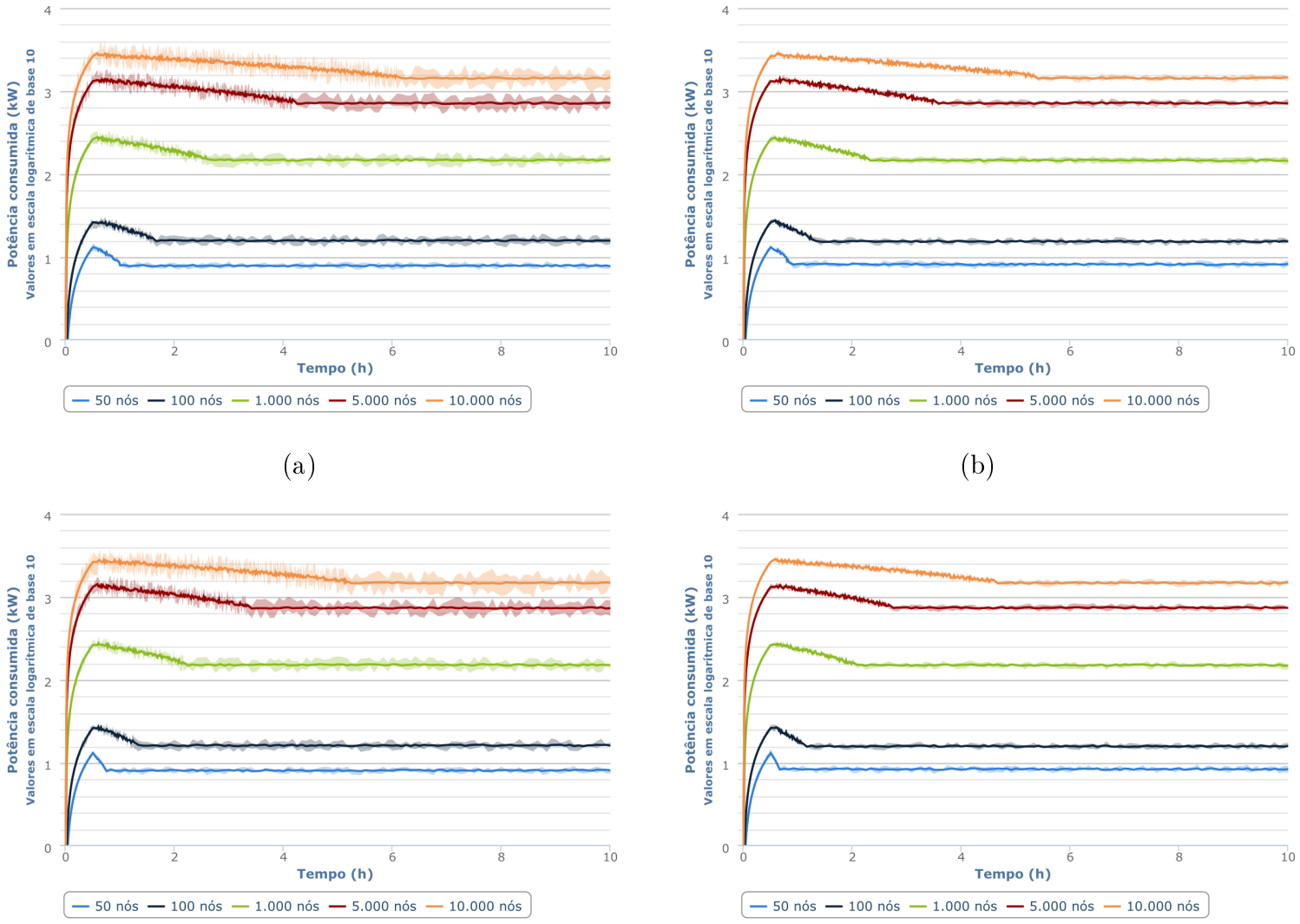
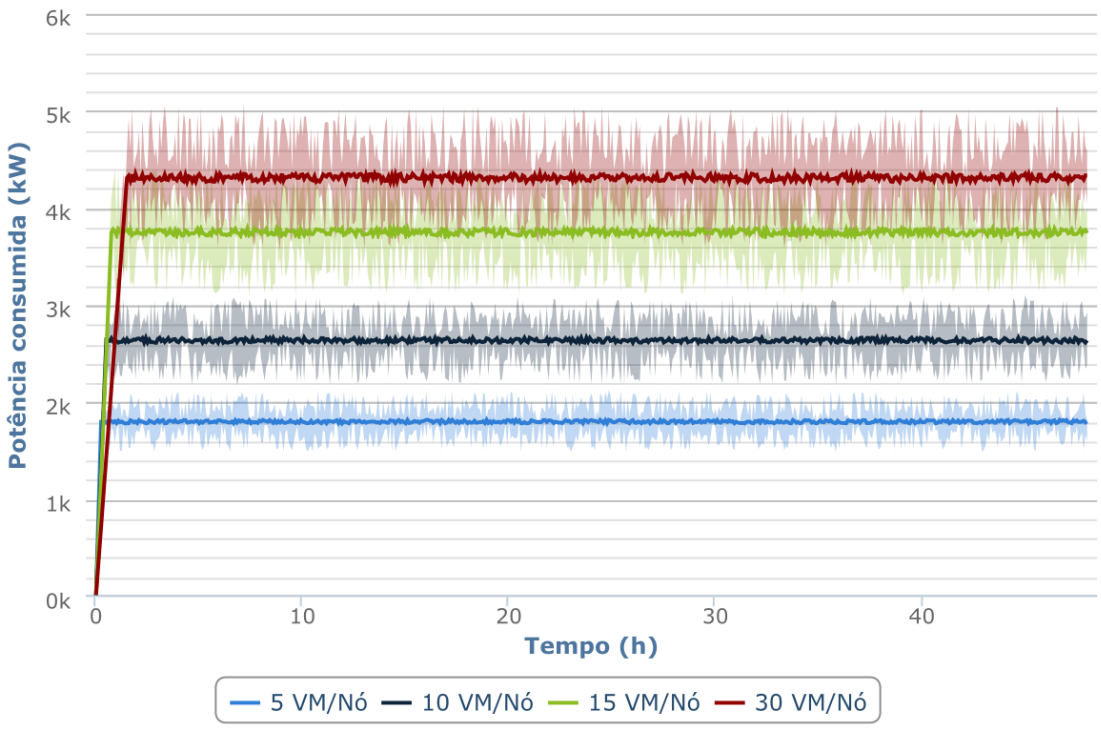


Figura 4.3: Consumo energético de acordo com a quantidade de nós hospedeiros do *datacenter* para as políticas de alocação (a) MAD-MMT centralizada, (b) MAD-MMT distribuída, (c) LR-MMT centralizada e (d) LR-MMT distribuída.

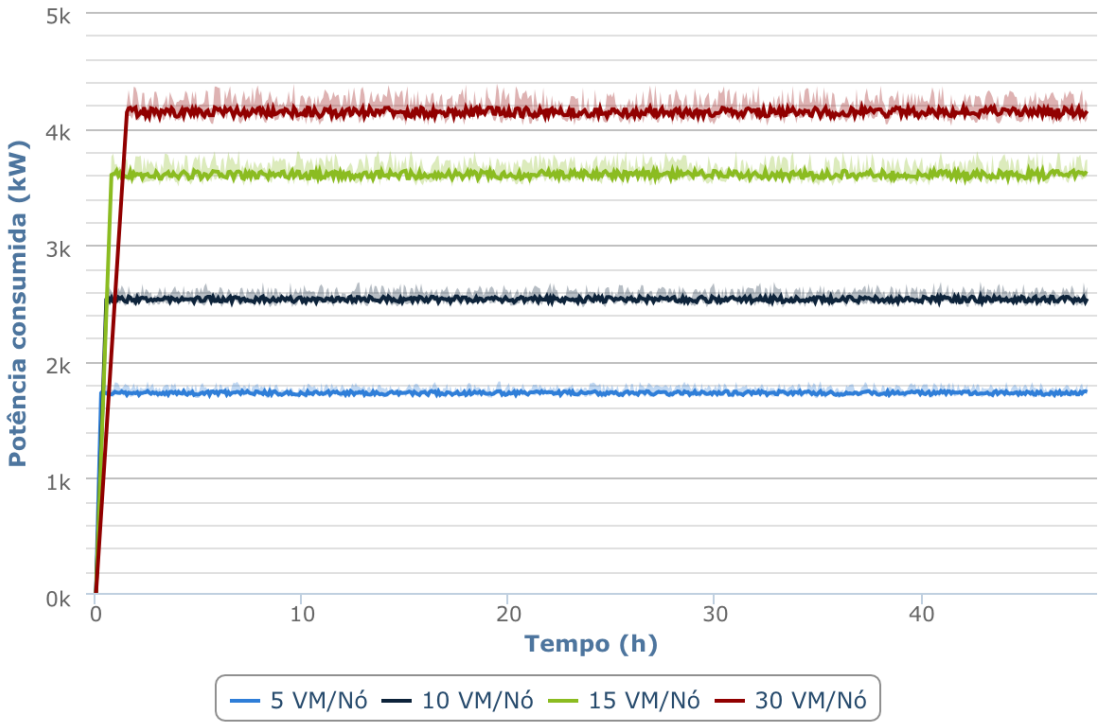
a estratégia distribuída mantém valores estáveis de variância independentemente do número de nós hospedeiros; *ii.* os valores de TEC obtidos para a estratégia de DCIM distribuída são menores para todas as políticas de alocação implementadas e todas as quantidades de nós hospedeiros avaliadas; *iii.* a estratégia de DCIM distribuída apresentou melhoras no consumo energético quando comparada à solução centralizada para todas as políticas de alocação que aplicam técnicas de consolidação (DST, MAD-MMT e LR-MMT).

Na Figura 4.4, é dada continuidade à análise do consumo energétco de acordo com a estratégia de DCIM, mas agora utilizando-se como fator a quantidade de máquinas virtuais alocadas por nó hospedeiro. Ambos os gráficos representam a simulação de um *datacenter* com 10.000 nós computacionais durante um período de 48 horas de operação. Como os valores são mostrados em escala linear, as áreas dos intervalos de confiança se tornam ainda mais visíveis para o gráfico da estratégia centralizada. Contudo, ao contrário dos resultados observados nas figuras anteriores, não ocorre um aumento significativo da variância amostral à medida que uma quantidade maior de máquinas virtuais é alocada no ambiente para ambas as estratégias de DCIM. O gráfico confirma a relação proporcional entre o tempo de *setup* do ambiente simulado e a taxa de máquinas virtuais alocadas por nó hospedeiro, pois pode-se observar uma variação entre 0.25h, para a taxa de 5VM/Nó, e 1.5h, para a taxa de 30VM/Nó. Ademais, torna-se claro que o consumo energético cresce proporcionalmente à quantidade de máquinas virtuais alocadas. Tal comportamento era esperado, pois quanto maior a carga aplicada ao ambiente, mais elevados os níveis de consumo de recursos, o que, por conseguinte, resulta em maiores índices de consumo energético total.

