Pós-Graduação em Ciência da Computação

Erica Teixeira Gomes de Sousa

MODELAGEM DE DESEMPENHO, DEPENDABILIDADE E CUSTO PARA O PLANEJAMENTO DE INFRAESTRUTURAS DE NUVENS PRIVADAS

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2015

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Erica Teixeira Gomes de Sousa

MODELAGEM DE DESEMPENHO, DEPENDABILIDADE E CUSTO PARA O PLANEJAMENTO DE INFRAESTRUTURAS DE NUVENS PRIVADAS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: *Prof. Dr. Paulo Romero Martins Maciel*

RECIFE
2015

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

S725m Sousa, Erica Teixeira Gomes de
Modelagem de desempenho, dependabilidade e custo para o planejamento de infraestruturas de nuvens privadas / Erica Teixeira Gomes de Sousa. – Recife: O Autor, 2015.
174 f.: Il., fig., tab., quadro

Orientador: Paulo Romero Martins Maciel.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2015.
Inclui referências.

1. Avaliação de desempenho. 2. Redes de computadores. 3. Sistemas distribuídos. 4. Modelagem de sistemas. I. Maciel, Paulo Romero Martins (orientador). II. Título.

004.029 CDD (23. ed.) UFPE- MEI 2015-48

Tese de Doutorado apresentada por **Erica Teixeira Gomes Sousa** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Modelagem de Desempenho, Dependabilidade e Custo para o Planejamento de Infraestruturas de Nuvens Privadas**" orientada pelo **Prof. Paulo Romero Martins Maciel** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Ricardo Massa Ferreira Lima
Centro de Informática/UFPE

Prof. Kelvin Lopes Dias
Centro de Informática / UFPE

Prof. Stênio Flávio de Lacerda Fernandes
Centro de Informática / UFPE

Prof. Artur Ziviani
Laboratório Nacional de Computação Científica

Prof. Luiz Carlos Pessoa Albini
Departamento de Informática / UFPR

Visto e permitida a impressão.
Recife, 26 de fevereiro de 2015.

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

A Deus.

A Elita Gomes, Elisabete Gomes e Felipe Sousa.

A Fernando Aires e a nossa bebê Elisa.

Agradecimentos

Gostaria de agradecer a todos que contribuíram para o desenvolvimento deste trabalho.

Agradeço ao professor Paulo Maciel, pela orientação, dedicação, apoio e comentários que foram essenciais para o desenvolvimento desse trabalho. Eu também gostaria de agradecer por todas as oportunidades de crescimento profissional e pessoal proporcionadas desde o início do mestrado.

Agradeço também aos professores Artur Ziviani, Kelvin Dias, Luiz Albini, Ricardo Massa e Stênio Fernandes que aceitaram o convite para compor essa banca de doutorado e contribuiram para a melhoria deste trabalho através de importantes conselhos e observações.

Agradeço a todos do grupo MoDCS pela amizade e parceria. Agradeço também a Jamilson Dantas e Rubens Matos pela colaboração. Em especial, eu agradeço a Eduardo Tavares, Edson Silva, Demetrio Silva e Erico Medeiros pela amizade e colaboração com esse trabalho.

Agradeço a Elisabete Gomes, minha mãe, pelo seu amor incondicional e suas preces. Agradeço a Elita Gomes, minha avó, e Felipe Sousa, meu irmão, pela paciência e amor. Em especial, agradeço a Elisabete Gomes pelo seu esforço em fornecer uma boa educação aos seus filhos.

Agradeço a Fernando Aires, meu noivo, pelo amor, compreensão, amizade, parceria e colaboração com este trabalho. Eu também agradeço por dividir os melhores momentos dessa caminhada e por ser *my sunshine* nos momentos mais difíceis.

Agradeço a amizade de Nívia Quental, Maria Aparecida Sibaldo, Bruno Nogueira, Kádna Camboim e Marcelo Marinho.

Agradeço a Enery Melo, Juliana Diniz e Sônia França pelo suporte com as informações sobre a Unidade de Educação à Distância da UFRPE.

Agradeço ao CNPQ pelo suporte financeiro para o desenvolvimento deste trabalho. Agradeço também ao CIn/UFPE pelo suporte com a infraestrutura necessária ao desenvolvimento deste trabalho.

Agradeço, principalmente, a Deus, que colocou várias oportunidades e todas essas pessoas no meu caminho.

Um passo à frente e você não está no mesmo lugar.

—CHICO SCIENCE

Resumo

Este trabalho apresenta uma solução integrada composta por uma metodologia, métodos, modelos de representação, modelos de otimização e uma ferramenta, para o planejamento de infraestruturas de nuvens privadas, considerando aspectos de desempenho, dependabilidade, performabilidade e custo. Os modelos de otimização propostos são baseados na metaheurística GRASP para geração de cenários de infraestruturas de nuvens privadas. O modelo para geração de cenários de desempenho e custo permite a criação de cenários de infraestruturas de nuvem com diferentes configurações de *software* e *hardware* e o modelo para geração de cenários de disponibilidade e custo possibilita a criação de cenários de infraestruturas de nuvem com diferentes mecanismos de redundância atribuídos aos componentes dessas infraestruturas. Os modelos de representação propostos são baseados em redes de *Petri* estocásticas, diagramas de bloco de confiabilidade e expressões matemáticas para avaliação de desempenho, de dependabilidade, de performabilidade e de custo de infraestruturas de computação em nuvem. O modelo de desempenho permite a avaliação do impacto da variação da carga de trabalho submetida a nuvem computacional com diferentes configurações de *software* e *hardware*. Esse modelo de desempenho possibilita o cálculo do tempo de resposta e da utilização de recursos das infraestruturas de nuvem computacional. Os modelos de disponibilidade possibilitam a avaliação do efeito da atribuição de diferentes mecanismos de redundância aos componentes da nuvem computacional. Esses modelos de disponibilidade permitem o cálculo da disponibilidade e do *downtime* das infraestruturas de computação em nuvem. Os modelos de custo possibilitam a avaliação dos custos da aquisição de equipamentos do sistema de TI, da aquisição de equipamentos e *softwares* redundantes, da substituição de equipamentos, da aquisição de licenças de *software*, da equipe técnica e da equipe de manutenção. As métricas de desempenho, de dependabilidade e de custo podem ser combinadas para seleção de infraestruturas de nuvem que atendam aos requisitos dos clientes. A ferramenta proposta possibilita a automação da solução integrada. Três estudos de caso são apresentados para aplicação da solução proposta. O primeiro estudo de caso apresenta a avaliação do impacto da variação da carga de trabalho atribuída a plataforma *Eucalyptus* com diferentes configurações de *software* e de *hardware*. O segundo estudo de caso apresenta a avaliação do efeito da atribuição de mecanismos de redundância aos componentes da plataforma *Eucalyptus*. O terceiro estudo de caso apresenta a avaliação do impacto da ocorrência de defeitos e atividades de reparo no desempenho da plataforma *Eucalyptus*.

Palavras-chave: Avaliação de Desempenho. Avaliação de Dependabilidade. Avaliação de Performabilidade. Avaliação de Custo. Modelo Estocástico. Rede de *Petri* Estocástica. Diagrama de Bloco de Confiabilidade. Metaheurística GRASP.

Abstract

This work presents an integrated solution composed of a methodology, methods, representation models, optimization models and a tool, for the planning of private cloud infrastructures, considering aspects of performance, dependability, performability and cost. The proposed optimization models are based on the GRASP metaheuristic for generating private clouds infrastructure scenarios. The model for performance and cost scenario generation allows the creation of cloud infrastructure scenarios with different software and hardware configurations and the model for availability and cost scenario generation permits the creation of cloud infrastructure scenarios with various redundancy mechanisms assigned to the components of these infrastructures. The proposed representation models are based on stochastic Petri nets, reliability block diagrams and mathematical expressions for performance, dependability, performability and cost evaluation of cloud infrastructures. Performance model allows the evaluation of the impact of varying the workload submitted to cloud computing with different software and hardware configurations. This performance model permits the calculation of the response time and resource utilization of the cloud infrastructures. Availability models allow the evaluation of the effect of assigning different redundancy mechanisms to the cloud components. The availability models permits the calculation of the availability and downtime of the cloud infrastructures. Cost models allow the evaluation of the acquisition of equipment of the IT system, acquisition of redundant equipment and software, equipment replacement, acquisition of software licenses, technical team and maintenance team costs. Performance, dependability and cost metrics can be combined to select cloud infrastructures that meet the client requirements. The proposed tool provides the automation of the integrated solution. Three case studies are presented to implement the proposed solution. The first case study presents the evaluation of the impact of the workload variation assigned to the Eucalyptus platform with different software and hardware configurations. The second case study presents the evaluation of the effect of the assigning redundancy mechanisms to the components of the Eucalyptus platform. The third case study presents the evaluation of the impact of the occurrence of fault and repair activities in the performance of the Eucalyptus platform.

Keywords: Performance Evaluation. Dependability Evaluation. Performability Evaluation. Cost Evaluation. Stochastic Model. Stochastic Petri Net. Reliability Block Diagram. GRASP Metaheuristic.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AG - *Genetic Algorithm*.

AMI - *Amazon Machine Image*.

API - *Application Programming Interface*.

AVA - Ambiente Virtual de Aprendizagem.

AWS - *Amazon Web Services*.

CAPEX - *Capital Expenses*.

CC - *Cluster Controller*.

CIn - Centro de Informática.

CLC - *Cloud Controller*.

CLS - *Candidate List*.

CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico.

CSV - *Comma Separated Values*.

CTMC - *Continuous-Time Markov Chain*.

DIP - *Direct IP Address*.

ECU - *Elastic Compute Unit*.

EC2 - *Elastic Compute Cloud*.

EMUSIM - *Emulation and Simulation Environment*.

ES - *Equipment Set*.

ET - *Equipment Type*.

EVMSA - *Efficient Virtual Machines Scheduling Algorithm*.

FaaS - *Framework as a Service*.

FT - *Fault Tree*.

GRASP - *Greed Randomized Search Procedure*.

GSPN - *Generalized Stochastic Petri Net*.

HaaS - *Hardware as a Service*.

HS - *Hardware Set*.

HSN - *Hardware Set Number*.

HTTP - *HyperText Transfer Protocol*.

IaaS - *Infrastructure as a Service*.

IBM - *International Business Machines*.

ID - *Identifier*.

IES - Instituições de Ensino.

INA - *Integrated Net Analyzer*.

IP - *Internet Protocol*.

ISP - *Internet Service Provider*.

KVM - *Kernel-based Virtual Machine*.

kW - *kilowatt*.

LC - Lista de Candidatos.

LRC - Lista Restrita de Candidatos.

LVS - *Linux Virtual Server*.

MC - *Markov Chain*.

MEC - Ministério da Educação.

MNET - *Maximum Number of Equipment Type*.

MNHS - *Maximum Number of Hardware Set*.

MNSS - *Maximum Number of Software Set*.

MoDCS - *Modeling of Distributed and Concurrent Systems*.

Moodle - *Modular Object-Oriented Dynamic Learning Environment*.

MNRT - *Maximum Number of Redundancy Type*.

MTA - *Mean Time to Active*.

MTBF - *Mean Time Between Failures*.

MTTF - *Mean Time to Failure*.

MTTR - *Mean Time to Repair*.

NC - *Node Controller*.

NIST - *National Institute of Standards and Technology*.

NLB - *Network Load Balancing*.

NYT - *New York Times*.

OPEX - *Operating Expenses*.

OVF - *Open Virtualization Format*.

PaaS - *Platform as a Service*.

PDF - *Portable Document Format*.

PN - *Petri Net*.

QEMU - *Quick EMUlator*.

RBD - *Reliability Block Diagram*.

RS - *Redundancy Set*.

RT - *Redundancy Type*.

RHEL - *Red Hat Enterprise Linux*.

SA - *Simulated Annealing*.

SaaS - *Software as a Service*.

SC - *Storage Controller*.

SLA - *Service Level Agreement*.

SMG4PCIP - *Scenarios and Models Generator for Private Cloud Infrastructure Planning*.

SPN - *Stochastic Petri Net*.

SRN - *Stochastic Reward Net*.

SS - *Software Set*.

SSN - *Software Set Number*.

S3 - *Simple Storage Service*.

TI - Tecnologia da Informação.

TS - *Tabu Search*.

UFPE - Universidade Federal de Pernambuco.

UFRPE - Universidade Federal Rural de Pernambuco.

UML - *User-Mode Linux*.

UML - *Unified Modeling Language*.

VDC - *Virtual Data Center*.

VDI - *Virtual Desktop Infrastructure*.

VHD - *Virtual Hard Disk*.

VIP - *Virtual IP Address*.

VLAN - *Virtual Local Area Network*.

VM - *Virtual Machine*.

VMDK - *Virtual Machine Disk Format*.

VMM - *Virtual Machine Monitor*.

XML - *eXtensible Markup Language*.

WS3 - *Walrus*.

Sumário

1 Introdução

Este capítulo apresenta uma breve introdução sobre computação em nuvem, destacando aspectos de desempenho, de dependabilidade, de performabilidade e de custo a serem avaliados nesse contexto. Em seguida, são apresentadas as motivações, os objetivos e as contribuições do trabalho proposto. Por fim, a estrutura geral da tese é apresentada.

1.1 Contexto

O conceito de computação em nuvem está mudando a forma como a infraestrutura de TI está sendo implantada nas empresas, na educação, na pesquisa e no governo (FURHT; ESCALANTE, 2010). A tendência pela busca do termo *cloud computing* aumentou drasticamente após outubro de 2007, quando a *Google* e a *IBM* anunciaram suas pesquisas sobre computação em nuvem (GOOGLE, 2013). Esse nível de interesse pelo termo *cloud computing* está relacionado à flexibilidade no fornecimento de *hardware*, *software*, aplicativos e serviços aos usuários (FURHT; ESCALANTE, 2010).

A computação em nuvem abrange os conceitos de *grid computing*, *cluster computing*, *autonomic computing* e *utility computing* (KIM et al., 2010; XIONG; PERROS, 2009).

O *National Institute of Standards and Technology (NIST)* (NIST, 2014) definiu a computação em nuvem como um modelo para acesso conveniente, sob demanda, e de qualquer localização, a uma rede compartilhada de recursos de computação (isto é, redes, servidores, armazenamento, aplicativos e serviços) que possam ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços (BAUER; ADAMS, 2012).

A computação em nuvem é composta de uma rede de servidores ou *data centers* virtualizados que podem fornecer aplicativos, plataformas, e infraestruturas como serviço através da *Internet* (TAURION, 2009). Essas infraestruturas, aplicativos e plataformas são oferecidos como serviço à medida que são demandados, através da rede para uma variedade de aplicativos que permitem a interatividade com os usuários, os quais pagam por sua utilização (BRIAN; CURTIS JR., 2013).

Alguns estudos demonstram que as empresas de pequeno e médio portes destinam 70% do seu tempo ao gerenciamento dos recursos de TI e apenas 30% do seu tempo em atividades relacionadas aos seus negócios (TAURION, 2009). Essas empresas necessitam de um investimento prévio no espaço físico, no fornecimento de energia, na refrigeração e na infraestrutura do *data center* para manterem seus serviços funcionando. Além disso, os níveis de utilização da infraestrutura de TI estão entre 5% e 10%, ocorrendo períodos de pico de 30% a 40% (TAURION, 2009).