A Figura 4.5 mostra os resultados da simulação de um ambiente semelhante ao da Figura 4.4, mas aplicando-se a política de alocação DST. Neste caso, percebe-se um aumento significativo da variância amostral para ambas as estratégias de DCIM e todos os valores de taxa de máquina virtuais por nó. Contudo, a estratégia centralizada ainda apresenta valores de variância em média 67.4% maiores quando comparados aos obtidos na estratégia distribuída. Outra característica evidenciada nos gráficos da Figura 4.5 é a relação aproximadamente linear entre os valores de TEC e a taxa de máquinas virtuais alocadas por nó hospedeiro. Para o caso particular da taxa de 30VM/Nó, percebe-se que a política DST se comporta de forma semelhante à SST, o que leva a acreditar que a sua fase de consolidação não está

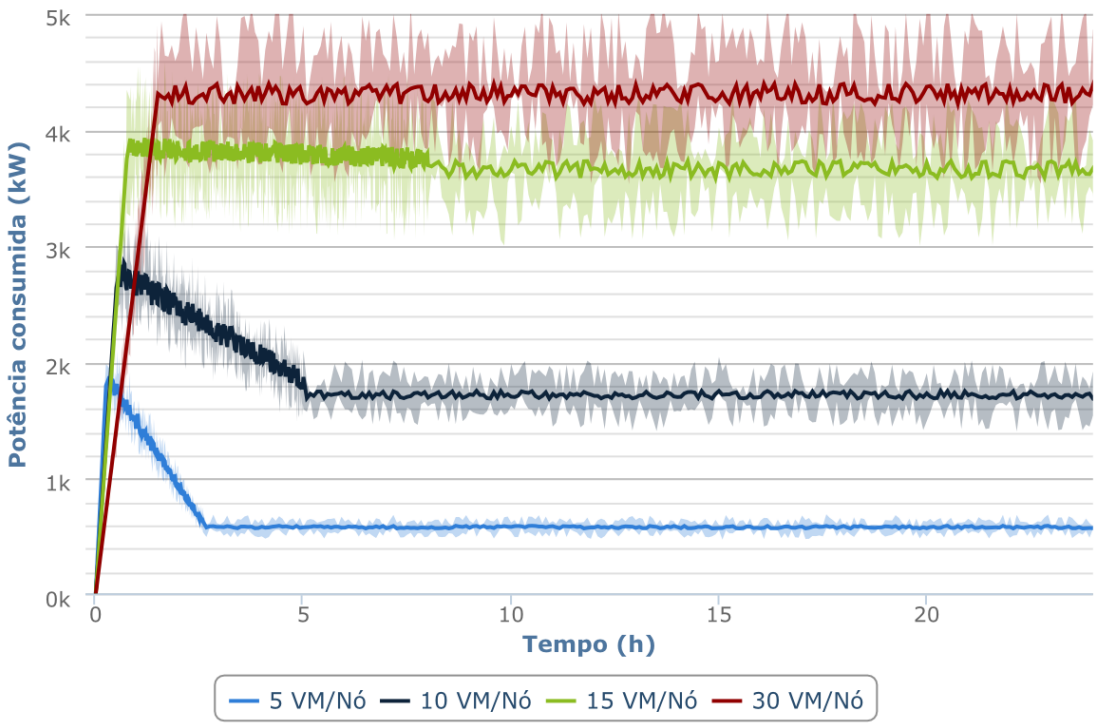


(a)

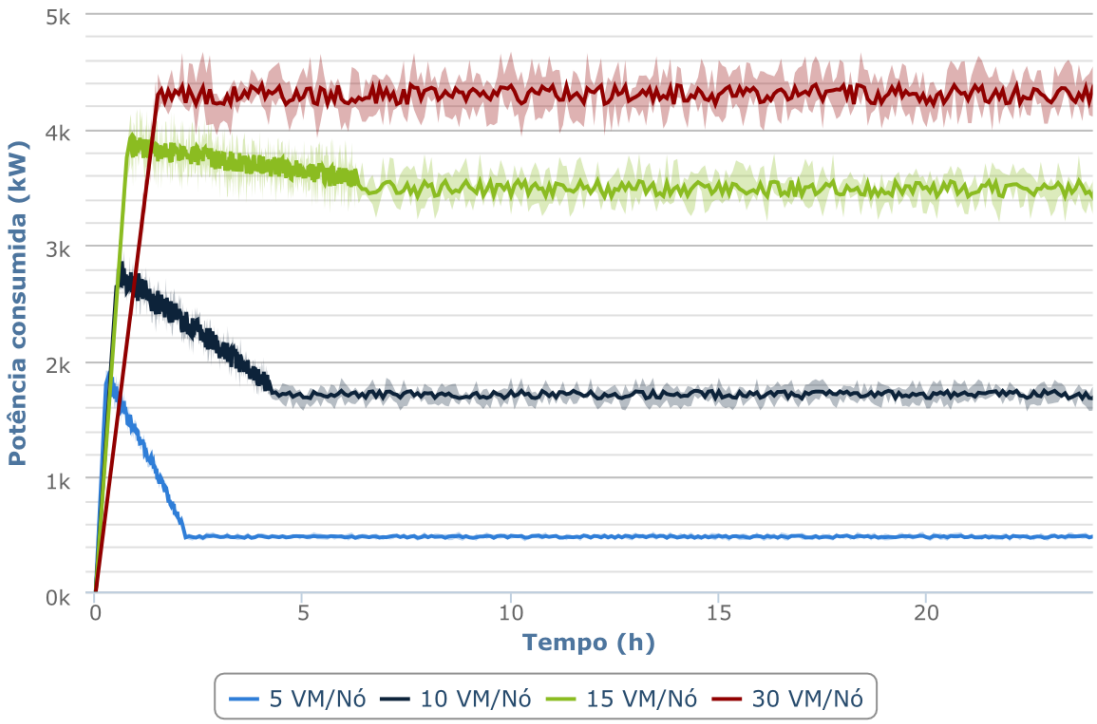


(b)

Figura 4.4: Consumo energético da política de alocação SST de acordo com a quantidade de máquinas virtuais alocadas por nó hospedeiro para estratégia de DCIM (a) centralizada e (b) distribuída.



(a)



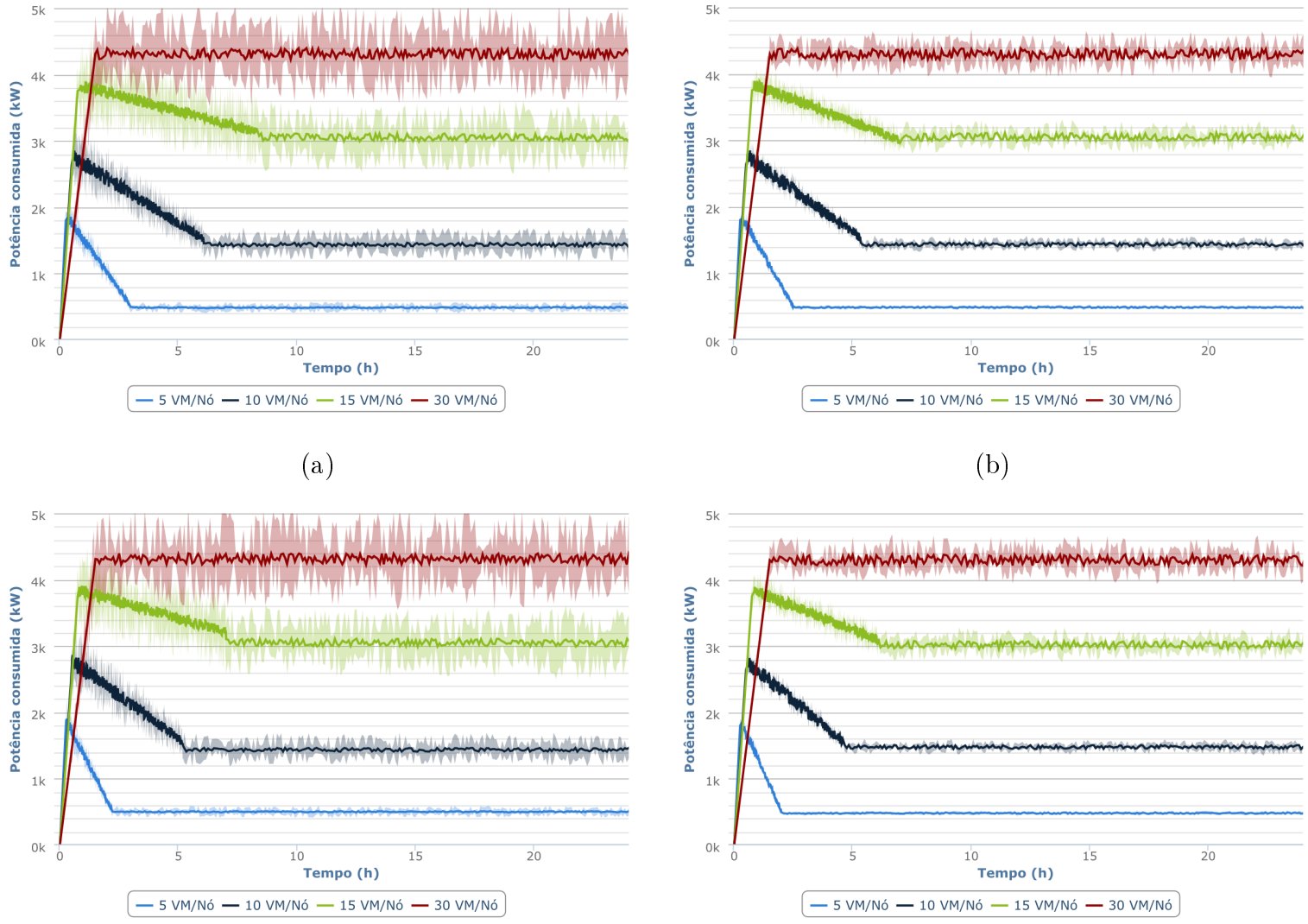
(b)

Figura 4.5: Consumo energético da política de alocação DST de acordo com a quantidade de máquinas virtuais alocadas por nó hospedeiro para estratégia de DCIM (a) centralizada e (b) distribuída.

sendo executada. Entretanto, tal comportamento deve-se ao fato de a quantidade total de máquinas virtuais alocadas no ambiente ser elevada ao ponto de manter sempre algum nó hospedeiro em estado de sobrecarga, o que faz com que a fase de consolidação da política de alocação não consiga diminuir o nível de consumo energético da infraestrutura (TEC $\rightarrow \infty$). Ademais, os resultados mostraram a redução média de 6.8% no consumo energético para a estratégia distribuída quando comparada aos valores obtidos para a estratégia centralizada, confirmando os resultados semelhantes já obtidos durante as simulações que empregaram a quantidade de nós hospedeiros como fator.

A Figura 4.6 complementa a avaliação do consumo energético total do *datacenter* de acordo com a quantidade de máquinas virtuais alocadas por nó hospedeiro para as políticas de alocação MAD-MMT e LR-MMT. Assim como na Figura 4.3, os resultados obtidos para ambas as políticas confirmaram as características fundamentais que puderam ser extraídas das Figuras 4.4 e 4.5: *i.* valores de variância amostral do consumo energético permanecem estáveis ao variar a quantidade de máquinas virtuais alocadas por nó para ambas as estratégia de DCIM; *ii.* a estratégia de DCIM centralizada apresentou valores maiores de variância amostral de consumo energético quando comparados aos obtidos para a estratégia distribuída; *iii.* existe uma relação aproximadamente linear entre os valores de TEC e a quantidade de máquinas virtuais alocadas por nó hospedeiro para todas as políticas de alocação simuladas; *iv.* houve uma redução média de aproximadamente 6.8% no consumo energético dos ambientes que empregaram a estratégia de DCIM distribuída quando comparado ao consumo médio observado para ambientes que empregaram a estratégia centralizada.

Para concluir a avaliação de consumo energético total do *datacenter*, foram realizadas simulações de 48 horas de operação de uma infraestrutura de 10.000 nós computacionais com uma taxa de alocação de 10VM/Nó e variando-se a quantidade de vizinhos a serem monitorados por cada nó hospedeiro para a estratégia de DCIM distribuída. A Figura 4.7 mostra o consumo energético dessa estratégia para as políticas de alocação DST, MAD-MMT e LR-MMT. Como a política SST não possui fase de consolidação e seu consumo energético é muito próximo para ambas as estratégias de DCIM, seu gráfico não é exibido. As linhas vermelhas indicam valores médios de consumo energético (horizontal) e TEC (vertical) para a estratégia de DCIM centralizada e servem como base de comparação para os resultados obtidos



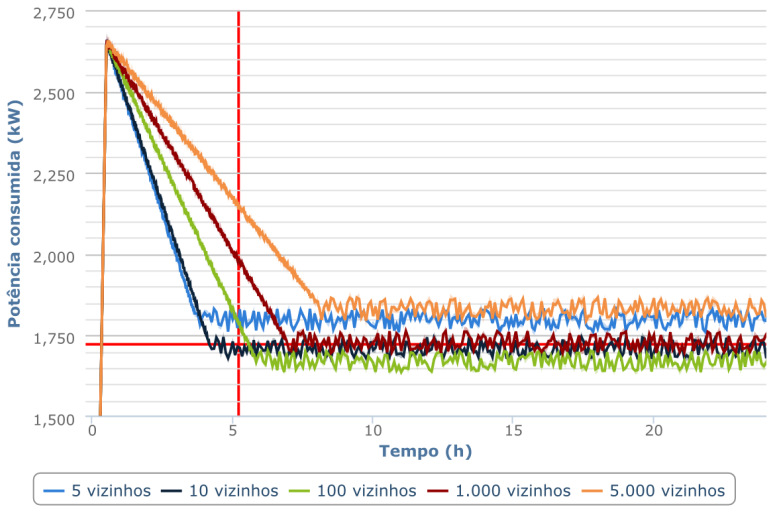
(a)

(b)

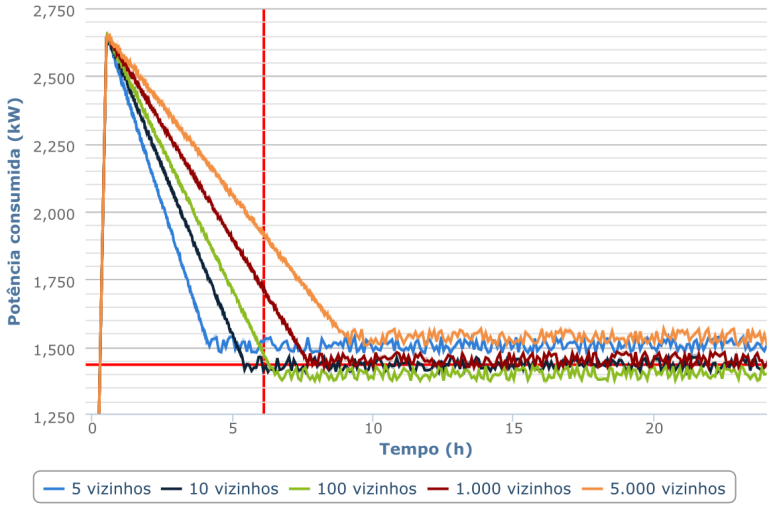
(c)

(d)

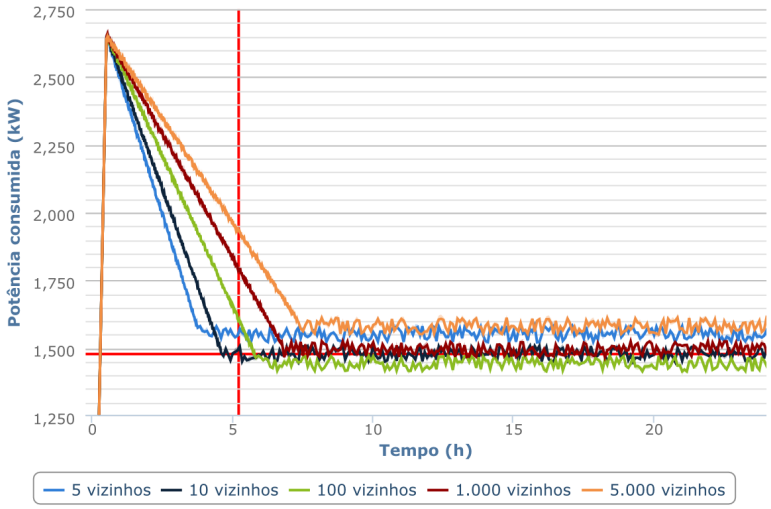
Figura 4.6: Consumo energético de acordo com a quantidade de máquinas virtuais alocadas por nó hospedeiro para as políticas de alocação (a) MAD-MMT centralizada, (b) MAD-MMT distribuída, (c) LR-MMT centralizada e (d) LR-MMT distribuída.



(a)



(b)



(c)

Figura 4.7: Consumo energético da estratégia de DCIM distribuída de acordo com a quantidade vizinhos monitorados por cada nó hospedeiro para as políticas de alocação (a) DST e (b) MAD-MMT e (c) LR-MMT.

para cada quantidade de nós vizinhos monitorados. Ademais, apenas as primeiras 24 horas de operação são exibidas para facilitar a visualização das diferenças entre as curvas. Por fim, em todos os gráficos é observado primeiramente o período de *setup* da simulação, representado pelo crescimento brusco de consumo por conta da alocação de todas as máquinas virtuais de forma uniforme na infraestrutura, seguido do TEC, quando as políticas de alocação aplicam suas regras de consolidação até que seja observada uma estabilização do nível de consumo energético do *datacenter*.

Como já havia sido constatado nos gráficos das figuras anteriores, o uso de uma taxa de 10 vizinhos monitorados por nó resulta em um consumo médio 6.8% menor que média obtida para a estratégia de DCIM centralizada em todas as políticas de alocação. Também pode ser observado em todas as políticas que essa taxa resulta em valores inferiores de TEC. Caso a taxa seja elevada a 100 vizinhos monitorados por nó, observa-se uma redução média de 2,8% no consumo energético quando comparado ao obtido com a taxa de 10 vizinhos por nó para todas as políticas de alocação. Entretanto, essa redução é acompanhada de um aumento médio de 28,7%, 16,6% e 22,1% nos valores do TEC para as políticas DST, MAD-MMT e LR-MMT, respectivamente. Portanto, percebe-se a existência de um *trade-off* entre consumo de energia e tempo de resposta da política de alocação ao definir a quantidade de nós vizinhos a serem monitorados por cada nó hospedeiro. A redução de energia deve-se ao fato de a estação de controle contar com uma quantidade maior de nós vizinhos para aplicar políticas de distribuição e consolidação. Isso significa que a quantidade de vezes que o nó controlador deve recorrer a um nó não vizinho para resolver problemas de alocação (vide Figura 3.4) é menor, o que eleva a eficiência da política de alocação. Por outro lado, os valores de TEC se elevam como consequência direta da maior quantidade de informações a serem processadas por cada nó hospedeiro antes que uma notificação seja enviada à estação de controle.

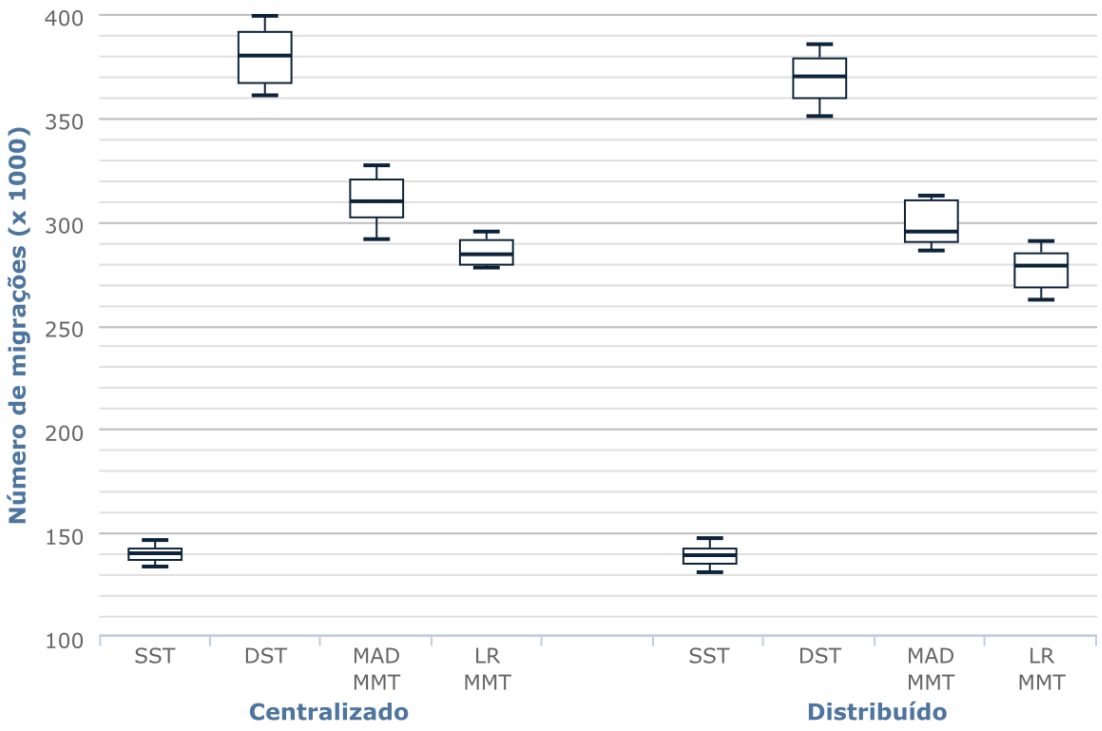
Ao elevar ainda mais o número de vizinhos monitorados para uma taxa de 1.000, constata-se que o consumo energético volta a subir e aproxima-se dos valores observados para a taxa de 10 vizinhos/nó. Nesse caso, o aumento do tempo de resposta das políticas de alocação causado pelo volume excessivo de informações a serem processadas por cada nó hospedeiro faz com que estados de sobrecarga se tornem cada vez mais constantes e tenham um impacto negativo no consumo energético. Tal cenário é agravado ainda mais quando se eleva a taxa a 5.000 vizinhos/nó, quando são observadas as maiores médias de consumo energético para