A computação em nuvem permite uma maior eficiência na utilização dos recursos computacionais e no custo desses em relação aos recursos dedicados a usuários individuais, uma vez que os recursos são compartilhados por um vasto número de usuários (BAUER; ADAMS, 2012).

> Jim Stanten da *Forrestier Research* forneceu um exemplo de como o *New York Times* (NYT) utilizou a infraestrutura de processamento e armazenamento da computação em nuvem (BRIAN; CURTIS JR., 2013). O *New York Times* queria colocar seus arquivos históricos disponíveis para acesso *online*. Eles precisavam processar 11 milhões de artigos e colocá-los em arquivos PDF. As estimativas iniciais apontaram que seriam necessários centenas de servidores e aproximadamente 4TB de armazenamento. A organização de tecnologia da informação (TI) no *New York Times* estimou alguns meses, a necessidade de um grande orçamento e destacou a dificuldade de localizar os recursos computacionais. A organização de TI fez uma tentativa junto à *Amazon Web Services* e usou 100 instâncias EC2 (AMAZON, 2013a) e 4 terabytes de armazenamento S3 (AMAZON, 2013b). O trabalho foi terminado no dia seguinte com um custo total de 240,00 dólares (BRIAN; CURTIS JR., 2013).
>
> Outro exemplo vem do *Washington Post*. A organização de TI no *Washington Post* ativou 200 instâncias da *Amazon EC2* para processar 17.481 páginas de imagens PDF. Com uma velocidade de 60 segundos por página, o trabalho foi executado dentro de 9 horas e possibilitou o acesso ao portal público 26 horas depois. A organização de TI usou as instâncias EC2 (AMAZON, 2013a) por 1.407 horas a um custo total de 144,62 dólares (BRIAN; CURTIS JR., 2013).

A computação em nuvem permite que as empresas aumentem ou diminuam suas infraestruturas virtuais conforme a demanda, enquanto investem o capital nos seus negócios e em recursos humanos, ao invés de arcar com os custos de infraestruturas caras, licenças de *software*, manutenção de *hardware* e *software* e equipe técnica (TAURION, 2009).

O modelo de TI tradicional está baseado no investimento em capital (*capital expenses - capex*) onde as empresas precisam investir na infraestrutura de TI antes de utilizá-la. A computação em nuvem permite que as empresas eliminem o investimento inicial em infraestrutura de TI, possibilitando uma maior competitividade entre elas pois permite que essas transformem

os investimentos em capital (*capital expenses - capex*) em investimentos operacionais (*operating expenses - opex*) (ROSENBERG; MATEOS, 2010; TAURION, 2009).

A Figura 1.1 apresenta uma comparação da capacidade e da demanda entre infraestruturas tradicionais e infraestruturas de computação em nuvem (CHADES, 2013). A capacidade da infraestrutura tradicional pode ser expandida através da aquisição de recursos computacionais ou da terceirização desses recursos. Essa figura mostra que a capacidade da infraestrutura tradicional nunca coincide com a demanda, ocorrendo períodos de superprovisionamento o que resulta na subutilização desses recursos computacionais e na redução dos lucros devido aos elevados custos da manutenção de uma infraestrutura subutilizada e períodos de subprovisionamento o que ocasiona gargalos na utilização desses recursos computacionais e o não atendimento de novas solicitações (ROSENBERG; MATEOS, 2010; TAURION, 2009). Em contrapartida, essa figura também mostra que a capacidade da infraestrutura da computação em nuvem coincide com a demanda ao longo do tempo. Essa capacidade da infraestrutura da computação em nuvem de atender a demanda está relacionada a escalabilidade dessa infraestrutura. A diferença entre a capacidade e a demanda nas infraestruturas tradicional e da computação em nuvem resulta em um maior investimento em capital (*capex*) para a infraestrutura tradicional.

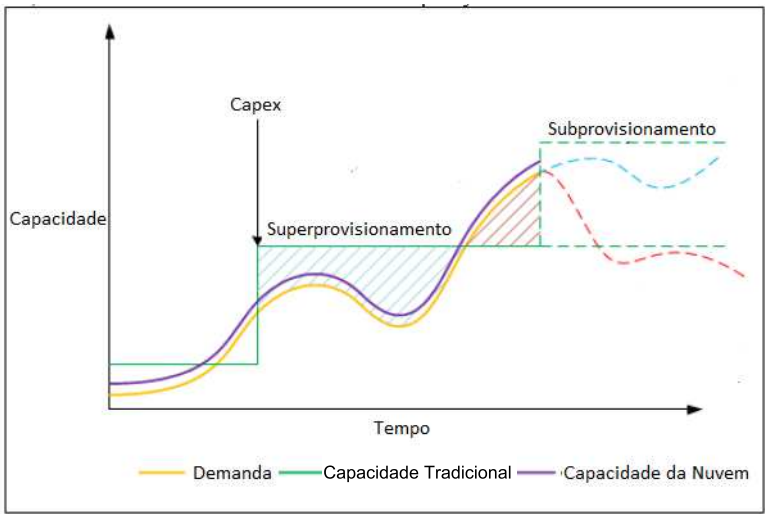


Figura 1.1: Capacidade X Demanda (ROSENBERG; MATEOS, 2010; TAURION, 2009)

A computação em nuvem surge como uma nova alternativa, pois aloca dinamicamente recursos computacionais conforme as solicitações dos usuários, eliminando os riscos de superprovisionamento e de subprovisionamento. Assim, a computação em nuvem possibilita uma utilização mais eficiente dos recursos computacionais e dos investimentos financeiros (ROSENBERG; MATEOS, 2010; TAURION, 2009).

1.2 Motivação

O tráfego de muitos aplicativos têm padrões de utilização que variam de acordo com a hora do dia, o dia da semana e a época do ano. O tráfego de algumas aplicações apresentam

outros padrões de sazonalidade, tais como, as aplicações de varejo que são submetidas a altos volumes de transações antes de datas comemorativas como o Natal e as aplicações financeiras que são submetidas a picos de utilização quando há necessidade de preparação dos resultados financeiros trimestrais e anuais. As empresas que adotam os sistemas computacionais tradicionais precisam adquirir recursos suficientes para atenderem a esses picos de utilização, assim como, uma capacidade extra de recursos computacionais para mitigar a ocorrência de defeitos durante os períodos de pico de utilização. Entretanto, quando a carga de trabalho submetida é menor que a projetada, os recursos excedentes são desperdiçados. A característica de compartilhamento de recursos e virtualização da computação em nuvem permite que os serviços hospedados na sua infraestrutura atendam aos diferentes níveis de requisições dos usuários (BAUER; ADAMS, 2012).

O planejamento da infraestrutura de nuvem privada é uma atividade essencial pois possibilita que o provedor da nuvem tenha recursos suficientes para alocá-los e liberá-los dinamicamente, quando submetido aos diferentes níveis de requisição dos usuários. Esse planejamento também permite o dimensionamento da infraestrutura de nuvem privada para suportar altos níveis de carga de trabalho com tempos de resposta aceitáveis (BAUER; ADAMS, 2012). A avaliação de desempenho da nuvem privada permite o atendimento dos diversos níveis de solicitações dos usuários mantendo a qualidade do serviço oferecido (XIONG; PERROS, 2009).

Um outro desafio é a garantia dos níveis de disponibilidade requeridos pelos diferentes serviços hospedados na nuvem privada. A ocorrência de defeitos nesses serviços pode ocasionar a degradação dos tempos de resposta deles e a interrupção do atendimento de uma requisição devido a indisponibilidade do recurso requerido. A interrupção desses serviços pode ser ocasionada pela ocorrência de eventos de falha no *hardware*, *software*, sistema energético, sistema de resfriamento e rede da nuvem privada. Quando a ocorrência de defeitos é constante, os usuários dão uma menor preferência a contratação dos provedores dos serviços devido à redução da disponibilidade, da confiabilidade e do desempenho desses serviços (BAUER; ADAMS, 2012).

A avaliação de dependabilidade pode minimizar a ocorrência de defeitos e eventos de falha (LAPRIE; AVIZIENIS; KOPETZ, 1992) na nuvem privada e promover os níveis de disponibilidade e confiabilidade definidos nos SLAs, evitando o pagamento de multas contratuais. Uma opção para a garantia da disponibilidade dos serviços ofertados na nuvem privada é atribuir equipamentos redundantes aos seus componentes. Os equipamentos redundantes permitem o restabelecimento do serviço, minimizando os efeitos da ocorrência dos eventos de falha. O grande problema dessa atribuição é a estimativa do número de equipamentos redundantes e a escolha do tipo de redundância que deve ser considerada para garantia da qualidade do serviço oferecido. A estimativa do tipo e número de equipamentos redundantes também deve considerar o custo do quantitativo de cada tipo de mecanismo de redundância atribuído aos componentes da nuvem computacional (AHSON; ILYAS, 2010; KONDO et al., 2009).

O planejamento das infraestruturas de energia, de resfriamento e de TI considerando os

investimentos financeiros realizados também é um dos grandes desafios para a nuvem privada (AHSON; ILYAS, 2010; KONDO et al., 2009). A avaliação de custos possibilita o planejamento do investimento financeiro na infraestrutura de nuvem privada de forma que ela hospede serviços com diferentes níveis de carga de trabalho dos usuários e vários níveis de disponibilidade (TAURION, 2009; ROSENBERG; MATEOS, 2010).

A avaliação de performabilidade permite a redução do impacto dos níveis de degradação do serviço ofertado na nuvem privada, em decorrência de defeitos e atividades de reparo em um determinado período de tempo (HAVERKORT et al., 2001).

O emprego de técnicas de modelagem pode representar aspectos de desempenho, de dependabilidade, de performabilidade e de custo dos sistemas configurados na nuvem privada (JAIN, 1991; MENASCÉ; ALMEIDA, 2005). Os formalismos adotados neste trabalho para a modelagem dos sistemas configurados na nuvem privada são as redes de *Petri* estocásticas (*Stochastic Petri Nets - SPN*) (BOLCH et al., 2006; GERMAN, 2000; MURATA, 1989) e os diagramas de bloco de confiabilidade (*Reliability Block Diagram - RBD*) (TRIVEDI, 2008; XIE; DAI; POH, 2004).

1.3 Objetivos

O principal objetivo deste trabalho é a proposição de uma solução integrada composta por uma metodologia, métodos, modelos de representação, modelos de otimização e uma ferramenta, para o planejamento de infraestruturas de nuvens privadas, com base na modelagem hierárquica e heterogênea dessa infraestrutura. Essa solução integrada permitirá a seleção de infraestruturas de nuvens privadas de acordo com os requisitos estabelecidos com os usuários.

Os objetivos específicos deste trabalho são:

• a confecção de modelos de otimização baseados na metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) (FEO; RESENDE, 1989, 1995) para a geração de infraestruturas de nuvens privadas;

• a confecção de modelos estocásticos e expressões algébricas que permitam representar e avaliar as infraestruturas de nuvens privadas;

• o desenvolvimento de uma estratégia de modelagem de disponibilidade baseada em redes de *Petri* estocásticas (*Stochastic Petri Nets - SPN*) (BOLCH et al., 2006; GERMAN, 2000), diagramas de bloco de confiabilidade (*Reliability Block Diagram - RBD*) (XIE; DAI; POH, 2004) e expressões algébricas, com a qual será possível a representação e avaliação de infraestruturas de nuvens privadas;

• o desenvolvimento do Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas (*Scenarios and Models Generator for Private Cloud Infrastructure Planning - SMG4PCIP*) (SOUSA et al., 2013) para geração automática dos modelos propostos, proporcionando o planejamento de infraestruturas de nuvens privadas.

1.4 Contribuições

A principal contribuição do trabalho é a proposição de uma solução integrada composta por uma metodologia, métodos, modelos de representação, modelos de otimização e uma ferramenta, para o planejamento de infraestruturas de nuvens privadas.

As demais contribuições do trabalho são as seguintes:

• a proposição de dois modelos baseados na metaheurística GRASP. O primeiro modelo provê a geração de cenários de infraestruturas de nuvens através da atribuição de diferentes configurações de *software* e *hardware* e o segundo modelo proporciona a geração de cenários de infraestruturas de nuvens por meio da atribuição de diferentes mecanismos de redundância aos componentes das infraestruturas de nuvens privadas;

• a apresentação de um modelo de desempenho baseado em redes de *Petri* estocásticas (*Stochastic Petri Nets* - SPN) (BOLCH et al., 2006; GERMAN, 2000) para o planejamento de infraestruturas de processamento e de armazenamento de nuvens privadas adequadas aos requisitos de desempenho. O modelo proposto avalia o impacto da atribuição de diferentes configurações de *software* e de *hardware* no desempenho da nuvem computacional. Esse modelo de desempenho também avalia o impacto de vários níveis de requisições na infraestrutura de computação em nuvem;

• a proposição de modelos de disponibilidade baseados em SPN (BOLCH et al., 2006; GERMAN, 2000) e RBD (XIE; DAI; POH, 2004) para o planejamento de infraestruturas de nuvens privadas que atendam as disponibilidades e os *downtimes* estabelecidos com os usuários. Os modelos de disponibilidade propostos representam as infraestruturas de nuvens privadas, a redundância ativo-ativo, as redundâncias ativo-passivo e a manutenção corretiva. Esses modelos de disponibilidade avaliam o efeito da atribuição de mecanismos de redundância aos componentes da nuvem computacional;

• a apresentação de modelos de custo baseados em expressões algébricas para o planejamento dos custos da aquisição de equipamentos do sistema de TI, da aquisição de equipamentos e *softwares* redundantes, da substituição de equipamentos, da aquisição de licenças de *software*, da equipe técnica e da equipe de manutenção;

• uma ferramenta para implementar a solução integrada proposta para o planejamento de infraestruturas de nuvens privadas.

1.5 Organização da Tese

O presente documento está dividido em 8 capítulos, os quais serão brevemente destacados nesta seção.

O Capítulo 2 apresenta a fundamentação teórica do trabalho proposto. Esse capítulo apresenta uma visão geral sobre a computação em nuvem, conceitos básicos sobre avaliação de desempenho e de dependabilidade, uma introdução aos mecanismos de redundância, noções

gerais sobre redes de *Petri* e diagramas de bloco de confiabilidade e conceitos básicos sobre a metaheurística GRASP.

O Capítulo 3 apresenta os trabalhos relacionados à tese proposta. Esse capítulo está dividido em trabalhos relacionados à avaliação de desempenho, à avaliação de dependabilidade, à avaliação de performabilidade, à avaliação de custo das infraestruturas de computação em nuvem e na comparação dos trabalhos relacionados.

O Capítulo 4 apresenta a metodologia e os métodos da solução integrada proposta para o planejamento de infraestruturas de nuvens privadas.

O Capítulo 5 apresenta os modelos de representação de desempenho, de disponibilidade, de custo e os modelos de otimização adotados pela solução integrada proposta.

O Capítulo 6 apresenta uma ferramenta para implementar a solução integrada proposta, o Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas (*Scenarios and Models Generator for Private Cloud Infrastructure Planning - SMG4PCIP*).