todas as políticas de alocação avaliadas. Outro ponto importante nos resultados observados na Figura 4.7 é o caso em que uma taxa de 5 vizinhos/nó é aplicada para a estratégia distribuída de DCIM. Como cada nó hospedeiro deve monitorar uma quantidade bastante reduzida de vizinhos, são observados os menores valores de TEC em todas as políticas de alocação. Entretanto, o número reduzido de vizinhos monitorados diminui sobremaneira a quantidade de vezes em que a estação de controle consegue resolver problemas de alocação sem recorrer a nós não vizinhos, o que tem um impacto negativo na eficiência das políticas de alocação. Ademais, como a técnicas de consolidação de máquinas virtuais são aplicadas apenas entre nós vizinhos, um valor muito baixo da taxa de vizinhos monitorados distribui demasiadamente a carga no datacenter e, por conseguinte, eleva desnecessariamente seu consumo energético total.

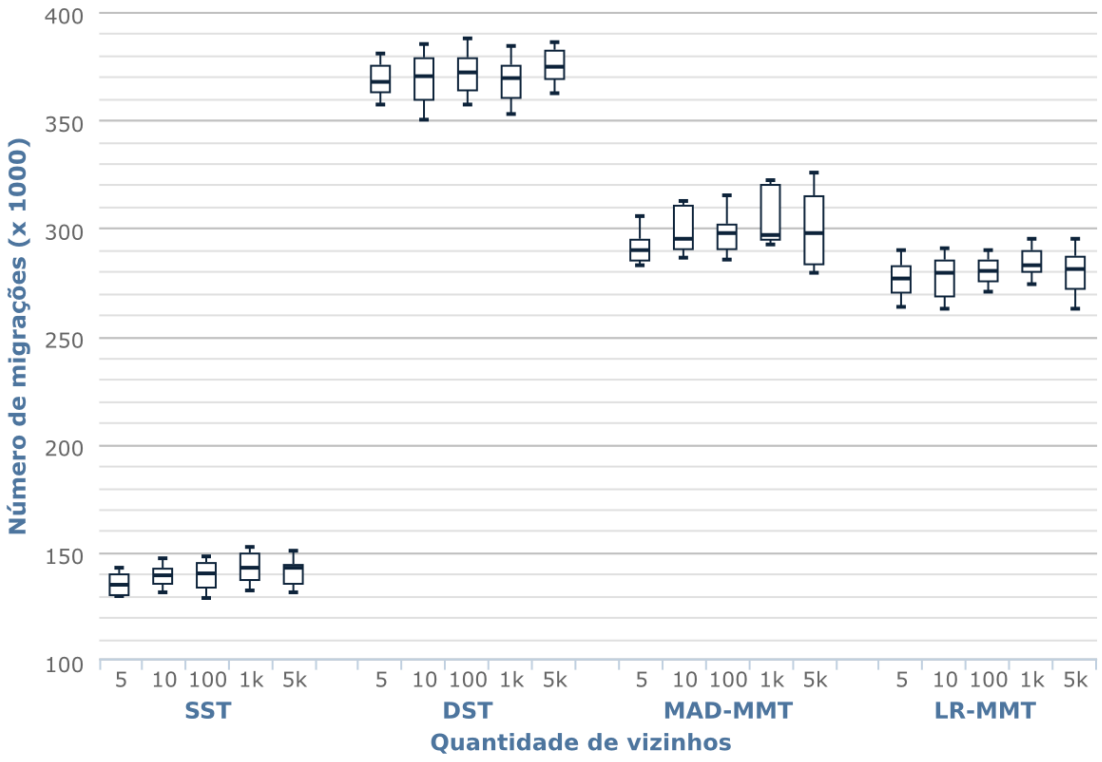
4.3 Qualidade de serviço

As métricas selecionadas para avaliar a estratégia de DCIM proposta neste trabalho são a quantidade total de migrações de máquinas virtuais executadas e o percentual médio de operação do *datacenter* abaixo dos limiares superiores de utilização de recursos estabelecidos pelas políticas de alocação utilizadas. Quanto maior a quantidade de migrações de máquinas virtuais, pior a qualidade de serviço oferecida, pois a migração tem impactos na performance e no *uptime* do serviço hospedado na infraestrutura. Por outro lado, quanto maior o percentual de operação do *datacenter* abaixo dos limiares superiores de utilização, melhor a qualidade de serviço oferecida, pois trata-se de um indicador de que o *datacenter* não operou com frequência em estado de sobrecarga.

A Figura 4.8 mostra a quantidade de migrações realizadas de acordo com a estratégia de DCIM empregada e com o número de vizinhos monitorados adotado para a estratégia de DCIM distribuída. Na Figura 4.8.a, observa-se que os menores índices de migração são apresentados pela política SST. O resultado é consequência da ausência de técnicas de consolidação, o que restringe as migrações a ações de distribuição de carga. Contudo, como observado durante toda a Seção 4.2, a ausência de migrações de consolidação tem um impacto negativo direto no consumo energético do *datacenter*. Dentre todas as outras políticas, o menor índice de migrações é apresentado pela LR-MMT, confirmando novamente os resultados obtidos no trabalho do qual a mesma foi extraída (BELOGLAZOV; BUYYA,



(a)

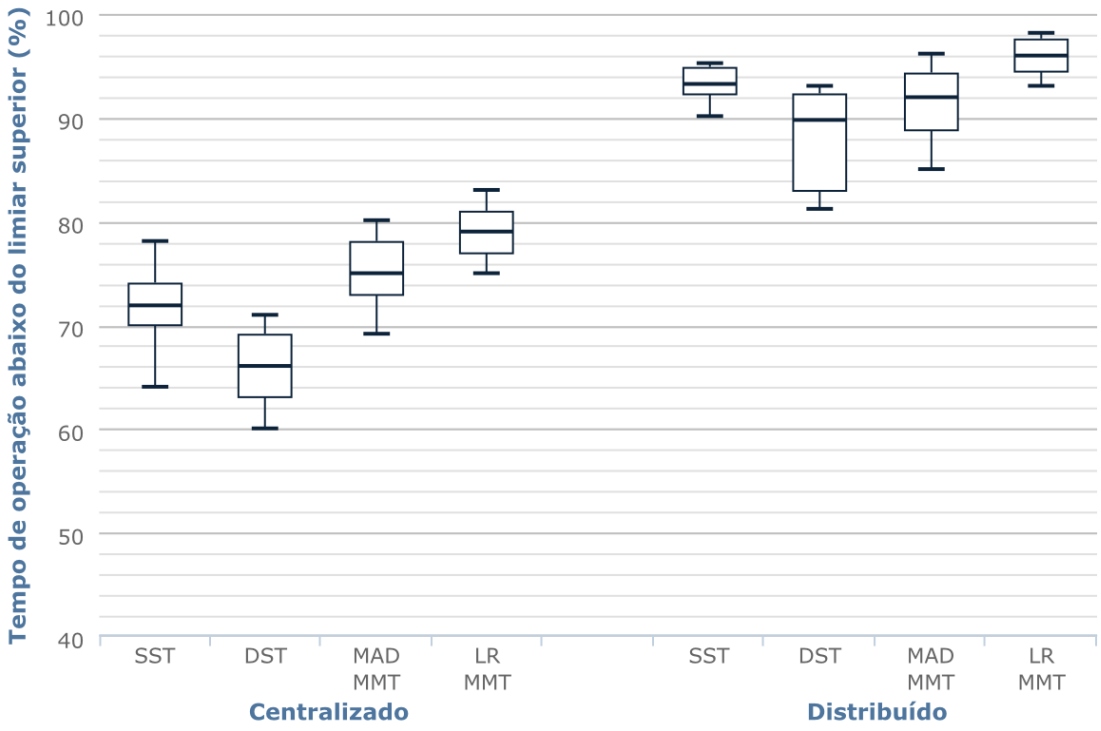


(b)

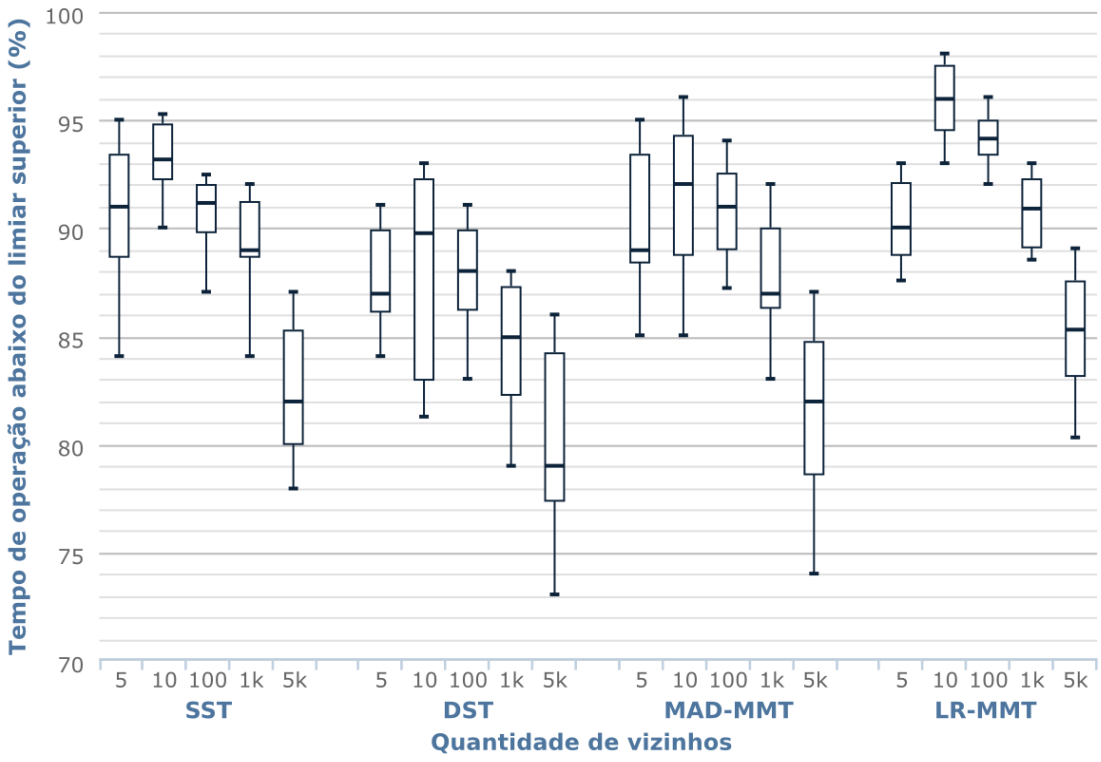
Figura 4.8: Quantidade de migrações de máquinas virtuais realizadas de acordo com (a) a estratégia de DCIM adotada e (b) a quantidade de vizinhos monitorados por cada nó hospedeiro.

2012). Ao comparar diretamente os resultados obtidos para ambas as estratégias de DCIM, vemos que a quantidade de migrações permenece muito próxima quando utilizada a mesma política de alocação. Portanto, quando combinado este fato com os resultados obtidos na seção anterior, pode-se concluir que a estratégia de DCIM distribuída consegue diminuir em média 6.8% do consumo energético com aproximadamente a mesma quantidade de migrações realizada pela estratégia centralizada. Por fim, a Figura 4.8.b mostra que a variação do número de nós monitorados na estratégia de DCIM distribuída não exerce uma influência significativa na quantidade total de migações realizadas, dado que os valores de mediana permanecem muito próximos independentemente da taxa adotada para cada uma das políticas de alocação.

Na Figura 4.9, os percentuais de operação do *datacenter* abaixo dos limiares superiores de utilização de recursos são mostrados de acordo com a estratégia de DCIM adotada e com a quantidade de vizinhos monitorados para o caso da estratégia de DCIM distribuída. Na Figura 4.9.a, a política SST mostra uma performance geral superior à DST e próxima à MAD-MMT. Contudo, já foi observado que, quando combinados com dados referentes a consumo energético, a política SST não apresenta bons resultados. Por outro lado, a política LR-MMT apresenta os maiores índices de operação abaixo dos limiares superiores para ambas as estratégias de DCIM. Ao comparar diretamente ambas as estratégias, pode-se observar um ganho claro de performance relacionado à alternativa distribuída para todas as políticas de alocação adotadas. Na Figura 4.9.b, podemos constatar muitas das observações realizadas a respeito dos gráficos da Figura 4.7. Por exemplo, pode-se perceber que, para qualquer política de alocação, existe uma relação de *trade-off* de consumo energético e QoS entre as taxas de 10 e 100 vizinhos monitorados por nó. Embora o consumo energético para 100 vizinhos/nó seja inferior quando comparado a 10 vizinhos/nó (vide Figura 4.7), a taxa de 10 viznhos/nó possui os melhores índices de operação abaixo dos limiares superiores. Confirma-se também que a elevação do consumo energético das taxas de 1.000 e 5.000 vizinhos/nó está diretamente relacionada a um percentual baixo de operação abaixo dos limiares superiores. Por fim, pode-se confirmar que o mau resultado de performance para consumo energético obtido para a taxa de 5 vizinhos/nó exibido na Figura 4.7 não está diretamente relacionado a sobrecargas, e sim a uma distribuição demasiada de carga resultante da limitação das ações de consolidação a uma quantidade muito reduzida de nós vizinhos.



(a)



(b)

Figura 4.9: Percentual de operação do *datacenter* abaixo dos limiares superiores de utilização de recursos de acordo com (a) a estratégia de DCIM adotada e (b) a quantidade de vizinhos monitorados por cada nó hospedeiro.

4.4 Resumo

Este capítulo apresentou a metodologia aplicada para a viabilização e criação dos ambientes de simulação e a avaliação dos resultados provenientes dos experimentos realizados. As Seções 4.1.1 e 4.1.2 discutiram as modificações realizadas nas ferramentas de simulação utilizadas para que fosse possível simular a estratégia de DCIM proposta neste trabalho. A Seção 4.1.3 discutiu a modelagem da carga aplicada aos ambientes, enquanto a Seção 4.1.4 discutiu como o consumo energético do *datacenter* foi modelado durante as simulações. Em seguida, as Seções 4.1.5, 4.1.6 e 4.1.7 apresentaram, respectivamente, os cenários simulados, as métricas adotadas e os fatores empregados durante a análise de desempenho realizada nas seções subsequentes. Por fim, os resultados dos experimentos foram avaliados sob o ponto de vista da eficiência energética do *datacenter* na Seção 4.2 e da qualidade de serviço na Seção 4.3.

Capítulo 5

Considerações finais

Este capítulo apresenta as principais contribuições alcançadas durante a execução deste trabalho (Seção 5.1) e elenca possibilidades de melhoria e perspectivas para trabalhos futuros diretamente relacionadas à estratégia de DCIM proposta (Seção 5.2).

5.1 Conclusões e principais contribuições

A principal contribuição deste trabalho consiste na modelagem e avaliação de uma estratégia de gerenciamento de infraestrutura de *datacenters* que combina um modelo de controle centralizado já disseminado na literatura relacionada a um modelo de monitoramento onde os nós hospedeiros do *datacenter*, antes passivos, adotam um papel ativo e passam a auxiliar o nó controlador na gerência da infraestrutura. Com base nos resultados discutidos nas Seções 4.2 e 4.3, as hipóteses levantadas na Seção 1.3 podem ser recapituladas com as seguintes conclusões: *i.* a distribuição do monitoramento resultou efetivamente em menores níveis de sobrecarga na infraestrutura; *ii.* tais níveis reduzidos de sobrecarga fizeram com que o tempo de resposta das ações de controle aplicadas pelo nó controlador fosse diminuído, o que resultou em impactos positivos em métricas de QoS e *iii.* foi confirmada uma relação entre a quantidade de nós vizinhos a serem monitorados por cada nó hospedeiro e métricas relacionadas a consumo energético e QoS.

No que diz respeito aos objetivos traçados na Seção 1.4, a distribuição da responsabilidade de monitoramento do *datacenter* foi modelada e posteriormente implementada em um ambiente de simulação. Para viabilizar essa implementação,

as Seções 4.1.1 e 4.1.2 apresentaram as modificações realizadas nas ferramentas de simulação utilizadas. Algumas dessas modificações representaram evoluções importantes para a ferramenta CloudReports e culminaram na publicação de um artigo no *Workshop* em *Clouds*, *Grids* e Aplicações (WCGA), que ocorreu durante o XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) (SÁ; SOARES; GOMES, 2011). Ademais, foram aplicadas quatro políticas de alocação de máquinas virtuais aos ambientes simulados, sendo duas de limiar estático simples e duas extraídas da literatura relacionada com limiares de utilização dinâmicos. Isso possibilitou uma avaliação isolada da performance oferecida pela estratégia de gerenciamento proposta independentemente da política de alocação adotada, o que ajudou a confirmar os ganhos provenientes da distribuição das ações de monitoramento no *datacenter*. Por fim, identificou-se uma relação direta entre a quantidade de vizinhos monitorados por cada nó controlador e métricas de consumo energético e QoS. Contudo, os dados resultantes dos experimentos realizados não foram suficientes para que se pudesse chegar a uma expressão analítica que descrevesse formalmente essa relação.

5.2 Limitações e perspectivas para trabalhos futuros

Durante a execução deste trabalho, foram identificadas algumas possibilidades de melhoria para que se pudesse extrair informações mais precisas e que fornecessem uma visão ainda mais completa dos resultados provenientes da aplicação da estratégia de DCIM proposta. Além disso, essa estratégia abre possibilidades para evoluções que podem servir como base para novos trabalhos de pesquisa. Os itens abaixo descrevem tais possibilidades de melhoria e perspectivas para trabalhos futuros:

i. A estratégia apresentada limitou-se ao gerenciamento da infraestrutura de um único datacenter. Tendo em vista que muitas corporações administram múltiplos *datacenters* e que seus usuários podem utilizar recursos disponíveis em infraestruturas localizadas em pontos geográficos distintos, pode-se analisar a possibilidade de aplicar as técnicas de gerenciamento apresentadas neste trabalho em múltiplos *datacenters* de forma integrada por meio de *brokers*, que atuariam como coordenadores de recursos entre diferentes infraestruturas;

ii. As implementações das políticas de alocação para a estratégia de DCIM

proposta aplicaram as regras de consolidação de carga apenas nos nós vizinhos de cada nó hospedeiro. Pode-se desenvolver técnicas para evoluir tal característica e consolidar a carga do *datacenter* entre todos os nós disponíveis, mantendo os ganhos de escalabilidade obtidos;

iii. Nos experimentos realizados, os nós hospedeiros se limitaram a enviar notificações de estados anômalos ao nó controlador para que o mesmo tomasse ações de distribuição ou consolidação de carga. Entretando, o modelo apresentado na Seção 3.2 suporta a notificação de eventos diversos à estação de controle. Como exemplo, pode-se notificar falhas em nós vizinhos, explorando a capacidade da estratégia de aumentar a tolerância da infraestrutura a tais tipos de evento.

iv. Como o nó controlador sempre recebe n notificações de um evento, dado que cada nó hospedeiro é monitorado por n nós vizinhos, tal redundância pode ser aplicada em regras de controle para reduzir a incidência de falsos positivos;

v. Nos experimentos realizados, todos os nós hospedeiros tinham seu conjunto de nós vizinhos estabelecido durante o período de *setup* das simulações. Tal conjunto não era mais modificado no decorrer do experimento. Pode-se avaliar estratégias de mudança dinâmica dos nós vizinhos aplicando-as aos experimentos para verificar possíveis melhorias na eficiência da estratégia de DCIM;

vi. Todos os experimentos realizados alocavam uma quantidade fixa de máquinas virtuais com o objetivo de analisar o Tempo de Estabilização de Consumo (TEC). Contudo, pode-se analisar a performance das estratégias avaliadas neste trabalho em ambientes onde a quantidade de máquinas virtuais alocadas muda de forma dinâmica;

vii. O processo de avaliação deste trabalho limitou-se à modelagem da proposta e a implementação de ambientes de simulação para a realização de experimentos. Pode-se criar uma implementação real da estratégia em um *middleware* de nuvem de código aberto (e.g. OpenNebula e Eucalyptus) para que se avalie sua performance em um *testbed* real.