O Capítulo 7 apresenta três estudos de caso para a aplicação da solução integrada proposta. Esses estudos de caso são baseados no planejamento de infraestruturas de nuvens privadas onde ambientes virtuais de aprendizagem são configurados.

As conclusões e os trabalhos futuros são apresentados no Capítulo 8.

2

Fundamentação Teórica

Este capítulo apresenta os conceitos básicos para o entendimento do trabalho proposto neste documento. Inicialmente, apresenta uma introdução à computação em nuvem. Essa introdução engloba a classificação dos serviços oferecidos na computação em nuvem, a classificação dos modelos de nuvem computacional, as plataformas de nuvem computacional e uma visão geral delas. Em seguida, apresenta os conceitos básicos sobre desempenho e dependabilidade. Então, conceitos básicos sobre mecanismos de redundância são apresentados. Logo após, apresenta os principais conceitos sobre redes de *Petri* (*Petri Nets* - PNs), assim como características, propriedades e técnicas de análise. Em seguida, são apresentadas as redes de *Petri* estocásticas (*Stochastic Petri Nets* - SPNs), que são uma extensão à teoria inicial das redes de *Petri*. Então, são introduzidos o *moment matching* e a técnica de aproximação de fases. Posteriormente, apresenta diagramas de blocos de confiabilidade *(Reliability Block Diagram - RBD)*. Finalmente, esse capítulo apresenta a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*).

2.1 Computação em Nuvem

O surgimento da computação em nuvem sucedeu décadas de desenvolvimento dos sistemas computacionais. Na década de 60, houve o desenvolvimento dos primeiros *mainframes* comerciais. Em princípio, esses sistemas computacionais eram utilizados por apenas um usuário. Já na década de 70, esses sistemas passaram a ser compartilhados por vários usuários. Assim, surgiu o conceito de virtualização. Os usuários compartilhavam os recursos computacionais por meio de máquinas virtuais as quais eram alocadas para cada usuário. Esses usuários acessavam instâncias de máquinas virtuais por meio de terminais. No passado, a escassez de recursos computacionais impulsionou o surgimento da virtualização. Hoje, a virtualização na nuvem computacional ressurgiu devido à necessidade do aproveitamento pleno de todos os recursos físicos disponíveis (MENKEN, 2008; ROSENBERG; MATEOS, 2010).

Com o crescimento do poder computacional, ocorreu gradativamente a descentralização dos recursos computacionais e a expansão dos sistemas distribuídos. Assim, a década de 90 foi marcada pelo surgimento da arquitetura cliente-servidor. Nessa arquitetura, sistemas clientes

realizam requisições as quais são respondidas pelos sistemas servidores (MENKEN, 2008; ROSENBERG; MATEOS, 2010).

Em 2000 surgiram a computação em *cluster* e a computação em grade. Na computação em *cluster*, há um conjunto de nós computacionais semelhantes conectados por meio de uma rede de computadores local. Esse conjunto de nós computacionais é controlado e acessado por meio de um único nó mestre (MENKEN, 2008; ROSENBERG; MATEOS, 2010). Já na computação em grade, recursos computacionais de diferentes organizações são reunidos para permitir a colaboração de um grupo de pessoas ou instituições (MENKEN, 2008; ROSENBERG; MATEOS, 2010).

A computação em nuvem (MENKEN, 2008; ROSENBERG; MATEOS, 2010) foi proposta em 2005. A computação em nuvem pode oferecer uma série de recursos computacionais e serviços. Além disso, a computação em nuvem proporciona o acesso de forma dinâmica a recursos computacionais virtualizados de *data centers* por meio de *web browsers* na *Internet* (MENKEN, 2008; ROSENBERG; MATEOS, 2010).

A vantagem da computação em nuvem é a capacidade de virtualizar e compartilhar recursos entre diferentes aplicações melhorando a utilização do servidor. Nesse aspecto, o *Virtual Machine Monitor* é o componente de *software* que hospeda as máquinas virtuais. O VMM é responsável pela virtualização e controle dos recursos compartilhados pelas máquinas virtuais, tais como, processadores, dispositivos de entrada e saída, memória, armazenagem. Alguns exemplos de ferramentas de virtualização são o *KVM*, o *VMware* e o *Xen* (FURHT; ESCALANTE, 2010).

2.1.1 Classificação dos Serviços Oferecidos pela Computação em Nuvem

Segundo o NIST (NIST, 2014) os três principais serviços da computação em nuvem são a infraestrutura como serviço, a plataforma como serviço e o *software* como serviço.

Infraestrutura como Serviço (*Infrastructure as a Service - IaaS*): os recursos fornecidos ao usuário são processamento, armazenamento, comunicação de rede e outros recursos de computação fundamentais nos quais o usuário pode instalar e executar *softwares* em geral, incluindo sistemas operacionais e aplicativos (HUGOS; HULITZKY, 2010; RITTINGHOUSE; RANSOME, 2009; VELTE et al., 2010; SHROFF, 2010).

Plataforma como Serviço (*Platform as a Service - PaaS*): os recursos fornecidos ao usuário são aplicativos criados ou adquiridos pelo consumidor, desenvolvidos com linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo fornecedor (HUGOS; HULITZKY, 2010; RITTINGHOUSE; RANSOME, 2009; VELTE et al., 2010).

***Software* como Serviço (*Software as a Service - SaaS*)**: o recurso fornecido ao usuário é a utilização de aplicações do provedor executadas em uma infraestrutura na nuvem. As aplicações podem ser acessadas por vários dispositivos clientes através de interfaces, tais como um navegador *web* (HUGOS; HULITZKY, 2010; RITTINGHOUSE; RANSOME, 2009; SHROFF,

2010; VELTE et al., 2010).

2.1.2 Classificação dos Modelos de Computação em Nuvem

Os modelos de computação em nuvem segundo o NIST (NIST, 2014) são a nuvem privada, a nuvem pública, a nuvem híbrida e a nuvem comunitária, os quais estão descritos a seguir:

Nuvem pública: a infraestrutura de nuvem computacional é provisionada para uso aberto ao público em geral. A sua propriedade, gerenciamento e operação podem ser de uma empresa, uma instituição acadêmica, uma organização do governo, ou de uma combinação mista. Ela fica nas instalações do fornecedor (BABCOCK, 2010; BAUER; ADAMS, 2012; ROSENBERG; MATEOS, 2010).

Nuvem privada: a infraestrutura de computação em nuvem é provisionada para utilização exclusiva por uma única organização composta de diversos consumidores (como unidades de negócio). A sua propriedade, gerenciamento e operação podem ser da organização, de terceiros ou de uma combinação mista, e pode estar dentro ou fora das instalações da organização (BAUER; ADAMS, 2012; ROSENBERG; MATEOS, 2010).

Nuvem Comunitária: a infraestrutura de computação em nuvem é provisionada para utilização exclusiva por uma determinada comunidade de consumidores de organizações que têm interesses em comum (de missão, requisitos de segurança, políticas, observância de regulamentações). A sua propriedade, gerenciamento e operação podem ser de uma ou mais organizações da comunidade, de terceiros ou de uma combinação mista, e pode estar dentro ou fora das instalações das organizações participantes (BABCOCK, 2010; BAUER; ADAMS, 2012; ROSENBERG; MATEOS, 2010).

Nuvem híbrida: a infraestrutura de nuvem computacional é uma composição de duas ou mais infraestruturas na nuvem (como as nuvens privadas, comunitárias ou públicas) que permanecem entidades distintas, mas são interligadas por tecnologia padronizada ou proprietária que permite a comunicação de dados e portabilidade de aplicações (como a transferência de processamento para balanceamento de carga entre nuvens) (BABCOCK, 2010; BAUER; ADAMS, 2012; ROSENBERG; MATEOS, 2010).

2.1.3 Plataformas de Nuvem

Algumas plataformas de nuvens privadas oferecem o serviço de IaaS, dentre essas, serão descritas as plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack*.

Plataforma Eucalyptus é uma plataforma que permite a criação de nuvens privadas e híbridas em *data centers* de empresas (EUCALYPTUS, 2013). O *Eucalyptus* proporciona a compatibilidade com a *API* da infraestrutura de uma das nuvens comerciais mais populares, *Amazon Web Services* (AWS), permitindo que as ferramentas de gerenciamento sejam usadas em ambas as nuvens computacionais. Isso possibilita que as imagens instanciadas possam migrar

entre essas nuvens computacionais. Esse *framework* é projetado para ser compatível com várias distribuições *Linux* (como exemplo temos: *Ubuntu*, *RHEL*, *OpenSUSE*) e monitores de máquina virtual (exemplo: KVM, Xen) (EUCALYPTUS, 2013; NURMI et al., 2009).

A plataforma *Eucalyptus* é composta por vários componentes que interagem por meio de interfaces (D et al., 2010). Esses componentes são o Controlador de Nuvem (CLC), Controlador de Cluster (CC), Controlador de Nó (NC) e Controlador de Armazenamento (SC) (EUCALYPTUS, 2013; NURMI et al., 2009), que serão detalhados a seguir.

O Controlador de Nuvem é o ponto de acesso à nuvem computacional para os usuários e administradores. Esse componente consulta os controladores de nós para obter informações sobre os recursos computacionais disponíveis, toma decisões de escalonamento sobre as requisições dos usuários e realiza o escalonamento por meio de requisições aos controladores de *clusters* (EUCALYPTUS, 2013; NURMI et al., 2009). O Controlador de Cluster atua como um *gateway* entre o controlador de nuvem e os controladores de nós. Esse componente coleta informações sobre o escalonamento e a execução da máquina virtual em um controlador de nó específico, e gerencia as instanciações de máquinas virtuais (EUCALYPTUS, 2013; NURMI et al., 2009). O Controlador de *Cluster* contém um conjunto de máquinas físicas que fornecem recursos computacionais. Cada uma dessas máquinas físicas contém um serviço de Controlador de Nó que é responsável pelo controle da execução, inspeção, terminação das instâncias das máquinas virtuais (EUCALYPTUS, 2013; NURMI et al., 2009). O Controlador de Armazenamento/*Walrus* é um serviço de armazenamento que implementa a interface da *Amazon S3*, proporcionando um mecanismo de armazenamento, de acesso das imagens das máquinas virtuais e dos dados dos usuários (EUCALYPTUS, 2013; NURMI et al., 2009).

Plataforma Nimbus é uma plataforma que permite a criação de nuvens privadas e híbridas voltadas para aplicações científicas. Essa plataforma é projetada para ser compatível com várias distribuições *Linux* e *Unix* e com vários monitores de máquina virtual (exemplo: KVM e Xen). A plataforma *Nimbus* é composta pelo *Service Node* e pelo *VMM Node* (PROJECT, 2013a).

O *Service Node* executa os serviços *Nimbus IaaS* e *Cumulus Storage* da nuvem computacional. Esse componente recebe solicitações dos usuários por máquinas virtuais. O serviço *Nimbus IaaS* é responsável por atender as requisições dos usuários para criação e terminação das instâncias de máquinas virtuais. Já o serviço *Cumulus Storage* é responsável pelo atendimento das solicitações relacionadas as imagens das máquinas virtuais (PROJECT, 2013a). O *VMM Node* é responsável pela instanciação das máquinas virtuais solicitadas pelos usuários. Esse componente executa o programa *workspace-control* que interage com a biblioteca *libvirt* para o gerenciamento das máquinas virtuais (PROJECT, 2013a).

Plataforma OpenNebula é uma plataforma que permite a criação de nuvens públicas, privadas e híbridas. Essa plataforma é projetada para ser compatível com várias distribuições *Linux* e com vários monitores de máquina virtual (exemplo: KVM, Xen, VMware). A plataforma *OpenNebula* é composta pelo *Front-end* e pelo *Cluster Node* (PROJECT, 2013b).

O *Front-end* executa os serviços de administração da infraestrutura. Esse componente administra o ciclo de vida das máquinas virtuais, os *hypervisores*, as imagens das máquinas virtuais, a rede de computadores e a infraestrutura de armazenamento (PROJECT, 2013b). O *Cluster Node* fornece recursos computacionais para a instanciação das máquinas virtuais (PROJECT, 2013b).

Plataforma OpenStack é uma plataforma que permite a criação de nuvens públicas e privadas. O *OpenStack* proporciona compatibilidade com a API da *Amazon EC2* (AMAZON, 2013a). O gerenciamento das imagens das máquinas virtuais são baseados no *euca2ools* da plataforma *Eucalyptus* (EUCALYPTUS, 2013) e as imagens são disponibilizadas pelo *OpenStack Imaging Service* com o suporte do *Amazon S3* (AMAZON, 2013b), *OpenStack Object Storage* ou armazenamento local. Essa plataforma é projetada para ser compatível com várias distribuições *Linux* (como exemplo temos: *Ubuntu*, *RHEL* e *CentOS*) e monitores de máquina virtual (como exemplo temos: *Hyper-V*, KVM, QEMU, UML e Xen) (PROJECT, 2013c).

A plataforma *OpenStack* é constituída por três componentes que são: *OpenStack Object Storage*, *OpenStack Imaging Service* e *OpenStack Compute* (PROJECT, 2013c).

O *OpenStack Object Storage* permite a criação de uma infraestrutura de armazenamento escalável (PROJECT, 2013c). O *OpenStack Imaging Service* fornece as imagens das máquinas virtuais. Essas imagens são armazenadas no *OpenStack Imaging Service* e quando um usuário da nuvem computacional requisita uma instância, ele deve informar qual imagem será utilizada para gerar a instância da máquina virtual. Esse componente permite o gerenciamento dessas imagens, definindo formatos, metadados e realizando o armazenamento. Os formatos suportados são Raw, AMI, VHD, VDI, qcow2, VMDK, e OVF (PROJECT, 2013c). O *OpenStack Compute* gerencia os recursos computacionais das máquinas físicas, a rede de computadores, a segurança e o acesso ao serviço de IaaS da plataforma. Esse componente também gerencia os mecanismos de virtualização das máquinas físicas com base na API da biblioteca *libvirt* (PROJECT, 2013c).