Apêndice A

CloudReports

Este apêndice apresenta o CloudReports, uma ferramenta de simulação extensível voltada para ambientes computacionais em nuvem. O CloudReports permite a modelagem de múltiplas infraestruturas através de uma interface gráfica de fácil utilização. A ferramenta também oferece recursos como geração automática de relatórios, que compila resultados dos experimentos de simulação e os apresenta graficamente com alto nível de detalhes, e uma API que possibilita a criação de extensões, as quais são carregadas pelo simulador na forma de plugins com o uso da *Java Reflection* API. O CloudReports trata-se de um projeto de código aberto criado através de múltiplos módulos. As seções deste apêndice apresentam a *engine* de simulação empregada (Seção A.1), uma descrição da estrutura dos ambientes simulados (Seção A.2) e uma discussão detalhada sobre a arquitetura do *software* (Seção A.3).

A.1 CloudSim

O *framework* CloudSim (CALHEIROS *et al.*, 2011) é utilizado pelo CloudReports como sua *engine* de simulação. Trata-se de um conjunto de ferramentas para a modelagem e simulação nuvens computacionais. O CloudSim oferece abstrações para a rerpesentação de nós físicos, *datacenters*, máquinas virtuais e clientes de serviços em nuvem. Suas últimas versões também oferecem suporte à modelagem da estrutura de rede interna ao *datacenter* e do consumo energético de vários elementos de *hardware*. As abstrações oferecidas possibilitam sobretudo a simulação de componentes relacionados à camada de IaaS, mas podem ser extendidas

por usuários para dar suporte a aspectos de PaaS e SaaS.

Uma simulação é constituída pela interação entre provedores de IaaS (representados por *datacenters*) e usuários de serviços em nuvem (modelados na forma de *brokers*, que podem representar um ou mais usuários gerando requisições ao provedor). Tais usuários podem questionar os *datacenters* a respeito de seus recursos, requisitar a criação de máquinas virtuais e submeter requisições para a execução de aplicações (chamadas de *cloudlets* no CloudSim). Uma política de provisão (*provisioning policy*) decide como as máquinas virtuais requisitadas serão mapeadas para os nós físicos dos *datacenters*. Similarmente, decisões sobre como os recursos do nó hospedeiro são compartilhados entre suas máquinas virtuais e como os recursos atribuídos a uma VM são compartilhados entre as aplicações executadas pela mesma são definidas por políticas de escalonamento próprias. O CloudSim oferece por padrão algumas implementações dessas políticas, e usuários podem desenvolver e avaliar seus próprios algoritmos voltados para tais propósitos.

A modelagem da execução de aplicações é alcançada através do campo *length* do objeto *Cloudlet*, que representa a quantidade de instruções computacionais necessárias para que a *cloudlet* seja executada integralmente. Núcleos de CPU, os quais são características atribuídas aos nós físicos, têm sua capacidade de processamento expressa em instruções por segundo. Ambas as propriedades supracitadas são genéricas, de forma que nenhuma unidade de medida é atribuída à capacidade de processamento ou ao tamanho da *cloudlet*. Portanto, núcleos de processamento podem ser modelados de acordo com *benchmarks* de CPU conhecidos, como o SPEC CPU[1], ou podem assumir valores arbitrários definidos pelo usuário para representar uma capacidade computacional relativa dentre processadores diferentes e tempos de execução relativos dentre um conjunto de *cloudlets*. Quando uma *cloudlet* é atribuída a uma máquina virtual específica, uma estimativa do tempo total de execução é calculada baseada na quantidade de recursos alocados para a VM, na política de escalonamento aplicada e na quantidade de outras *cloudlets* sendo executadas simultaneamente. Uma vez que a estimativa é calculada, um evento interno é gerado na entidade *datacenter* e agendado para o tempo de fim de execução obtido. Quando tal evento é disparado, o estado de execução das *cloudlets* e o número de instruções já realizadas são atualizados. Caso todas as intruções já tenham sido executadas, a *cloudlet* é dada como finalizada. A cada rodada de

[1] http://www.spec.org/benchmarks.html

atualização, o tempo estimado para finalização das *cloudlets* também é recalculado e eventos de atualização são gerados, pois a quantidade de *cloudlets* na máquina virtual pode ter sido modificada e, por conseguinte, mais recursos podem ter se tornado disponíveis para outras *cloudlets*, o que reduz seu tempo estimado para finalização.

Além de abstrações para modelagem de entidades relacionadas à nuvem computacional, o CloudSim contém um núcleo de simulação de eventos discretos que coordena as interações entre provedores e usuários da infraestrutura. O núcleo recebe mensagens de entidades, controla o avanço do relógio de simulação e entrega mensagens para entidades de destino respeitando seus respectivos tempos de entrega. Nas primeiras versões do CloudSim, o núcleo de simulação era provido pela biblioteca SimJava (HOWELL; MCNAB, 1998). Contudo, a utilização dessa biblioteca impunha restrições à escalabilidade e performance do CloudSim pelo fato do SimJava ser baseado em *threads*. Na prática, três *threads* eram criadas para cada entidade: uma para um canal de entrada (para o recebimento de mensagens de outras entidades), uma para um canal de saída (para o envio de mensagens a outras entidades) e uma para a entidade em si (para controle de suas operações). Como *threads* são recursos escassos gerenciados pelo sistema operacional, há um limite para a quantidade máxima que pode ser executada em um dado momento. Indiretamente, tal característica limita a escalabilidade da simulação, pois o número de clientes e *datacenters* são restringidos pelo limite de *threads* sendo executadas. Ademais, a utilização de *threads* dá origem a ineficiências no processo de escalonamento do sistema operacional, pois, eventualmente, tempos de CPU serão fornecidos a *threads* que não possuem operações a serem executadas.

Para mitigar os fatores supracitados que limitavam a escalabilidade do CloudSim, foi desenvolvido um núcleo de simulação baseado em uma *thread* única que substituiu a biblioteca SimJava desde a versão 2.0 do CloudSim. Para manter a retrocompatibilidade com simulações escritas para versões anteriores, o novo núcleo implementa a mesma API que o SimJava e contém objetos equivalentes, acessíveis pelo código gerado pelo usuário. Portanto, as classes representativas de *datacenters* e clientes extendem a classe *SimEntity*, cuja manipulação de mensagens é controlada pelo núcleo de simulação. As mensagens são objetos do tipo *SimEvent* e contêm indicadores de destino, *tag*, tempo de envio e uma carga genérica, que é desempacotada e interpretada por cada entidade. O novo núcleo também adiciona

funcionalidades à *engine* de simulação como a possibilidade de definir termos para a filtragem de eventos baseada em suas características, como fonte, destino e tipo. Eventos filtrados podem ser manipulados de formas diferentes pelo sistema, caso necessário para atender a demandas específicas de usuários do CloudSim.

No fim de cada simulação, informações sobre tempo de execução de *cloudlets*, custos relacionados ao uso de recursos e outras informações definidas pelo usuário são disponibilizadas em objetos gerados durante o experimento. Os usuários são responsáveis pela escrita do código que extrai tais informações. Todavia, o único tipo de saída oferecido pelo CloudSim é a impressão de dados em um terminal de linha de comando. Similarmente, a única forma nativa oferecida pelo CloudSim para a escrita de simulações é através da criação do código Java correspondente. Portanto, se modos de visualização mais ricos ou método mais intuitivos para a representação dos ambientes de simulação forem necessários, os mesmos terão que ser escritos pelos usuários. Isto motivou a modelagem e o desenvolvimento do CloudReports, abordado nas próximas seções.

A.2 Ambientes de simulação

O CloudReports pode gerenciar um ou mais ambientes de simulação simultaneamente. Esses ambientes reproduzem a interação entre um provedor de IaaS e seus clientes. Como mostrado na Figura A.1, o provedor detém uma nuvem de recursos com uma quantidade arbitrária de *datacenters*, os quais são modelados de acordo com seu sistema operacional, arquitetura de processadores, *hypervisor* utilizado para virtualização, capacidade de rede disponível, custos de utilização e políticas de alocação de máquinas virtuais. Ademais, cada *datacenter* é composto por um número customizável de nós físicos (*hosts*), que são configurados de acordo com a quantidade de RAM , largura de banda disponível, capacidade de armazenamento, poder de processamento, escalonadores de máquinas virtuais e um modelo de consumo energético.

Os clientes são modelados por meio de um perfil de utilização de recursos e configurações relacionadas a suas máquinas virtuais, as quais serão alocadas nos nós hospedeiros localizados na infraestrutura do provedor de IaaS. O perfil de utilização de recursos descreve as aplicações dos clientes e uma política de alto nível que seleciona *datacenters* destino para a alocação de máquinas virtuais. Esta política é representada pela entidade de simulação denominada *broker*, que também define

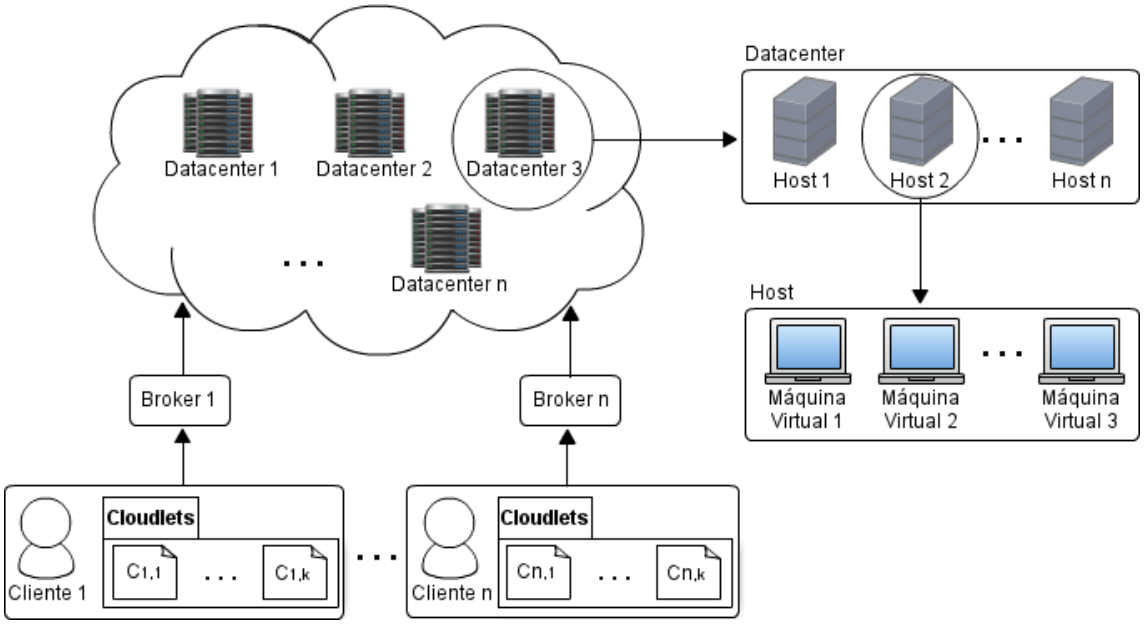


Figura A.1: Ambiente de simulação do CloudReports.

como as *cloudlets* serão gerenciadas e em qual VM serão alocadas. *Cloudlets* são modeladas utilizando-se características como quantidade necessária de núcleos de processamento, quantidade de instruções para finalização da execução, tamanhos de entrada e saída, que são utilizados para a simulação de sua tranferência entre clientes e provedores, e modelos de utilização para processamento, rede e memória. Por fim, uma configuração de máquina virtual inclui o tamanho de sua imagem, número de processadores, capacidade de processamento em milhões de instruções por segundo (MIPS), quantidade de RAM, largura de banda disponível, tipo de *hypervisor* e uma política de escalonamento de *cloudlets*.

A.3 Arquitetura do *software*

A arquitetura de software do CloudReports segue um *design* modular, como mostrado na Figura A.2. Em sua versão mais recente, existem cinco módulos fixos e um módulo de extensões opcional. As próximas seções descrevem em detalhes as funcionalidades de cada um desses módulos e suas formas de interação.

A.3.1 Núcleo de entidades

O núcleo de entidades define a estrutura básica de operação do CloudReports. Ele é formado por classes que representam entidades como cliente, *datacenters*, nós físicos, máquinas virtuais e rede. Embora pertença ao CloudReports, essas classes

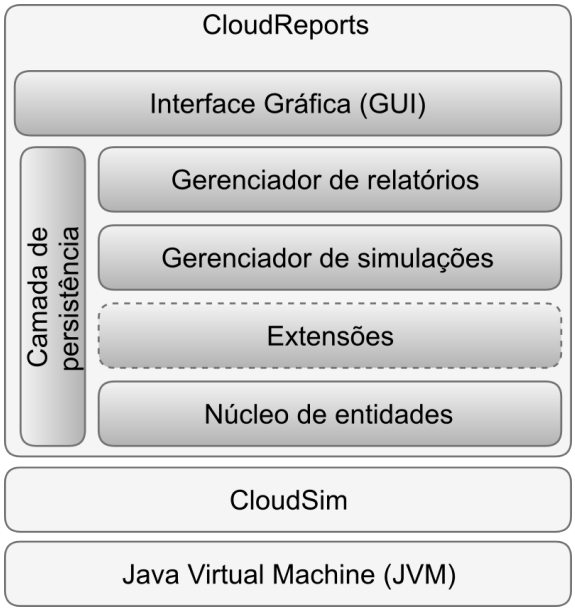


Figura A.2: Arquitetura de *software* modular do CloudReports.

trabalham em conjunto com o módulo de gerência de simulações para realizar a tradução dos ambientes criados por meio da interface gráfica para entidades do CloudSim, que são as únicas entidades utilizadas durante o processo de simulação. Portanto, o CloudReports trabalha como uma camada de abstração que auxilia usuários a manipular dados de experimentos de forma fácil, enquanto o CloudSim permanece como uma *engine* de simulação.

Algumas das entidades do núcleo do CloudReports auxiliam o gerenciamento de alguns eventos e aspectos relacionados ao sofware em si, como habilitação de migrações de máquinas virtuais, geração de números randômicos reais[2], dados de relatórios e configurações como quantidade de repetições de experimentos a serem realizadas e notificações por *e-mail*.

Para permitir que pesquisadores simulem suas próprias políticas, o CloudReports oferece uma API simples que consiste em um conjunto de enumerações, interfaces e um carregador de extensões. As enumerações classificam todos os tipos de extensões suportados pelo CloudReports, enquanto o carregador de extensões é responsável pela carga de todas as novas políticas em tempo de execução com o uso da *Java Reflection* API.

A Tabela A.1 lista todos os tipos de enumerações e mostra quais classes devem ser estendidas e quais interfaces devem ser implementadas para a criação

[2]Obtidos através do Quantum Random Bit Generator Service (http://random.irb.hr).

de uma extensão. A extensão *AllocationPolicy* estende a classe *VmAllocationPolicy* e implementa a interface *VmAllocationPolicyExtensible*. Ela determina como um *datacenter* irá alocar máquinas virtuais dentre seus nós hospedeiros. A extensão *BrokerPolicy* estende a classe *Broker* e descreve um conjunto de regras utilizadas por clientes para definir como as máquinas virtuais são enviadas para alocação e como as *cloudlets* são enviadas para execução considerando todos os *datacenters* disponíveis. Os tipos *BwProvisioner*, *RamProvisioner* e *PeProvisioner* estendem classes do CloudSim de mesmo nome e definem como nós hospedeiros provêm rede, RAM e elementos de processamento para as máquinas virtuais que alocam. Ademais, a extensão *VmScheduler* estende a classe *VmScheduler* do CloudSim e descreve como os nós hospedeiros escalonam a execução dessas máquinas virtuais. O *CloudletScheduler* estende a classe de mesmo nome do CloudSim e determina como as máquinas virtuais escalonam as *cloudlets* que executam. Por fim, a extensão *PowerModel* estende a classe *PowerModel* do CloudSim e possibilita a criação de novos modelos de consumo energético.

Tabela A.1: Lista de enumerações, classes e interfaces utilizadas para a criação de extensões do CloudReports.

Enumeração do CloudReports	**Extensões**	
	Deve estender	**Deve implementar**
AllocationPolicy	VmAllocationPolicy	VmAllocationPolicyExtensible
BrokerPolicy	Broker	-
BwProvisioner	BwProvisioner	-
RamProvisioner	RamProvisioner	-
PeProvisioner	PeProvisioner	-
VmScheduler	VmScheduler	-
CloudletScheduler	CloudletScheduler	-
UtilizationModels	-	UtilizationModel
PowerModel	-	PowerModel

A.3.2 Extensões

O módulo de extensões é inteiramente composto de código implementado por usuários. Embora sua existência seja facultativa, ele representa uma das principais funcionalidades do CloudReports, pois torna possível a criação de novas

políticas por pesquisadores sem que seja necessário modificar o código fonte do CloudSim ou do próprio CloudReports. Seguindo um pequeno conjunto de regras, pesquisadores podem adicionar novas políticas de alocação de máquinas virtuais, *brokers*, escalonadores de *cloudlets*, modelos de utilização de recursos, escalonadores de máquinas virtuais e modelos de provisão de elementos de processamento, RAM e rede. Ademais, este módulo também possibilita a criação de novos modelos de consumo energético. A versão 3.0 do CloudSim já oferece cerca de uma dúzia de implementações de modelos de consumo energético, incluindo opções baseadas em especificações reais de *hardware*. Pesquisadores podem estender modelos já existentes ou criar modelos inteiramente novos, desde que sigam as regras estabelecidas pela API de extensões oferecidada pelo CloudReports.

A.3.3 Gerenciador de simulações

O módulo de gerência de simulações consiste em dois elementos básicos: uma *factory* de entidades e um manipulador de simulações. A *factory* é responsável pela transformação de entidades do CloudReports em um cojunto de entidades do CloudSim, que serão então empregadas durante a execução dos experimentos. O manipulador de simulações obtém todas as configurações relacionadas à execução dos experimentos e inicia o processo de simulação. Após cada repetição, ele dispara a geração do respectivo relatório de simulação e dá início à próxima repetição. O módulo também é responsável pelo envio de notificações por *e-mail* e manipulação de erros durante os experimentos.

A.3.4 Camada de persistência

A camada de persistência armazena todos os dados da aplicação e dos experimentos em bancos do tipo SQLite ou PostgreSQL. Esta técnica facilita a gerência de múltiplos ambientes de simulação, pois cada banco pode ser utilizado de forma independente e manipulado externamente para fins de backup dos ambientes modelados. Ademais, como cada ambiente faz uso de um banco próprio, o crescimento demasiado das tabelas é evitado e o tempo de acesso aos dados é mantido em patamares razoavelmente baixos. Contudo, como os bancos SQLite não são apropriados para grandes volumes de dados, a simulação de ambientes de larga escala podem ter tempos de execução muito elevados. Tal característica motivou a implementação do suporte a bancos do tipo PostgreSQL, como descrito na Seção 4.1.2.

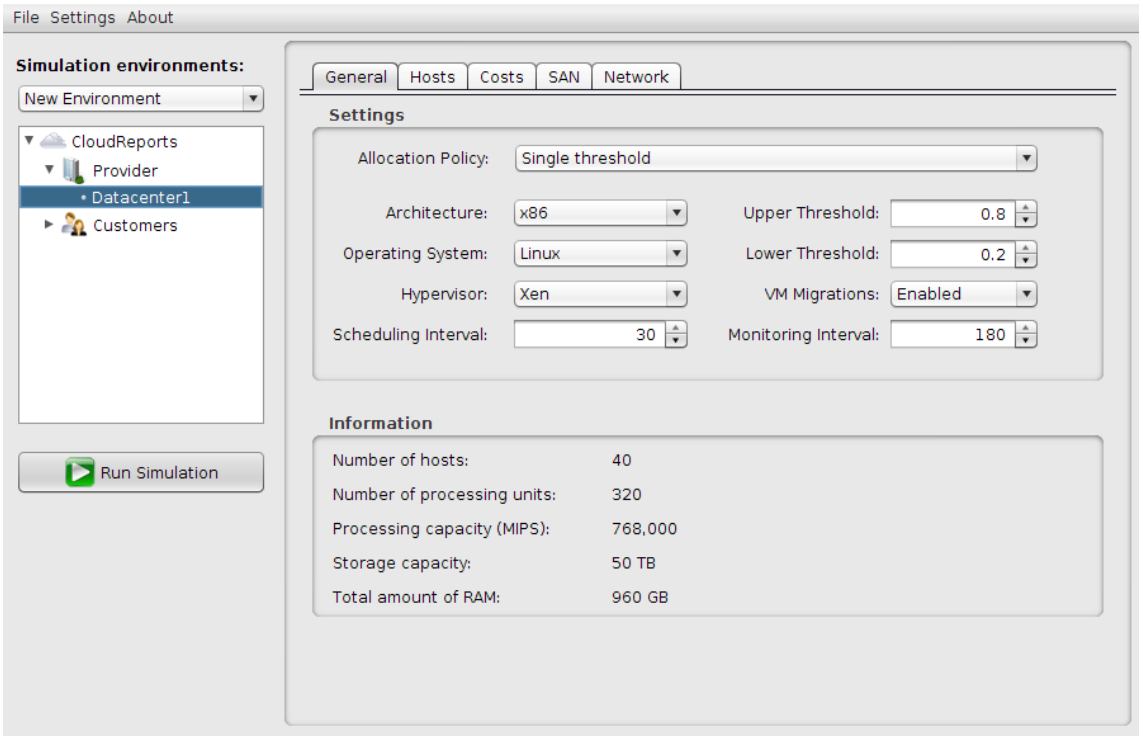


Figura A.3: Interface gráfica do CloudReports.