2.2 Avaliação de Desempenho

A avaliação de desempenho de sistemas computacionais consiste de um conjunto de técnicas classificadas como as baseadas em medição e as baseadas em modelagem. As técnicas baseadas em modelagem podem ser classificadas como técnicas analíticas e técnicas baseadas em simulação (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

A medição de desempenho envolve essencialmente a monitoração do sistema enquanto está sob a ação de uma carga de trabalho. Para adquirir resultados representativos, a carga de trabalho deve ser cuidadosamente selecionada. Essa carga é utilizada nos estudos de desempenho, podendo ser real ou sintética. Embora a carga de trabalho real seja uma boa escolha por representar, de forma fiel, o sistema, ocasionalmente essa opção não é a desejável. Isso acontece quando o tamanho da carga não é considerável, e também quando esses dados receberam muitas perturbações ou, até mesmo, por questões de acessibilidade dos mesmos. Devido a esses motivos,

alguns mecanismos são usados para geração de carga de trabalho: *Kernels*, programas sintéticos e *Benchmarks* (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

A escolha da carga de trabalho é tão importante quanto à definição de qual estratégia de medição deve ser seguida. As diferentes estratégias de medição têm em sua base o conceito de evento, que é uma mudança no estado do sistema. A definição precisa de um evento depende da métrica que está sendo medida. Por exemplo, um evento pode ser definido como um acesso ao disco, uma referência de memória, uma operação de comunicação de uma rede de computadores ou uma mudança interna de um processador (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

As ferramentas desenvolvidas para a avaliação de desempenho de sistemas de computadores modificam o comportamento do que está sendo medido. Quanto maior a quantidade de informações e resolução que a ferramenta de medição pode fornecer, maior será a perturbação introduzida por essa ferramenta. Essa perturbação introduzida pela ferramenta de medição torna os dados coletados por ela menos confiáveis. A ferramenta de medição dirigida a evento ocasiona uma perturbação (*overhead*) no sistema apenas quando os eventos ocorrem. A vantagem das ferramentas de medição dirigidas a evento é a pouca ou nenhuma perturbação ocasionada por eventos que ocorrem com pouca ou nenhuma frequência. A desvantagem dessa ferramenta de medição é a grande perturbação ocasionada por eventos que ocorrem frequentemente. A ferramenta de medição por amostragem ocasiona perturbações independentes do número de vezes que o evento ocorre. A perturbação dependerá da frequência de amostragem determinada para coleta dos eventos (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

A qualidade das medições pode ser indicada através da precisão e exatidão da ferramenta de medição. A exatidão é a diferença absoluta entre o valor medido e o valor de referência. A precisão é a menor dispersão entre os valores obtidos através de múltiplas medições de uma determinada característica do sistema. Medições muito precisas são bem mais agrupadas em torno de um único valor medido. Medições imprecisas têm uma tendência a ter uma maior dispersão. A precisão dessas medidas é indicada pela dispersão delas em torno da média. A exatidão é a diferença entre o valor da média e o valor medido (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

A simulação é utilizada tanto na avaliação desempenho, quanto na validação de modelos analíticos. Ao contrário das medições, as simulações baseiam-se em modelos abstratos do sistema, logo não exigem que o sistema esteja totalmente implantado para que sejam aplicadas. Assim, os modelos utilizados durante a simulação são elaborados através da abstração de características essenciais do sistema, sendo que a complexidade e o grau de abstração dele podem variar de um sistema para outro. Durante a simulação, controlam-se, com maior eficiência, os valores assumidos por parâmetros do sistema. Com isso, fica mais fácil obter informações relevantes para a avaliação de desempenho (LILJA, 2005; MENASCÉ; ALMEIDA, 2005).

A técnica analítica utiliza fórmulas fechadas ou um conjunto de sistema de equações para descrever o comportamento de um sistema. As métricas de interesse podem ser fornecidas por meio da solução de fórmulas fechadas ou da solução exata ou aproximada de um conjunto

de sistema de equações providas por algoritmos da matemática numérica (BOLCH et al., 2006). Os fatores que influenciam e interferem no comportamento do sistema são modelados e representados através dos parâmetros de equações matemáticas. Apesar desses modelos considerarem parâmetros específicos de um sistema, podem ser adaptados para outros sistemas. Durante a construção dos modelos, deve-se levar em consideração a sua complexidade e praticidade. Os modelos permitem uma análise ampla e aprofundada em relação aos efeitos causados pelos parâmetros definidos nas equações sobre a aplicação. Além disso, também pode-se estabelecer possíveis relacionamentos entre cada um dos parâmetros considerados. Para validar os resultados alcançados através dos modelos elaborados, a técnica analítica pode compará-los aos valores reais medidos em testes experimentais. Esses valores deverão comprovar as predições realizadas através dos modelos (LILJA, 2005; MENASCÉ; ALMEIDA, 2005). Essa técnica apresenta menor custo de execução, quando comparada às demais técnicas de avaliação de desempenho (BOLCH et al., 2006).

As soluções de fórmulas fechadas são fornecidas por meio de um sistema de filas simples ou pequenas cadeias de *Markov* de tempo contínuo (BOLCH et al., 2006).

As soluções numéricas são fornecidas através de modelos baseados em espaço de estados que representam o comportamento do sistema por meio dos seus estados e da ocorrência de eventos (GERMAN, 2000; MARSAN et al., 1998; SAHNER; TRIVEDI; PULIAFITO, 2012; TRIVEDI, 2008).

2.3 Avaliação de Dependabilidade

A avaliação de dependabilidade denota a capacidade que um sistema tem de oferecer um serviço de forma confiável. As medidas de dependabilidade são confiabilidade, disponibilidade, mantenabilidade, performabilidade, segurança, testabilidade, confidencialidade e integridade (LAPRIE; AVIZIENIS; KOPETZ, 1992).

Avaliação da dependabilidade está relacionada ao estudo do efeito de erros, defeitos e falhas no sistema, uma vez que estes provocam um impacto negativo nos atributos de dependabilidade. Uma falha é definida como a falha de um componente, subsistema ou sistema que interage com o sistema em questão (MACIEL et al., 2012). Um erro é definido como um estado que pode levar a ocorrência de uma falha. Um defeito representa o desvio do funcionamento correto de um sistema. Um resumo das principais medidas de dependabilidade é mostrado a seguir.

A confiabilidade de um sistema é a probabilidade (P) de que esse sistema execute a sua função, de modo satisfatório, sem a ocorrência de defeitos, por um determinado período de tempo (T). A confiabilidade é representada pela Equação 2.1, onde T é uma variável aleatória que representa o tempo para ocorrência de defeitos no sistema (KUO; ZUO, 2003; RUPE, 2003).

$$R(t) = P\{T > t\}, t \geq 0 \tag{2.1}$$

A probabilidade da ocorrência de defeitos até um instante t, é representada pela Equação 2.2, onde T é uma variável aleatória que representa o tempo para ocorrência de falhas no sistema (KUO; ZUO, 2003; RUPE, 2003).

$$F(t) = 1 - R(t) = P\{T \leq t\} \tag{2.2}$$

A Equação 2.3 representa a confiabilidade, considerando a função de densidade F(t) do tempo para ocorrência de falhas (T) no sistema (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$R(t) = P\{T > t\} = \int_t^{\infty} F(t)dt \tag{2.3}$$

O tempo médio para defeito (*Mean Time to Failure* - MTTF) é o tempo médio para a ocorrência de defeitos no sistema. Quando esse tempo médio segue a distribuição exponencial com parâmetro λ, o MTTF é representado pela Equação 2.4 (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$MTTF = \int_0^{\infty} R(t)dt = \int_0^{\infty} \exp^{(-\lambda)t} = \frac{1}{\lambda} \tag{2.4}$$

As falhas podem ser classificadas em relação ao tempo, de acordo com o mecanismo que as originaram. O comportamento da taxa de falhas pode ser representado graficamente através da curva da banheira que apresenta três fases distintas: mortalidade infantil (1), vida útil (2) e envelhecimento (3). A Figura 2.1 mostra a variação da taxa de falhas de componentes de *hardware* em função do tempo (EBELING, 2004):

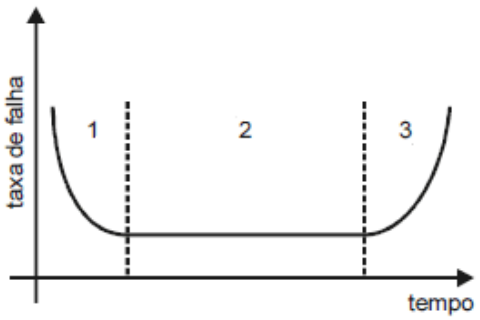


Figura 2.1: Curva da Banheira

Durante a fase de mortalidade infantil (1), ocorre uma redução na taxa de falhas. Falhas ocorridas nesse período são decorrentes dos defeitos da fabricação do equipamento. Com o intuito de encurtar esse período, fabricantes submetem os equipamentos a um processo chamado *burn-in*, onde eles são expostos a elevadas temperaturas de funcionamento. Na fase de vida útil (2), as falhas ocorrem aleatoriamente. Valores de confiabilidade de equipamentos fornecidos por fabricantes aplicam-se a esse período. O período de vida útil do equipamento normalmente não é uma constante. Ele depende do nível de estresse em que o equipamento é submetido durante

esse período. Durante a fase de envelhecimento (3), ocorre um aumento na taxa de falhas.

Em ambientes de alta disponibilidade, deve-se ter certeza de que a fase de mortalidade infantil tenha passado. Em alguns casos, é necessário deixar os equipamentos funcionando em um ambiente de testes durante esse período. Ao mesmo tempo, deve-se tomar cuidado para que o equipamento seja substituído antes de entrar na fase de envelhecimento.

A disponibilidade de um sistema é a probabilidade de que ele esteja operacional durante um determinado período de tempo, ou tenha sido restaurado após a ocorrência de um defeito. *Uptime* é o período de tempo em que o sistema está operacional, *downtime* é o período de tempo em que o sistema não está operacional devido a ocorrência de um defeito ou atividade de reparo, e o *uptime* + *downtime* é o período de tempo de observação do sistema. A Equação (2.5) representa a disponibilidade de um sistema (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$A = \frac{uptime}{uptime + downtime} \tag{2.5}$$

Os sistemas computacionais e aplicações requerem diferentes níveis de disponibilidades e portanto, podem ser classificados conforme esse níveis. *U.S. Federal Aviation Administration's National Airspace System's Reliability Handbook* classifica os sistemas computacionais e aplicações conforme os seus níveis de criticidade (BAUER; ADAMS, 2012). Esses sistemas computacionais e aplicações podem ser considerados como críticos seguros quando a disponibilidade necessária for 99,99999%, críticos quando a disponibilidade necessária for 99,999%, essenciais quando a disponibilidade necessária for 99,9% e rotineiros quando a disponibilidade necessária for 99% (BAUER; ADAMS, 2012).

A mantenabilidade é a probabilidade de que um sistema seja reparado em um determinado período de tempo (TR). A mantenabilidade é descrita pela Equação 2.6, onde TR denota o tempo de reparo. Essa equação representa a mantenabilidade, visto que o tempo de reparo TR tem uma função de densidade G(t) (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$V(t) = P\{TR \leq tr\} = \int_0^{tr} G(t)dt \tag{2.6}$$

O tempo médio para reparo (*Mean Time to Repair* - MTTR) é o tempo médio para reparo do sistema. Quando a função de distribuição do tempo de reparo é representado por uma distribuição exponencial com parâmetro μ, o MTTR é representado pela Equação 2.7 (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$MTTR = \int_0^{\infty} 1 - G(tr)dt = \int_0^{\infty} 1 - \exp^{(\mu)tr} = \frac{1}{\mu} \tag{2.7}$$

O tempo médio entre defeitos - (*Mean Time Between Failures* - MTBF) é o tempo médio entre os defeitos do sistema, representado pela Equação 2.8 (XIE; DAI; POH, 2004; KUO; ZUO, 2003; RUPE, 2003).

$$MTBF = MTTR + MTTF \tag{2.8}$$

A performabilidade descreve a degradação do desempenho de sistemas provocada pela ocorrência de defeitos (XIE; DAI; POH, 2004; KUO; ZUO, 2003).

Os modelos usados para avaliação de dependabilidade podem ser classificados como combinatoriais e baseados em espaço de estados. Os modelos combinatoriais capturam as condições que provocam falhas no sistema ou propiciam o seu funcionamento quando são consideradas as relações estruturais dos seus componentes. Os modelos combinatoriais mais conhecidos são diagrama de bloco de confiabilidade (RBD - *Reliability Block Diagram*) e árvore de falhas (FT - *Fault Tree*) (SAHNER; TRIVEDI; PULIAFITO, 2012; TRIVEDI, 2008). Os modelos baseados em espaço de estados representam o comportamento do sistema (ocorrência de falhas e atividades de reparo) por meio dos seus estados e da ocorrência de eventos. Esses modelos permitem a representação de relações de dependência entre os componentes dos sistemas. Os modelos baseados em espaço de estados mais usados são cadeias de *Markov* e redes de *Petri* estocásticas (SPN - *Stochastic Petri Net*) (GERMAN, 2000; MARSAN et al., 1998; SAHNER; TRIVEDI; PULIAFITO, 2012; TRIVEDI, 2008).

Os modelos SPN proporcionam grande flexibilidade na representação de aspectos de dependabilidade. Entretanto, esses modelos sofrem com problemas relacionados ao tamanho do espaço de estados para sistemas computacionais com grande número de componentes (GERMAN, 2000; MARSAN et al., 1998). Os modelos RBD são simples, fáceis de serem entendidos e seus métodos de solução têm sido extensivamente estudados. Esses modelos podem representar os componentes da computação em nuvem que não tenham uma relação de dependência para permitir uma representação eficiente, evitando problemas de crescimento demasiado do espaço de estados (SAHNER; TRIVEDI; PULIAFITO, 2012; TRIVEDI, 2008).

2.4 Mecanismos de Redundância

Os mecanismos de redundância proporcionam maior disponibilidade e confiabilidade ao sistema durante a ocorrência de eventos de falha, devido à manutenção de componentes operando em paralelo, ou seja, um sistema redundante possui um componente secundário que estará disponível quando o componente primário falhar. Assim, os mecanismos de redundância têm o objetivo de evitar pontos únicos de falha e consequentemente, proporcionar alta disponibilidade e recuperação de desastres, se necessário (BAUER; ADAMS; EUSTACE, 2011; SCHMIDT, 2006).

2.4.1 Mecanismos de Redundância Ativo-Ativo e Ativo-Espera

Os mecanismos de redundância podem ser classificados como ativo-ativo (*active-active*) e ativo-espera (*active-standby*).

Os mecanismos de redundância do tipo ativo-ativo são empregados quando os componentes primários e secundários compartilham a carga de trabalho do sistema. Quando qualquer um desses componentes falhar, o outro componente será o responsável pelo atendimento às requisições dos usuários do sistema. Esses mecanismos de redundância podem ser classificados como N+K, onde K componentes secundários idênticos aos N componentes primários são necessários para o compartilhamento da carga de trabalho do sistema. Na configuração N+1, um componente secundário idêntico aos N componentes primários é necessário para o compartilhamento da carga de trabalho do sistema. Na configuração N+2, dois componentes secundários idênticos aos N componentes primários são necessários para o compartilhamento da carga de trabalho do sistema (BAUER; ADAMS; EUSTACE, 2011).

Os mecanismos de redundância do tipo ativo-espera são empregados quando os componentes primários atendem às requisições dos usuários do sistema e os componentes secundários estão em espera. Quando os componentes primários falharem, os componentes secundários serão responsáveis pelo atendimento às requisições dos usuários do sistema. Os mecanismos de redundância ativo-espera podem ser classificados como *hot standby*, *cold standby* e *warm standby* (BAUER; ADAMS; EUSTACE, 2011).

No mecanismo de redundância *hot standby sparing*, os módulos redundantes que estão em *standby* ficam funcionando em sincronia com o módulo operacional, sem que sua computação seja considerada no sistema, e caso a ocorrência de um evento de falha seja detectada, ele está pronto para se tornar operacional imediatamente (BAUER; ADAMS; EUSTACE, 2011; RUPE, 2003).

No mecanismo de redundância *cold standby sparing*, os módulos redundantes ficam desligados e apenas quando um evento de falha ocorre é que eles serão ativados após um intervalo de tempo. No mecanismo de redundância *cold standby*, os módulos inativos que se encontram desenergizados, por hipótese, não falham, enquanto que o módulo ativo possui uma taxa de falha constante λ.

No mecanismo de redundância *warm standby sparing*, os módulos redundantes que estão em *standby* ficam funcionando em sincronia com o módulo operacional, sem que sua computação seja considerada no sistema. Caso um evento de falha seja detectado, o módulo redundante está pronto para se tornar operacional após em intervalo de tempo. Sistemas com *standby sparing* dos tipos *cold standby sparing* e *warm standby sparing* necessitam de mais tempo para recuperação, em relação ao *hot standby sparing*, porém os sistemas com *cold standby sparing* e *warm standby sparing* possuem a vantagem do menor consumo de energia e de não desgastarem os sistemas em *standby* (BAUER; ADAMS; EUSTACE, 2011; RUPE, 2003).

2.5 Redes de *Petri*

O conceito de redes de *Petri* foi introduzido por *Carl Adam Petri*, no ano de 1962, com a apresentação da sua tese de doutorado "*Kommunikation mit Automaten*" (Comunicação com

Autômatos) (MURATA, 1989) na faculdade de Matemática e Física da Universidade Darmstadf na Alemanha. Redes de Petri são ferramentas gráficas e matemáticas usadas para descrição formal de sistemas caracterizados pelas propriedades de concorrência, paralelismo, sincronização, distribuição, assincronismo e não-determinismo (MURATA, 1989).

A aplicabilidade das Redes de *Petri* como ferramenta para estudo de sistemas é importante por permitir representação matemática, análise dos modelos e também por fornecer informações úteis sobre a estrutura e o comportamento dinâmico dos sistemas modelados. As aplicações das Redes de *Petri* podem se dar em muitas áreas (sistemas de manufatura, desenvolvimento e teste de *software*, sistemas administrativos, entre outros) (MACIEL; LINS; CUNHA, 1996).

As redes de *Petri* são adotadas nessa tese para concepção dos modelos de desempenho e de dependabilidade propostos, pois esse formalismo matemático apresenta algumas características que são as suas vantagens em relação aos demais. Essas características são: a representação dinâmica do sistema modelado com o nível de detalhamento desejado; a descrição gráfica e formal que permite a obtenção de informações sobre o comportamento do sistema modelado através de suas propriedades comportamentais e estruturais; a representação de sincronismo, assincronismo, concorrência, compartilhamento de recursos, entre outros comportamentos; e a vasta aplicabilidade e documentação.

As redes de Petri são formadas por lugares (Figura 2.2(a)), transições (Figura 2.2(b)), arcos (Figura 2.2(c)) e marcas (Figura 2.2(d)). Os lugares correspondem às variáveis de estado e as transições, às ações ou eventos realizados pelo sistema. A realização de uma ação está associada a algumas pré-condições, ou seja, existe uma relação entre os lugares e as transições que possibilita ou não a realização de uma determinada ação. Após a realização de uma determinada ação, alguns lugares terão suas informações alteradas, ou seja, a ação criará uma pós-condição. Os arcos representam o fluxo das marcas pela rede de *Petri*, e as marcas representam o estado em que o sistema se encontra em determinado momento. Graficamente, os lugares são representados por elipses ou círculos, as transições, por retângulos, os arcos, por setas e as marcas, por meio de pontos (MACIEL; LINS; CUNHA, 1996).

(a) Lugar (b) Transição (c) Arco (d) Marca

Figura 2.2: Elementos de rede de Petri

Os dois elementos, lugar e transição, são interligados por meio de arcos dirigidos conforme mostrado na Figura 2.3. Os arcos que interligam lugares às transições (Lugar $\longrightarrow$ Transição) correspondem à relação entre as condições verdadeiras (pré-condição), que possibilitam a execução das ações. Os arcos que interligam as transições aos lugares (Transição $\longrightarrow$ Lugar) representam a relação entre as ações e as condições que se tornam verdadeiras com a

execução das ações (pós-condição) (MACIEL; LINS; CUNHA, 1996).

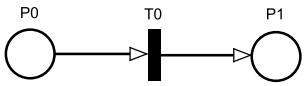


Figura 2.3: Exemplo de rede de Petri

A representação formal de um modelo em rede de *Petri (Petri Net - PN)* é a quíntupla $PN = \{P, T, F, W, M_0\}$ (MURATA, 1989), onde:

- P é o conjunto finito de lugares;
- T é o conjunto finito de transições, $P \cap T = \emptyset$;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;
- $W : F \rightarrow IR^+ \cup \{0\}$ é a função de atribuição de peso aos arcos;
- $M_0 : P \rightarrow IN$ é a função de marcação inicial, onde $P \cap T = \varnothing$ e $P \cup T \neq \varnothing$.

2.5.1 Propriedades das Redes de Petri

O estudo das propriedades de redes de *Petri* permite a análise do sistema modelado. Os tipos de propriedades podem ser divididos em duas categorias: as propriedades dependentes de marcação inicial, conhecidas como propriedades comportamentais, e as propriedades não dependentes de marcação, conhecidas como propriedades estruturais (MACIEL; LINS; CUNHA, 1996; MURATA, 1989).

Propriedades Comportamentais

As propriedades comportamentais são aquelas que dependem apenas da marcação inicial da rede de *Petri*. As propriedades abordadas são alcançabilidade, limitação, segurança, *liveness* e cobertura.

Alcançabilidade ou *reachability* indica a possibilidade de uma determinada marcação ser atingida pelo disparo de um número finito de transições a partir de uma marcação inicial. Dada uma rede de *Petri* marcada $RM = (R, M_0)$, o disparo de uma transição t_0 altera a marcação da rede de *Petri*. Uma marcação M' é acessível a partir de M_0 se existe uma sequência de transições que, disparadas, levam à marcação M'. Ou seja, se a marcação M_0 habilita a transição t_0, disparando-se essa transição, atinge-se a marcação M_1. A marcação M_1 habilita t_1 a qual, sendo disparada, atinge-se a marcação M_2 e assim por diante até a obtenção da marcação M'.

Seja um lugar $p_i \in P$, de uma rede de *Petri* marcada $RM = (R, M_0)$, esse lugar é k-limitado $(k \in IN)$ ou simplesmente limitado se para toda marcação acessível $M \in CA(R, M_0)$, $M(p_i) \leq k$.

O limite k é o número máximo de marcas que um lugar pode acumular. Uma rede de *Petri* marcada $RM = (R, M_0)$ é k-limitada se o número de marcas de cada lugar de RM não exceder k em qualquer marcação acessível de $RM(max(M(p)) = k, \forall p \in P)$.

Segurança ou *safeness* é uma particularização da propriedade de limitação. O conceito de limitação define que um lugar p_i é k-limitado se o número de marcas que esse lugar pode acumular estiver limitado ao número k. Um lugar que é 1-limitado pode ser simplesmente chamado de seguro.

Vivacidade ou *liveness* está definida em função das possibilidades de disparo das transições. Uma rede de *Petri* é considerada *live* se, independente das marcações que sejam alcançáveis a partir de M_0, for sempre possível disparar qualquer transição da rede de *Petri* através de uma sequência de transições $L(M_0)$. A ausência de bloqueio (*deadlock*) em sistemas está fortemente ligada ao conceito de vivacidade pois, *deadlock* em uma rede de *Petri* é a impossibilidade do disparo de qualquer transição da rede de *Petri*. O fato de um sistema ser livre de *deadlock* não significa que seja *live*, entretanto um sistema *live* implica em um sistema livre de *deadlocks*.

O conceito de cobertura está associado ao conceito de alcançabilidade e de *live*. Uma marcação M_i é coberta se existir uma marcação $M_j \neq M_i$, tal que $M_j \geq M_i$.

Propriedades Estruturais

As propriedades estruturais são aquelas que dependem apenas da estrutura da rede de *Petri*. Essas propriedades refletem características independentes de marcação. As propriedades analisadas neste trabalho são limitação estrutural e consistência.

Uma rede de *Petri* $R = (P, T, F, W, M_0)$ é classificada como estruturalmente limitada se for limitada para qualquer marcação inicial.

Ela será considerada consistente se, disparando uma sequência de transições habilitadas a partir de uma marcação M_0, retornar a M_0, porém todas as transições da rede de *Petri* são disparadas pelo menos uma vez.

Seja $RM = (R; M_0)$ uma rede de *Petri* marcada e s uma sequência de transições, RM é consistente se $M_0[s > M_0]$ e toda transição T_i, disparar pelo menos uma vez em s.

2.6 Rede de Petri Estocástica

Rede de *Petri* estocástica (SPN) (GERMAN, 2000) é uma das extensões de rede de *Petri* (PN) (MURATA, 1989) utilizada para a modelagem de desempenho e dependabilidade. Uma rede de *Petri* estocástica adiciona tempo ao formalismo de redes de *Petri*, com a diferença de que os tempos associados às transições temporizadas são distribuídos exponencialmente, enquanto o tempo associado às transições imediatas é zero. As transições temporizadas modelam atividades através dos tempos associados, de modo que o período de habilitação da transição temporizada corresponde ao período de execução da atividade, e o disparo da transição temporizada corresponde ao término da atividade. Níveis diferentes de prioridade podem ser atribuídos às transições.

A prioridade de disparo das transições imediatas é superior à das transições temporizadas. As prioridades podem solucionar situações de confusão (MARSAN et al., 1998). As probabilidades de disparo associadas às transições imediatas podem solucionar situações de conflito (BALBO, 2001; MARSAN et al., 1998).

Uma SPN é definida pela 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$, onde:

- $P = \{p_1, p_2, ..., p_n\}$ é o conjunto de lugares;
- $T = \{t_1, t_2, ..., t_m\}$ é o conjunto de transições imediatas e temporizadas, $P \cap T = \emptyset$;
- $I \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações);
- $H \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações);
- $\Pi \in \mathbb{N}^m$ é um vetor que associa o nível de prioridade a cada transição;
- $G \in (\mathbb{N}^n \to \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada a marcação do lugar à cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);

Os modelos SPN possuem dois tipos de estados (marcações), os estados tangíveis (*tangible*) e os estados voláteis (*vanish*). Os estados voláteis são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição imediata. O termo *vanish* é usado porque as marcações chegam a esses lugares e são instantaneamente consumidas. O tempo de permanência das marcações nesses lugares é zero. Os estados tangíveis são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição temporizada (MARSAN et al., 1998).

As transições temporizadas podem ser caracterizadas por diferentes políticas de memória tais como *Resampling*, *Enabling memory* e *Age memory* (MARSAN et al., 1998). As transições temporizadas também podem ser caracterizadas por diferentes semânticas de disparo conhecidas como *single server*, *multiple server* e *infinite server* (MARSAN et al., 1998).

Nos modelos SPN, as transições são disparadas obedecendo à semântica *interleaving* de ações (MARSAN et al., 1998). Essa semântica define que as transições são disparadas uma a uma, mesmo que o estado compreenda transições imediatas não conflitantes. A análise de um modelo SPN requer a solução de um sistema de equações igual ao número de marcações tangíveis. O gerador infinitesimal Q da cadeia de *Markov* de tempo contínuo (CTMC) associado ao modelo SPN é derivado de uma redução de um gráfico de alcançabilidade, rotulado com as

taxas das transições temporizadas ou pesos das transições imediatas. Modelos SPN permitem a geração de gráficos de alcançabilidade a partir dos quais cadeias de *Markov* de tempo contínuo (CTMC) são diretamente derivadas.

Redes de *Petri* estocásticas marcadas com um número finito de lugares e transições, são isomórficas às cadeias de *Markov* (HAVERKORT, 2001; MARSAN et al., 1998; MOLLOY, 1981; NATKIN, 1980). O isomorfismo de um modelo SPN com uma cadeia de *Markov* é obtido a partir do gráfico de alcançabilidade reduzido, que é dado através da eliminação dos estados voláteis e do rótulo dos arcos com as taxas das transições temporizadas e dos pesos das transições imediatas. As medições de desempenho e dependabilidade são obtidas através de simulações e de análises em estado estacionário e transiente baseadas na cadeia de *Markov* embutida no modelo SPN (BOLCH et al., 2006).

Os modelos SPN são usados para análise de desempenho e dependabilidade de sistemas, visto que permitem a descrição das atividades de sistemas através de gráficos de alcançabilidade. Esses gráficos podem ser convertidos em modelos *Markovianos*, que são utilizados para avaliação quantitativa do sistema analisado.

2.7 Técnica de Aproximação de Fases

Modelos SPN consideram somente transições imediatas e transições temporizadas com tempos de disparo distribuídos exponencialmente. Essas transições modelam ações, atividades e eventos.

Uma variedade de atividades podem ser modeladas através do uso dos construtores *throughput subnets* e *s-transitions*. Esses construtores são utilizados para representar distribuições expolinomiais, tais quais as distribuições *Erlang*, Hipoexponencial e Hiperexponencial (DESROCHERS; AL-JAAR, 1995).

A técnica de aproximação de fases pode ser aplicada para modelar ações, atividades e eventos não-exponenciais através do *moment matching*. O método apresentado calcula o primeiro momento em torno da origem (média) e o segundo momento central (variância) e estima os momentos respectivos da *s-transition* (DESROCHERS; AL-JAAR, 1995).

Dados de desempenho e de dependabilidade medidos ou obtidos de um sistema (distribuição empírica) com média μ_D e desvio-padrão σ_D podem ter seu comportamento estocástico aproximados através da técnica de aproximação de fases. O inverso do coeficiente de variação dos dados medidos ou obtidos de um sistema (ver Equação 2.9) permite a seleção da distribuição expolinomial que melhor se adapta à distribuição empírica. Esta distribuição empírica pode ser contínua ou discreta. Entre as distribuições contínuas, temos a Normal, Lognormal, *Weibull*, Gama, Uniforme Contínua, Pareto, Beta e Triangular e entre as distribuições discretas, temos a Geométrica, *Poisson* e Uniforme Discreta (JAIN, 1991).

$$\frac{1}{CV} = \frac{\mu_D}{\sigma_D} \tag{2.9}$$

A rede de *Petri* descrita na Figura 2.4 representa uma atividade temporizada com distribuição de probabilidade genérica.

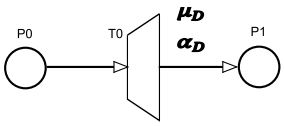


Figura 2.4: Distribuição Empírica

Dependendo do valor de inverso do coeficiente de variação dos dados medidos (ver Equação 2.9), a respectiva atividade tem uma dessas distribuições atribuídas: *Erlang*, Hipoexponencial ou Hiperexponencial.