A.3.5 Gerenciador de relatórios

O gerenciador de relatórios coleta, organiza e processa dados de experimentos obtidos dos bancos de dados e realiza a criação de relatórios de simulação. Os relatórios são compostos de páginas HTML e arquivos de dados brutos. As páginas HTML possuem informações gerais sobre *datacenters* e clientes, incluindo, por exemplo, dados de consumo energético geral do *datacenter*. O gerador de relatórios utiliza todos os dados de simulação para criar gráficos automaticamente e incluí-los nas páginas HTML. Os arquivos de dados brutos possuem uma compilação de informações dos experimentos em um arquivo de texto único. Tais informações são disponibilizadas em um formato que possibilita sua importação por aplicações de terceiros, como MATLAB e Octave. Este módulo age sempre que o gerenciador de simulações dispara a geração de um relatório. Após finalizar a geração, o módulo notifica o gerenciador de simulações e a próxima repetição do experimento é iniciada.

A.3.6 Interface gráfica

Assumindo a posição mais elevada na pilha de módulos, a interface gráfica provê uma forma simples para que pesquisadores gerenciem ambientes de simulação

e acompanhem o progresso dos experimentos. A interface permite a criação e manipulação de *datacenters*, nós hospedeiros, clientes, máquinas virtuais e enlaces de rede. Ademais, pesquisadores podem configurar o custo de operação dos *datacenters*, modificar configurações da aplicação, selecionar políticas de escalonamento e provisão de recursos, determinar os modelos de utilização a serem empregados e indicar quais os ambientes a serem utilizados durante a execução dos experimentos. O módulo utiliza o conjunto de ferramentas *Swing* disponibilizado pelo *Java Development Kit* (JDK). Portanto, é independente de plataforma e possui um alto poder de customização. A Figura A.3 mostra um *screenshot* da interface gráfica do CloudReports.

Referências Bibliográficas

AKSANLI, B.; VENKATESH, J.; ROSING, T. Using datacenter simulation to evaluate green energy integration. *Computer*, v. 45, n. 9, p. 56 –64, sept. 2012. ISSN 0018-9162.

BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, v. 28, n. 5, p. 755 – 768, 2012. ISSN 0167-739X. Special Section: Energy efficiency in large-scale distributed systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X11000689>.

BELOGLAZOV, A.; BUYYA, R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, 2010.

BELOGLAZOV, A.; BUYYA, R. Energy efficient allocation of virtual machines in cloud data centers. In: *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*. [S.l.: s.n.], 2010. p. 577 –578.

BELOGLAZOV, A.; BUYYA, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, John Wiley Sons, Ltd, v. 24, n. 13, p. 1397–1420, 2012. ISSN 1532-0634. Disponível em: <http://dx.doi.org/10.1002/cpe.1867>.

BUYYA, R.; MURSHED, M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency*

and Computation: Practice and Experience, Wiley Press, v. 14, n. 13, p. 1175–1220, 2002.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In: *Proceedings of the International Conference on High Performance Computing & Simulation (HPC&S'09)*. Leipzig, Germany: IEEE Computer Society, 2009. p. 1–11.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, John Wiley Sons, Ltd., v. 41, n. 1, p. 23–50, 2011. ISSN 1097-024X. Disponível em: <http://dx.doi.org/10.1002/spe.995>.

CARLSON, M. Systems and virtualization management: Standards and the cloud (a report on svm 2010). *Journal of Network and Systems Management*, Plenum Press, New York, NY, USA, v. 19, n. 4, p. 536–542, dez. 2011. ISSN 1064-7570. Disponível em: <http://dx.doi.org/10.1007/s10922-011-9205-1>.

CARRERA, D.; STEINDER, M.; WHALLEY, I.; TORRES, J.; AYGUADE, E. Utility-based placement of dynamic web applications with fairness goals. In: *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*. [S.l.: s.n.], 2008. p. 9–16. ISSN 1542-1201.

CASANOVA, H.; LEGRAND, A.; QUINSON, M. SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In: *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*. Washington, DC, USA: IEEE Computer Society, 2008. (UKSIM '08), p. 126–131. ISBN 978-0-7695-3114-4. Disponível em: <http://dx.doi.org/10.1109/UKSIM.2008.28>.

CHAVES, S. A. D.; URIARTE, R. B.; WESTPHALL, C. B. Toward an architecture for monitoring private clouds. *IEEE Communications Magazine*, p. 130–137, 2011.

CHEN, X.; ZHANG, J.; LI, J. Resource management framework for collaborative computing systems over multiple virtual machines. *Service Oriented Computing and Applications*, 2011.

FAMAEY, J.; WAUTERS, T.; TURCK, F. D.; DHOEDT, B.; P.DEMEESTER. Towards efficient service placement and server selection for large-scale deployments. *In Advanced International Conference on Telecommunications*, p. 13–18, 2008.

FU, S. Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing*, 2010.

GALLARD, J. *et al.* Architecture for the next generation system management tools. *Future Generation Computer Systems*, 2012.

HOWELL, F.; MCNAB, R. SimJava: A discrete event simulation library for java. In: *Proceedings of the first International Conference on Web-Based Modeling and Simulation*. San Diego: SCS, 1998. p. 51–56.

IDC. U.s. datacenter 2012–2016 forecast. *Technical Report*, Setembro - 2012. Disponível para aquisição em http://www.idc.com/getdoc.jsp?containerId=237070. Último acesso em 10/07/2013.

ISLAM, S.; KEUNG, J.; LEE, K.; LIU, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 2011.

KIM, K. H.; BELOGLAZOV, A.; BUYYA, R. Power-aware provisioning of cloud resources for real-time services. In: *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*. New York, NY, USA: ACM, 2009. (MGC '09), p. 1:1–1:6. ISBN 978-1-60558-847-6. Disponível em: <http://doi.acm.org/10.1145/1657120.1657121>.

KLIAZOVICH, D.; BOUVRY, P.; AUDZEVICH, Y.; KHAN, S. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. In: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1 –5. ISSN 1930-529X.

KOCAOGLU, M.; MALAK, D.; AKAN, O. Fundamentals of green communications and computing: Modeling and simulation. *Computer*, v. 45, n. 9, p. 40 –46, sept. 2012. ISSN 0018-9162.

LEE, Y. C.; ZOMAYA, A. Energy conscious scheduling for distributed computing systems under different operating conditions. *Parallel and Distributed Systems, IEEE Transactions on*, v. 22, n. 8, p. 1374–1381, 2011. ISSN 1045-9219.

MOURATIDIS, K.; BAKIRAS, S.; PAPADIAS, D. Continuous monitoring of top-k queries over sliding windows. *In ACM SIGMOD International Conference on Management of Data*, p. 635–646, 2006.

NÚÑEZ, A. *et al.* icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, Springer Netherlands, v. 10, p. 185–209, 2012. ISSN 1570-7873. 10.1007/s10723-012-9208-5. Disponível em: <http://dx.doi.org/10.1007/s10723-012-9208-5>.

PRIETO, A. G.; STADLER, R. A-gap: An adaptive protocol for continuous network monitoring with accuracy objectives. *IEEE Transactions on Network and Service Management*, p. 4(1):2–12, 2007.

REISS, C.; WILKES, J.; HELLERSTEIN, J. L. *Google cluster-usage traces: format + schema*. Mountain View, CA, USA, Novembro - 2011. Revisado em 20/03/2012. Disponível em: <http://code.google.com/p/googleclusterdata/wiki/TraceVersion2>.

SHACHNAI, H.; TAMIR, T. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, Springer-Verlag, v. 29, n. 3, p. 442–467, 2001. ISSN 0178-4617. Disponível em: <http://dx.doi.org/10.1007/s004530010057>.

SOSINSKY, B. *Cloud Computing Bible*. 1ª. ed. [S.l.]: Wiley Publishing, 2011. ISBN 9780470903568.

SULISTIO, A.; CIBEJ, U.; VENUGOPAL, S.; ROBIC, B.; BUYYA, R. A toolkit for modelling and simulating data grids: an extension to gridsim. *Concurrency and Computation: Practice and Experience*, John Wiley and Sons Ltd., Chichester, UK, v. 20, n. 13, p. 1591–1609, set. 2008. ISSN 1532-0626. Disponível em: <http://dx.doi.org/10.1002/cpe.v20:13>.

SÁ, T. T.; SOARES, J. M.; GOMES, D. G. Cloudreports: uma ferramenta gráfica para a simulação de ambientes computacionais em nuvem baseada no framework cloudsim. *Workshop em Clouds, Grids e Aplicações (WCGA 2011)*, 2011.

TANG, C.; STEINDER, M.; SPREITZER, M.; PACIFICI, G. A scalable application placement controller for enterprise data centers. *In International Conference on World Wide Web*, p. 331–340, 2007.

WUHIB, F.; DAM, M.; STADLER, R.; CLEM, A. Robust monitoring of network-wide aggregates through gossiping. *IEEE Transactions on Network and Service Management*, p. 6(2):95–109, 2009.

WUHIB, F.; STADLER, R.; SPREITZER, M. Gossip-based resource management for cloud environments. In: *Network and Service Management (CNSM), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 1–8.

XU, L.; YANG, J. A management platform for eucalyptus-based iaas. *Cloud Computing and Intelligence Systems (CCIS)*, 2011.

YAN, S.; LEE, B. S.; ZHAO, G.; MA, D.; MOHAMED, P. Infrastructure management of hybrid cloud for enterprise users. *Systems and Virtualization Management (SVM)*, 2011.