Quando o inverso do coeficiente de variação é um número inteiro e diferente de um, os dados devem ser caracterizados através da distribuição *Erlang*, que é representada por uma sequência de transições exponenciais, cujo tamanho é calculado através da Equação 2.10. A taxa de cada transição exponencial é calculada através da Equação 2.11. Os modelos de Redes de *Petri* descritos na Figura 2.5 representam uma atividade temporizada com comportamento definido por uma distribuição de probabilidade *Erlang*.

$$\gamma = (\frac{\mu}{\sigma})^2 \tag{2.10}$$

$$\lambda = \frac{\gamma}{\mu} \tag{2.11}$$

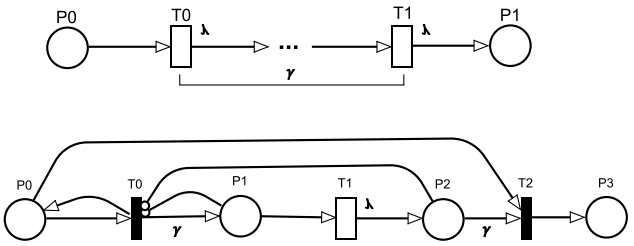


Figura 2.5: Distribuição Erlang

Quando o inverso do coeficiente de variação é um número maior que um (mas não é um número inteiro), os dados são representados através da distribuição hipoexponencial, a qual é representada por uma sequência de transições exponenciais, cujo tamanho é calculado através da Equação 2.12. As taxas das transições exponenciais são calculadas através das Equações 2.13 e 2.14, e os tempos médios atribuídos às transições exponenciais são calculados através das Equações 2.15 e 2.16. Os modelos de Redes de *Petri* apresentados na Figura 2.6 descrevem uma atividade temporizada com comportamento definido por uma distribuição de probabilidade hipoexponencial.

$$(\frac{\mu}{\sigma})^2 - 1 \leq \gamma < (\frac{\mu}{\sigma})^2 \tag{2.12}$$

$$\lambda_1 = \frac{1}{\mu_1} \tag{2.13}$$

$$\lambda_2 = \frac{1}{\mu_2} \tag{2.14}$$

$$\mu_1 = \mu \mp \frac{\sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{2.15}$$

$$\mu_2 = \gamma\mu \pm \frac{\sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{2.16}$$

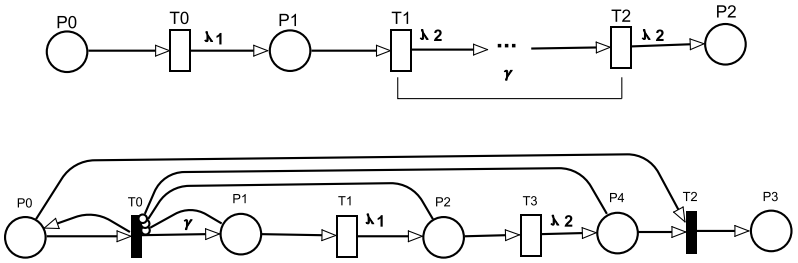


Figura 2.6: Distribuição Hipoexponencial

Quando o inverso do coeficiente de variação é um número menor que um, os dados devem ser caracterizados através de uma distribuição hiperexponencial. A taxa da transição exponencial deve ser calculada através da Equação 2.17, e os pesos das transições imediatas são calculados através das Equações 2.18 e 2.19. O modelo de Redes de *Petri* que representa uma atividade temporizada com comportamento definido por uma distribuição de probabilidade hiperexponencial é descrito na Figura 2.7.

$$\lambda_h = \frac{2\mu}{\mu^2 + \sigma^2} \tag{2.17}$$

$$r_1 = \frac{2\mu^2}{\mu^2 + \sigma^2} \tag{2.18}$$

$$r_2 = 1 - r_1 \tag{2.19}$$

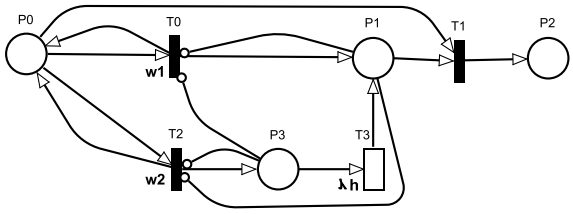


Figura 2.7: Distribuição Hiperexponencial

2.8 Diagrama de Blocos de Confiabilidade

O diagrama de blocos de confiabilidade *(Reliability Block Diagram - RBD)* é uma das técnicas mais usadas para a análise de confiabilidade de sistemas (MACIEL et al., 2012).

O diagrama de blocos de confiabilidade foi adotado na estratégia de modelagem de dependabilidade proposta nesta tese, devido à facilidade de analisar a disponibilidade e confiabilidade de sistemas (XIE; DAI; POH, 2004). O RBD também permite o cálculo da disponibilidade e confiabilidade por meio de fórmulas fechadas, pois é um modelo combinacional. Essas fórmulas fechadas tornam o cálculo do resultado mais rápido em relação à simulação, por exemplo.

Em um modelo de diagrama de blocos de confiabilidade, os componentes são representados com blocos combinados com outros blocos (ou seja, componentes) em série, paralelo ou combinações dessas estruturas. Um diagrama que tem componentes conectados em série exige que cada componente esteja funcionando para que o sistema seja operacional. Um diagrama que tem componentes conectados em paralelo exige que apenas um componente esteja funcionando para que o sistema seja operacional (TRIVEDI et al., 1996). Assim, o sistema é descrito como um conjunto de blocos funcionais interconectados para representar o efeito da disponibilidade e da confiabilidade de cada bloco na disponibilidade e na confiabilidade do sistema (SMITH, 2011).

A disponibilidade e a confiabilidade de dois blocos conectados em série é obtida através da Equação 2.20 (XIE; DAI; POH, 2004).

$$P_S = \prod_{i=1}^{n} P_i(t) \tag{2.20}$$

onde:

$P_i(t)$ descreve a confiabilidade $R_i(t)$, a disponibilidade instantânea $A_i(t)$ e a disponibilidade de estado estacionário A_i do bloco B_i.

A disponibilidade e a confiabilidade de dois blocos conectados em paralelo é obtida através da Equação 2.21 (XIE; DAI; POH, 2004).

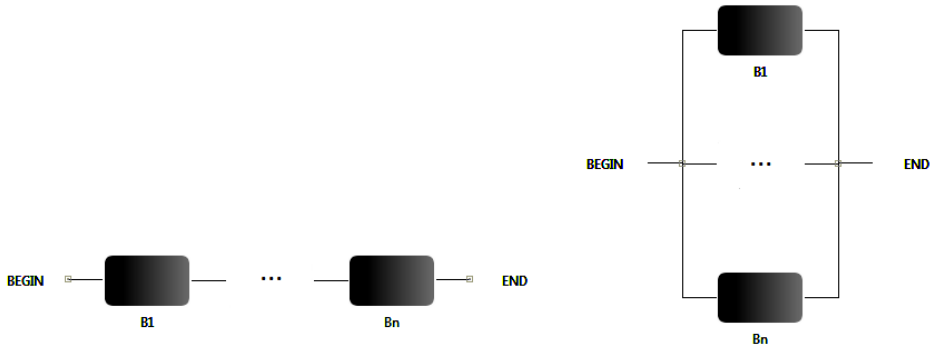
$$P_P = 1 - \prod_{i=1}^{n} (1 - P_i(t)) \tag{2.21}$$

onde:

$P_i(t)$ descreve a confiabilidade $R_i(t)$, a disponibilidade instantânea $A_i(t)$ e a disponibilidade de estado estacionário A_i do bloco Bi.

A Figura 2.8(a) mostra a conexão dos blocos em série e a Figura 2.8(b) mostra a conexão dos blocos em paralelo.

O diagrama de blocos de confiabilidade é utilizado, principalmente, em sistemas modulares que consistam de muitos módulos independentes, onde cada um pode ser facilmente representado por um bloco.



(a) Diagrama de Blocos em Série

(b) Diagrama de Blocos em Paralelo

Figura 2.8: Diagramas de Bloco de Confiabilidade

2.9 GRASP (*Greedy Randomized Adaptive Search Procedure*)

Metaheurísticas são métodos de solução que coordenam procedimentos. Esses procedimentos empregam o conceito de vizinhança de modo a criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções de um problema. Uma metaheurística visa produzir um resultado satisfatório para um problema, mas sem qualquer garantia de otimalidade (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

As metaheurísticas podem ser classificadas de acordo com a forma como exploram o espaço de soluções; elas podem ser definidas como metaheurísticas de busca local e metaheurísticas de busca populacional. Nas metaheurísticas de busca local, o procedimento de busca utiliza uma solução como ponto de partida em cada iteração. Como exemplo de métodos de busca local temos as metaheurísticas GRASP, Busca Tabu e *Simulated Annealing*. Nas metaheurísticas de busca populacional, um conjunto de soluções de boa qualidade é combinado com o intuito de produzir soluções melhores. Como exemplo de método populacional temos, os Algoritmos Genéticos (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

A metaheurística adotada neste trabalho para concepção dos modelos de otimização propostos foi o GRASP, devido às suas características que podem ser consideradas vantajosas em relação às demais metaheurísticas. Entre essas características temos a necessidade de poucos parâmetros de configuração, a utilização reduzida da memória, a facilidade de paralelização e a facilidade de implementação com a necessidade apenas de alterações nos parâmetros do método para problemas com a busca local já modelada (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

O GRASP foi introduzido em 1989, por Feo e Resende (FEO; RESENDE, 1989), e tornou-se uma das técnicas mais promissoras na busca de soluções para problemas de otimização combinatória. Um problema de otimização combinatória pode ser representado pela tarefa de

encontrar um subconjunto de elementos de um conjunto finito quando são consideradas uma série de regras. Esse subconjunto de elementos é definido como uma possível solução do problema com algum custo definido. O objetivo da otimização combinatorial é encontrar um subconjunto de elementos cujo custo seja o mínimo possível, ou seja, que represente a solução ótima do problema (GENDREAU; POTVIN, 2010)

O GRASP consiste em um processo iterativo, no qual cada iteração provê uma solução de boa qualidade para o problema de otimização (FEO; RESENDE, 1989).

Cada iteração consiste da fase de construção e da fase de busca local. A fase de construção permite a criação de elemento a elemento. Esses elementos compõem um conjunto que proporcionará a solução do problema. Além disso, cada elemento é selecionado aleatoriamente de uma lista restrita de candidatos (LRC) e adicionado ao conjunto solução. Na fase de construção, esses passos são repetidos até que seja encontrada uma solução inicial viável. Essa solução será submetida à fase de busca local (FEO; RESENDE, 1989, 1995).

A fase de busca local provê a busca na vizinhança dessa solução. Assim, soluções semelhantes à solução atual são investigadas com o objetivo de tentar obter uma melhoria na solução. O critério de parada das iterações é baseado no número máximo de interações e na melhor solução encontrada conforme apresentado no Algoritmo 1 (FEO; RESENDE, 1989, 1995).

```
1: Solution := ∞;
2: for interaction = 1 to MaxInteraction do
3:     Solution := Greed Randomized Construction(Seed);
4:     Solution := Approximate Local Search(Solution);
5:     Update Solution(Solution, BestSolution);
6: end for
7: return BestSolution;
```

Algoritmo 1: *GRASP(MaxInteraction, Seed)*

2.9.1 Fase Construção

Uma solução viável é construída iterativamente até que a solução esteja completa. Os elementos candidatos que compõem a solução são ordenados na Lista de Candidatos (LC) que contém todos os candidatos. Essa lista de candidatos é ordenada por uma função determinística que mede o benefício que o elemento escolhido proporciona a solução já construída. Um subconjunto denominado lista restrita de candidatos (LRC) é formado pelos melhores elementos que compõem a lista de candidatos (FEO; RESENDE, 1989, 1995).

O tamanho da LRC é controlado pelo parâmetro α, onde os valores definidos no intervalo $[0,1]$. Quando $\alpha = 0$, há um comportamento puramente construtivo do algoritmo e quando $\alpha = 1$, há comportamento totalmente aleatório. A componente probabilística do método é devido à escolha aleatória de um elemento da LRC. Esse procedimento permite que diferentes soluções

de boa qualidade sejam geradas a cada iteração. O Algoritmo 2 apresenta o pseudocódigo da fase de construção do GRASP (FEO; RESENDE, 1989, 1995).

1: $Solution := 0$;
2: *Evaluate the incremental costs of the candidate elements*;
3: **while** *Solution is not a complete solution* **do**
4: *Build the restricted candidate list* (RCL);
5: *Select an element s from the RCL at random*;
6: $Solution := Solution \cup s$;
7: *Reevaluate the incremental costs*;
8: **end while**
9: **return** *Solution*;

Algoritmo 2: *GreedyRandomizedConstruction(Seed)*

Assim, o principal parâmetro a ser configurado no GRASP é a quantidade de elementos da LRC. Esse parâmetro é o mais importante para o procedimento GRASP e é definido como $|LRC|$. Dessa forma, se $|LRC| = 1$, a solução inicial é totalmente determinística e, se $|LRC| = |LC|$, a solução é totalmente aleatória (FEO; RESENDE, 1989, 1995).

A média e a variância do valor da função objetiva das soluções construídas são afetadas por esse parâmetro. Dessa forma, se $|LRC|$ for pequeno, a variância e o espaço de soluções percorrido será menor, mas a possibilidade de encontrar ótimo local na fase de busca local será maior. Se $|LRC|$ for grande, a variância será maior e a possibilidade de encontrar um ótimo local será menor, mas a vizinhança a ser explorada e o número de iterações com soluções subótimas será maior (FEO; RESENDE, 1989, 1995).

O GRASP é adaptativo porque os benefícios associados a cada elemento são atualizados a cada iteração da fase de construção com o objetivo de refletir as mudanças ocorridas pela seleção de elementos anteriores. O GRASP é aleatório devido à forma de escolha dos melhores candidatos da lista (FEO; RESENDE, 1989, 1995).

Cada iteração é composta da construção da Lista Restrita de Candidatos (LRC), que contém um conjunto reduzido de elementos candidatos que pertencerão à solução; da escolha aleatória do elemento na LRC e inclusão desse elemento na solução; e da adaptação ou recálculo da função determinística para os elementos que ainda não pertencem à solução (FEO; RESENDE, 1989, 1995).

A melhor solução encontrada das iterações realizadas é retornada como resultado. Assim, para a aplicação eficaz do método é necessária a definição de um intervalo de valores para $|LRC|$. Dessa forma, haverá um balanceamento entre a qualidade das soluções, a quantidade de interações necessárias e a vizinhança explorada (FEO; RESENDE, 1989, 1995).

2.9.2 Fase Busca Local

Métodos de busca local em problemas de otimização constituem-se de técnicas baseadas na noção de vizinhança. Nesses métodos, o espaço de busca das soluções do problema é

percorrido iterativamente, de uma solução para a outra que seja sua vizinha (FEO; RESENDE, 1989, 1995).

A fase de busca local do GRASP aproveita a solução inicial da fase de construção e explora a vizinhança ao redor dessa solução. Se uma melhoria é encontrada em relação à solução corrente, essa solução é atualizada e a vizinhança ao redor da nova solução é pesquisada. O processo se repete até que nenhuma melhoria seja encontrada (FEO; RESENDE, 1989, 1995).

A eficácia da fase de busca local depende da escolha da vizinhança apropriada, do uso de estruturas de dados eficientes para acelerar a busca local, e da qualidade da solução inicial (FEO; RESENDE, 1989, 1995).

Uma vez obtida uma solução, consulta-se a estrutura da vizinhança relativa a essa solução. Uma solução é localmente ótima se não existir nenhuma solução melhor. As soluções iniciais encontradas na fase construtiva do GRASP não são necessariamente ótimos locais. Em consequência, há a necessidade da aplicação de um procedimento de busca local para melhoria das soluções provenientes da fase construtiva. Na fase de busca local, ocorre a realização de sucessivas trocas da solução atual, sempre que uma melhor solução é encontrada na vizinhança. Esse procedimento termina quando nenhuma solução melhor é encontrada. O Algoritmo 3 apresenta o pseudocódigo da fase de busca local do GRASP (FEO; RESENDE, 1989, 1995).

1: **while** *Solution is not locally optimal* **do**
2: $\quad$ *Find* $s^{'} \in N(Solution)$ *with* $f(s^{'}) < f(Solution)$;
3: $\quad$ $Solution = s^{'}$;
4: **end while**
5: **return** *Solution*;

Algoritmo 3: *Procedure ApproximateLocalSearch(Solution)*

2.9.3 Função Objetivo

A formulação matemática de um problema de otimização combinatória apresenta uma função objetivo e um conjunto de restrições, as quais estão relacionadas às variáveis de decisão. Essa função objetivo deve ser minimizada ou maximizada (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

Para o caso de minimização, um problema de otimização pode ser representado por min $f(s)$ sujeita a $s \in S$, onde $f : \Re^n \rightarrow \Re^n$ representa a função objetivo a ser minimizada e $S \subset \Re^n$ representa o conjunto finito das possíveis soluções viáveis para o problema. Para resolver um problema de otimização combinatória, deve-se encontrar uma solução ótima $s^* \in S$, que atenda à condição $f(s^*) \leq f(s), \forall\ s \subset S$ (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

A solução para o problema ou o ótimo global será o menor (ou maior) valor possível para a função, de forma que o valor atribuído às variáveis não viole nenhuma das restrições do

problema. A exploração de soluções na vizinhança da solução atual recebe o nome de ótimo local (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

2.10 Considerações Finais

Este capítulo apresentou uma introdução sobre computação em nuvem considerando aspectos relacionados à classificação dos serviços oferecidos, à classificação dos modelos e às plataformas de nuvens privadas. Em seguida, apresentou conceitos básicos sobre desempenho e dependabilidade. Então, apresentou os principais conceitos sobre redes de *Petri*, assim como sua extensão, as redes de *Petri* estocásticas. Depois, foram apresentados conceitos sobre os diagramas de blocos de confiabilidade. Finalmente, este capítulo apresentou a metaheurística GRASP. Os conceitos apresentados sobre computação em nuvem são fundamentais para o entendimento do ambiente avaliado pela solução integrada proposta neste trabalho, que será apresentada nos próximos capítulos. Os aspectos avaliados no ambiente de nuvem computacional são entendidos através dos conceitos de avaliação de desempenho e avaliação de dependabilidade. A metaheurística GRASP permite a geração de cenários de nuvens computacionais. Os formalismos matemáticos de redes de *Petri* estocásticas e diagramas de blocos de confiabilidade proporcionam a representação da nuvem computacional.

3

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados ao planejamento de infraestruturas de nuvens privadas. Esses trabalhos foram agrupados em seções de forma que cada uma apresente aspectos associados ao desempenho, à dependabilidade, à performabilidade e ao custo das infraestruturas de computação em nuvem. Finalmente, este capítulo apresenta uma comparação entre os trabalhos correlatos e o trabalho proposto.

3.1 Avaliação de Desempenho

Um dos maiores desafios para a computação em nuvem é a manutenção dos níveis de desempenho dos serviços ofertados, mais precisamente a minimização da latência, do tempo de resposta e maximização do *throughput* desses serviços. A avaliação de desempenho de infraestruturas da nuvem computacional proporciona o planejamento de infraestruturas que atendam a serviços com diferentes demandas e requisitos de desempenho (MENASCÉ; ALMEIDA, 2005).

A avaliação de desempenho pode ser realizada através de métodos analíticos, métodos numéricos e medição (LILJA, 2005). A escolha da técnica de avaliação depende do *status* de implementação ou construção do sistema a ser avaliado. Quando há o *design* de um novo sistema é possível usar o método analítico ou o método numérico. Já a técnica de medição necessita que o sistema a ser avaliado esteja implementado ou haja um protótipo desse sistema (JAIN, 1991).

O método analítico utiliza um conjunto de equações matemáticas para descrever o comportamento de um sistema. Os fatores que influenciam e interferem no comportamento do sistema são modelados e representados através dos parâmetros de equações matemáticas (BOLCH et al., 2006). Modelos analíticos são bastante utilizados para avaliação de infraestruturas de nuvens computacionais, pois têm como característica uma fundamentação teórica sólida, suporte a uma semântica precisa e a verificação de propriedades qualitativas e quantitativas (BOLCH et al., 2006).

Esta seção apresenta trabalhos que empregam modelos analíticos para a avaliação de desempenho e o planejamento de infraestruturas de nuvens computacionais. Xiong *et al.* (XIONG; PERROS, 2009) apresentam um modelo baseado em redes de filas para a avaliação de

desempenho e o planejamento da infraestrutura de nuvens computacionais. O modelo proposto representa o impacto de diferentes taxas de chegada de requisições dos usuários no desempenho da infraestrutura da nuvem computacional. Nesse modelo de desempenho, as requisições dos usuários são enviadas através do servidor *web* para infraestrutura da nuvem computacional que é representada pelo *data center*. O modelo concebido proporciona o planejamento da infraestrutura de nuvem necessário para o atendimento de determinado número de usuários com o tempo de resposta esperado. Esse modelo, no entanto, não proporciona o dimensionamento das máquinas físicas e máquinas virtuais necessárias para o atendimento de diferentes tipos de carga de trabalho com um determinado tempo de resposta.

Em (GHOSH; NAIK; TRIVEDI, 2011), (GHOSH et al., 2013), (GHOSH et al., 2013), Ghosh *et al.* apresentam uma estratégia de modelagem para avaliação de desempenho e o planejamento de nuvens computacionais. A infraestrutura de nuvem avaliada nesses trabalhos é composta de máquinas físicas distribuídas em três grupos com diferentes tempos de fornecimento das máquinas virtuais. Os três grupos são *hot* (*running*), *warm* (*turned on, but not ready*) e *cold* (*turned off*). No grupo *hot* de máquinas físicas, as máquinas virtuais pré-instanciadas são fornecidas com um tempo de resposta mínimo. Já no grupo *warm* de máquinas físicas, as máquinas virtuais não estão pré-instanciadas e o tempo de resposta corresponde ao tempo de instanciação. No grupo *cold* de máquinas físicas, as máquinas virtuais são inicializadas e instanciadas antes de serem fornecidas e o tempo de resposta corresponde ao tempo de inicialização e instanciação. As requisições dos usuários são encaminhadas para instanciação de máquinas virtuais provindas do grupo *hot* de máquinas físicas. Caso não haja máquinas físicas disponíveis no grupo *hot*, as máquinas virtuais provindas do grupo *warm* serão instanciadas. Caso não haja máquinas físicas disponíveis no grupo *warm*, as máquinas virtuais provindas do grupo *cold* serão instanciadas. Se nenhuma das máquinas físicas dos três grupos estiverem disponíveis, a requisição será rejeitada.

Em (GHOSH; NAIK; TRIVEDI, 2011), a estratégia de modelagem de desempenho proposta é composta de submodelos baseados em cadeias de *Markov*. Estes submodelos foram concebidos para representação das atividades necessárias para o atendimento das requisições dos usuários. Estes submodelos representam 1) a decisão do fornecimento do recurso, 2) a instanciação e fornecimento da máquina virtual e 3) a execução da máquina virtual. As atividades 1 e 3 são representadas por um submodelo cada e a atividade 2 é representada por três submodelos correspondentes as máquinas físicas dos grupos *hot*, *warm* e *cold*. Os submodelos concebidos foram combinados para o dimensionamento do número de máquinas físicas nos três grupos, de forma a atender as demandas dos usuários com os tempos de resposta e os custos requeridos. Essa estratégia de modelagem de desempenho (GHOSH; NAIK; TRIVEDI, 2011) permite a representação de nuvens computacionais com diferentes dimensões sem que haja um aumento significativo da complexidade dos modelos propostos.

O trabalho (GHOSH et al., 2013) apresenta a estratégia de modelagem de desempenho proposta em (GHOSH; NAIK; TRIVEDI, 2011) e um modelo baseado em redes *reward* esto-

cásticas (*Stochastic Reward Net* - SRN) para avaliação de desempenho e o planejamento de nuvens computacionais. A estratégia de modelagem de desempenho e o modelo SRN foram comparados em relação ao espaço de estados gerado e dos tempos de execução. Embora o modelo SRN permita uma maior flexibilidade quanto à representação da dependência entre as atividades de decisão do fornecimento do recurso, da instanciação e fornecimento da máquina virtual e da execução da máquina virtual, o resultado dessa comparação mostrou que a estratégia de modelagem de desempenho apresentou um menor espaço de estados e tempo de execução.

Em (GHOSH et al., 2013), Ghosh *et al.* apresentam a estratégia de modelagem de desempenho proposta em (GHOSH; NAIK; TRIVEDI, 2011; GHOSH et al., 2013), uma estratégia de modelagem para avaliação de dependabilidade baseada em redes *reward* estocásticas e modelos de custo baseados em expressões matemáticas. A estratégia de modelagem de dependabilidade é composta de submodelos SRN que representam os estados de falha e de reparo das máquinas físicas dos grupos *hot*, *warm* e *cold*. Os modelos de custo representam os gastos com as infraestruturas de TI, energética e de resfriamento, o reparo das máquinas físicas, o *downtime* e a multa devido às requisições não atendidas. O diferencial desse trabalho de Ghosh *et al.* é a apresentação de um modelo de otimização baseado na metaheurística *simulated annealing* para o dimensionamento das máquinas físicas dos grupos *hot*, *warm* e *cold* da nuvem computacional considerando parâmetros relacionados ao desempenho, disponibilidade e custo.

É fato que a avaliação de desempenho de nuvens computacionais tenha sido bastante discutida na literatura nos últimos anos. Neste contexto, este trabalho foca especificamente em uma metodologia para avaliação de desempenho e o planejamento de nuvens privadas. Essa metodologia proporciona a geração de cenários através da metaheurística GRASP, com vários *softwares* configurados em diferentes infraestruturas de nuvens. Esses cenários são representados através de redes de *Petri* estocásticas. O resultado da avaliação dos modelos concebidos para representação dos cenários permite a escolha de cenários que atendam aos requisitos de desempenho.

Enquanto o trabalho apresentado nesta tese possibilita o planejamento dos *softwares* que devem ser configurados nas diferentes máquinas virtuais da nuvem computacional, os trabalhos relacionados tratam apenas do planejamento das máquinas físicas utilizadas para a instanciação das máquinas virtuais da nuvem computacional.

3.2 Avaliação de Dependabilidade

Outro desafio para a computação em nuvem é a garantia de diferentes níveis de dependabilidade, conforme os requisitos dos serviços ofertados. Um provedor de serviços na nuvem computacional deve garantir diferentes níveis de dependabilidade para diferentes tipos de usuários, dependendo dos acordos de nível de serviço (SLAs) estabelecidos entre eles.

O emprego da avaliação de dependabilidade na infraestrutura de nuvem computacional permite o cálculo dos níveis de disponibilidade, de confiabilidade e de performabilidade da

infraestrutura de computação em nuvem. Essa avaliação pode ser realizada através de um conjunto de técnicas que podem ser sugeridas para aumentar os níveis desses atributos. Essas técnicas são os diferentes tipos de mecanismos de redundância (RUPE, 2003) e de políticas de manutenção (WANG; PHAM, 2006).

Apesar de existirem vários estudos que propõem a avaliação de dependabilidade de nuvens computacionais por meio de modelos analíticos, poucos trabalhos têm adotado uma estratégia de modelagem hierárquica e heterogênea visando uma redução da complexidade na representação deste ambiente. Em geral, essas estratégias são compostas de modelos combinatoriais e modelos baseados em espaço de estados. Os modelos combinatoriais proporcionam uma maior facilidade para análise da disponibilidade e confiabilidade de sistemas, mas apresentam pouca capacidade de representação de características dinâmicas (MACIEL et al., 2012; TRIVEDI, 2008). Os modelos baseados em espaço de estados podem representar sistemas complexos, mas demandam maior custo para o cálculo da disponibilidade e confiabilidade destes sistemas (MACIEL et al., 2012; TRIVEDI, 2008). Esta estratégia de modelagem associa as vantagens dos modelos combinatoriais e dos modelos baseados em espaço de estados. Há ainda um número reduzido de trabalhos que empregam uma estratégia de modelagem para avaliação do impacto da atribuição de mecanismos de redundância na nuvem computacional.

Embora a avaliação de dependabilidade de nuvens computacionais tenha sido bastante discutida na literatura, esta tese propõe uma metodologia para avaliação de dependabilidade e o planejamento de nuvens privadas. Essa metodologia proporciona a geração de cenários através da metaheurística GRASP, com a atribuição de mecanismos de redundância ativo-passivos e ativos-passivos aos componentes da nuvem computacional. Os cenários gerados são representados através de uma estratégia de modelagem hierárquica e heterogênea baseada em diagramas de blocos de confiabilidade e redes de *Petri* estocásticas. Os modelos RBD representam as plataformas de nuvens, a máquina física, a máquina virtual, os gerenciadores de recursos das plataformas de nuvem e o mecanismo de redundância ativo-passivo *hot standby* e os modelos SPN representam os mecanismos de redundância ativos-passivos *cold standby* e *warm standby*, o mecanismo de redundância ativo-ativo e a alocação da manutenção corretiva a nuvem computacional. A estratégia de modelagem proposta combina modelos RBD de baixo nível e modelos RBD ou SPN de alto nível para a representação dos cenários gerados. Os resultados da avaliação dos modelos concebidos por esta estratégia de modelagem possibilitam a escolha de cenários que atendam aos requisitos de dependabilidade.

Esta seção apresenta trabalhos que empregam modelos analíticos para a avaliação de dependabilidade e o planejamento de capacidade de infraestruturas de nuvens computacionais. Em (WEI B.; KONG, 2011), Wei *et al.* propõem uma estratégia de modelagem hierárquica e heterogênea para avaliação de *data centers* virtuais (*VDC - Virtual Data Center*) da computação em nuvem, com base em redes de Petri estocásticas generalizadas (*General Stochastic Petri Nets - GSPNs*) e diagramas de bloco de confiabilidade (*Reliability Block Diagram - RBD*). A modelagem hierárquica e heterogênea de VDCs permite a simplificação na representação desse

sistema computacional e evita a explosão de espaço de estados. Um modelo RBD de alto nível descreve toda a infraestrutura dos VDCs. Essa infraestrutura é representada por vários *clusters* com seus respectivos servidores, os quais são conectados por meio de módulos de rede. Um outro modelo RBD representa os servidores os quais são compostos pelas máquinas virtuais, monitores de máquinas virtuais e *hardware*. Um modelo GSPN de baixo nível representa os componentes do servidor em estado de falha e de reparo. Esse modelo GSPN foi estendido para representar as dependências entre os componentes dos servidores de um *cluster*. Embora essa estratégia de modelagem permita o cálculo da disponibilidade e confiabilidade de VDCs através de fórmulas fechadas, o trabalho poderia explorar o modelo GSPN expandido através do mapeamento de métricas de disponibilidade e confiabilidade para avaliar o impacto da atribuição de mecanismos de redundância na infraestrutura de VDCs da nuvem computacional.

Os trabalhos (DANTAS et al., 2012) e (DANTAS et al., 2012) apresentam uma estratégia de modelagem heterogênea e hierárquica baseada em diagramas de bloco de confiabilidade e cadeias de *Markov* para avaliação de nuvens computacionais configuradas com a plataforma *Eucalyptus*. Nessa estratégia de modelagem, modelos de baixo nível baseados em cadeias de *Markov* representam os *hardwares* e *softwares* da plataforma *Eucalyptus* com a atribuição do mecanismo de redundância *warm standby* e modelos de alto nível baseados em diagramas de bloco de confiabilidade representam os componentes da plataforma *Eucalyptus*. Essa estratégia de modelagem possibilitou a análise do efeito da atribuição do mecanismo de redundância *warm standby* aos componentes da plataforma *Eucalyptus* por meio de fórmulas fechadas da disponibilidade e *downtime*.

Silva *et al.* (SILVA et al., 2013) apresentam uma estratégia de modelagem heterogênea e hierárquica baseada em diagrama de bloco de confiabilidade e redes de *Petri* estocástica para avaliação de dependabilidade de serviços oferecidos em nuvens computacionais configuradas em *data centers* distribuídos geograficamente. Este trabalho considera *data centers* compostos por dois conjuntos de máquinas físicas conhecidos como *hot* e *warm*. O conjunto *hot* é composto por n máquinas físicas ativas e máquinas virtuais em execução e o conjunto *warm* é composto de m máquinas físicas ativas e máquinas físicas que não estão em execução. O número de máquinas físicas no *data center* é $t = m + n$ e cada uma pode ter várias máquinas virtuais instanciadas, de acordo com a sua capacidade computacional. Se ocorrer um falha em uma máquina física que executa uma máquina virtual, esta pode ser migrada para outra máquina física no mesmo *data center*. Se não houver mais máquinas físicas disponíveis no mesmo *data center*, esta pode ser migrada para outro *data center*. A estratégia de modelagem proposta avalia o impacto da migração de máquinas virtuais entre diferentes *data centers* por meio de métricas de dependabilidade. A estratégia de modelagem concebe modelos RBD de baixo nível cujos resultados são usados em modelos SPN de alto nível. O modelo RBD representa as máquinas físicas que compõem um *data center*. Os modelos SPN representam 1) os componentes da nuvem computacional em estado de falha ou de reparo, 2) o comportamento das máquinas virtuais em execução na máquina física e 3) a migração da máquina virtual para outro *data center*.

Esta estratégia de modelagem permitiu a avaliação do impacto da atribuição de mecanismos georedundantes na disponibilidade da nuvem computacional.

Os trabalhos (WEI B.; KONG, 2011), (DANTAS et al., 2012) e (DANTAS et al., 2012) apresentam uma estratégia de modelagem baseada em modelos combinacionais de alto nível e modelos baseados em espaço de estados de baixo nível, mas o trabalho (DANTAS et al., 2012) utiliza esta estratégia para proporcionar a avaliação do impacto da atribuição de um mecanismo de redundância. Silva *et al.* (SILVA et al., 2013) adota uma estratégia de modelagem baseada em modelos baseados em espaço de estados de alto nível e modelos combinatoriais de baixo nível e também avalia o impacto da atribuição de um mecanismo de redundância.

O trabalho proposto, no entanto, combina modelos RBD de baixo nível e modelos RBD e SPN de alto nível conforme o nível de complexidade do mecanismo de redundância atribuído aos componentes da nuvem computacional. Diferente do trabalho proposto nesta tese, nenhum dos trabalhos relacionados apresenta modelos para representação de um mecanismo de redundância ativo-ativo e da alocação de equipes de manutenção na nuvem computacional.

3.3 Avaliação de Performabilidade

A avaliação de performabilidade descreve o efeito de eventos de falhas e atividades de reparo na degradação do desempenho de sistemas. Para a avaliação de performabilidade é comum a utilização de técnicas de modelagem hierárquicas para combinação de um modelo de dependabilidade de alto nível e modelos de desempenho de baixo nível, um modelo de desempenho para cada estado do modelo de dependabilidade. Essa modelagem hierárquica tem o objetivo de evitar os problemas *largeness* e *stiffness* (PULIAFITO; RICCOBENE; SCARPA, 1996) (SAHNER; TRIVEDI; PULIAFITO, 1996).

A integração da modelagem de aspectos de desempenho e dependabilidade de sistemas é conhecida como modelagem de performabilidade. A modelagem de performabilidade permite a avaliação de desempenho considerando a degradação dos níveis de serviço provocados pelos eventos de falhas durante um determinado período de tempo (SAHNER; TRIVEDI; PULIAFITO, 1996).

Apesar de existirem vários estudos que propõem a avaliação de desempenho e a avaliação de dependabilidade de nuvens computacionais por meio de modelos analíticos, poucos trabalhos têm adotado uma estratégia de modelagem hierárquica e heterogênea para avaliação de performabilidade. No entanto, esta tese apresenta uma metodologia que propicia uma estratégia de modelagem hierárquica e heterogênea baseada em diagramas de blocos de confiabilidade e redes de *Petri* estocásticas. Nessa estratégia, modelos SPN de baixo nível representam aspectos de desempenho da nuvem computacional e modelos SPN ou RBD de alto nível representam aspectos de dependabilidade. Os resultados das avaliações dos modelos de desempenho e dos modelos de dependabilidade são combinados para representar o impacto da ocorrência de eventos de falhas e atividades de reparo no desempenho da nuvem computacional.

Esta seção apresenta trabalhos que empregam modelos analíticos para a avaliação de performabilidade e o planejamento de infraestruturas de nuvens computacionais. O artigo (GHOSH et al., 2010) emprega a técnica de modelagem hierárquica baseada em cadeias de *Markov* para avaliação de performabilidade de nuvens computacionais. Nessa técnica de modelagem, submodelos representam a infraestrutura da nuvem computacional e a solução geral do modelo proposto é obtida por meio das soluções individuais de cada submodelo. A nuvem computacional avaliada neste trabalho é a mesma dos trabalhos (GHOSH; NAIK; TRIVEDI, 2011), (GHOSH et al., 2013), (GHOSH et al., 2013). Os modelos de desempenho foram concebidos para representação das atividades necessárias para o atendimento das requisições dos usuários. Esses submodelos representam 1) a decisão do fornecimento do recurso, 2) a instanciação e fornecimento da máquina virtual e 3) a execução da máquina virtual. As atividades 1 e 3 são representadas por um submodelo cada e a atividade 2 é representada por três submodelos correspondentes as máquinas físicas dos grupos *hot*, *warm* e *cold*. Os submodelos concebidos foram combinados para o dimensionamento do número de máquinas físicas nos três grupos, de forma a atender as demandas dos usuários com os tempos de resposta requeridos. O modelo de dependabilidade representa a ocorrência de defeitos e atividades de reparo nas máquinas físicas dos grupos *hot*, *warm* e *cold*. A contribuição do trabalho por Ghosh *et al.* (GHOSH et al., 2010) é a estratégia de modelagem hierárquica para análise de performabilidade de infraestruturas de nuvens computacionais através da combinação de um modelo de dependabilidade de alto nível e modelos de desempenho de baixo nível, um modelo de desempenho para cada estado do modelo de dependabilidade.

Em (SILVA; MACIEL; ZIMMERMANN, 2013; SILVA et al., 2014), Silva *et al.* apresentam uma estratégia de modelagem hierárquica e heterogênea para avaliação de performabilidade de nuvens computacionais configuradas em *data centers* distribuídos geograficamente. Essa estratégia de modelagem avalia o impacto da migração de máquinas virtuais entre diferentes *data centers* por meio da combinação de um modelo RBD de baixo nível e de modelos SPN de alto nível. O *data center* da nuvem computacional avaliado neste trabalho é o mesmo do trabalho (SILVA et al., 2013). O modelo RBD representa as máquinas físicas que compõem o *data center*. Os modelos SPN representam 1) os componentes da nuvem computacional em estado de falha ou de reparo, 2) o comportamento das máquinas virtuais em execução na máquina física quando são submetidas a eventos de falhas e atividades de reparo e 3) a migração da máquina virtual para outro *data center*. Essa estratégia de modelagem permitiu a avaliação do impacto da atribuição de mecanismos georedundantes no desempenho da infraestrutura da nuvem computacional quando são considerados eventos de falha e atividades de reparo nessa infraestrutura. O trabalho (SILVA et al., 2014) também apresentou uma ferramenta para implementação da estratégia de modelagem proposta. A contribuição desses trabalhos (SILVA; MACIEL; ZIMMERMANN, 2013; SILVA et al., 2014) é a estratégia de modelagem hierárquica e heterogênea para análise de performabilidade de infraestruturas de nuvens computacionais configuradas em *data centers* distribuídos geograficamente.

O diferencial deste trabalho para os artigos (SILVA; MACIEL; ZIMMERMANN, 2013;

SILVA et al., 2014) é combinação dos resultados das avaliações de modelos de desempenho e de modelos de dependabilidade, proporcionando uma menor complexidade da avaliação da performabilidade de nuvens computacionais. Em relação ao artigo (GHOSH et al., 2010), a avaliação de performabilidade também apresenta uma menor complexidade, pois adota uma estratégia de modelagem baseada em modelos combinatoriais e modelos baseados em espaço de estados.

3.4 Avaliação de Custo

Um dos maiores benefícios da computação em nuvem é a capacidade de fornecer a alocação dinâmica de recursos conforme a necessidade dos seus clientes, cobrando de acordo com a utilização destes recursos.

A avaliação de custos tem sido bastante discutida na literatura nos últimos anos, mas poucos trabalhos empregam expressões matemáticas para avaliação de custo e o planejamento de infraestruturas de nuvens privadas. Em (LI et al., 2009), Li *et al.* propõem um conjunto de modelos de custo baseados em expressões matemáticos para o cálculo dos gastos com servidores, *softwares*, suporte da manutenção, equipamento de rede, sistema energético, sistema de refrigeração, instalações e imóvel. O trabalho (LI et al., 2009) também propôs uma ferramenta *web* para o cálculo e análise dos custos dos serviços oferecidos na nuvem computacional com base nas expressões matemáticas criadas. Essa abordagem apresentou insumos para o planejamento do custo da nuvem computacional.

De forma semelhante ao trabalho do Li *et al.* (LI et al., 2009), esta tese propõe expressões matemáticas para o cálculo de gastos com a aquisição de equipamentos de TI, aquisição de licenças de *software*, aquisição de equipamentos redundantes, equipe técnica, equipe de manutenção e substituição de equipamentos da nuvem privada. Entretanto, esta tese apresenta uma metodologia que provê essas expressões matemáticas e utiliza os resultados dos cálculos desses custos para seleção de cenários de infraestruturas de nuvens que atendam aos requisitos de custo.

3.5 Comparação dos Trabalhos Relacionados

Esta seção mostra uma comparação dos trabalhos apresentados em relação ao trabalho proposto nesta tese. Esses trabalhos relacionados focam nas avaliações de desempenho, de dependabilidade, de performabilidade e de custo da infraestrutura de computação em nuvem através de modelos analíticos. A Tabela 3.1 apresenta a comparação desses trabalhos correlatos em relação ao trabalho proposto nesta tese, considerando os formalismos matemáticos adotados para modelagem, os aspectos avaliados e a concepção de uma ferramenta para geração de modelos.

Os trabalhos relacionados *Service performance and analysis in cloud computing* (XIONG; PERROS, 2009) (A1), *Power-performance trade-offs in IaaS cloud: a scalable analy-*

tic approach (GHOSH; NAIK; TRIVEDI, 2011) (A2), *Modeling and performance analysis of large scale IaaS Clouds* (GHOSH et al., 2013) (A3), *Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud* (GHOSH et al., 2013) (A4), *Dependability modeling and analysis for the virtual data center of cloud computing* (WEI B.; KONG, 2011) (A5), *An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism* (DANTAS et al., 2012) (A6), *Models for dependability analysis of cloud computing architectures for Eucalyptus platform* (DANTAS et al., 2012) (A7), *Dependability models for designing disaster tolerant cloud computing systems* (SILVA et al., 2013) (A8), *End-to-end performability analysis for infrastructure-as-a-service cloud: an interacting stochastic models approach* (GHOSH et al., 2010) (A9), *Performability models for designing disaster tolerant Infrastructure-as-a-Service cloud computing systems* (SILVA; MACIEL; ZIMMERMANN, 2013) (A10), *GeoClouds Modcs: A perfomability evaluation tool for disaster tolerant IaaS clouds* (SILVA et al., 2014) (A11) e *The method and tool of cost analysis for cloud computing* (LI et al., 2009) (A12) foram analisados em relação ao trabalho proposto (*Baseline*), considerando as características P1 até P12.

Essas características são a adoção de modelos baseados em cadeias de *Markov* (P1), diagramas de bloco de confiabilidade (P2), redes de *Petri* (P3), redes de filas (P4), modelagem de custo (P5), modelagem de dependabilidade (P6), modelagem de desempenho (P7), modelagem de performabilidade (P8), modelagem de otimização (P9), modelagem hierárquica (P10), modelagem heterogênea (P11) e ferramenta (P12) para o planejamento de infraestruturas de computação em nuvem.

A Tabela 3.1 mostra as características dos doze trabalhos mais relacionados ao trabalho proposto. Essa tabela mostra que sete trabalhos relacionados adotam uma estratégia de modelagem hierárquica e heterogênea, três trabalhos relacionados modelam o desempenho e a dependabilidade, três trabalhos relacionados modelam a performabilidade, apenas um trabalho correlato modela o custo, apenas um trabalho relacionado adota um modelo de otimização e apenas dois trabalhos correlatos adota uma ferramenta para o planejamento da nuvem computacional. O trabalho proposto adota modelos de otimização para geração de cenários de infraestruturas da nuvem computacional. Esses cenários são modelados através de dois formalismos matemáticos e expressões matemáticas. Estes modelos são avaliados para o cálculo de métricas de desempenho, dependabilidade, performabilidade e custo. Além disso, uma ferramenta é proposta para geração automática e avaliação de modelos de desempenho, dependabilidade e custo.

Tabela 3.1: Características dos Trabalhos Relacionados

Artigos	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Baseline	-	X	X	-	X	X	X	X	X	X	X	X
Service performance and analysis in cloud computing (XIONG; PERROS, 2009) (A1)	-	-	-	X	-	-	X	-	-	-	-	-

Power-performance trade-offs in IaaS cloud: a scalable analytic approach (GHOSH; NAIK; TRIVEDI, 2011) (A2)	X	-	-	-	-	-	X	-	-	X	-	-
Modeling and performance analysis of large scale IaaS Clouds (GHOSH et al., 2013) (A3)	X	-	X	-	-	-	X	-	-	X	-	-
Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud (GHOSH et al., 2013) (A4)	X	-	X	-	X	X	X	-	X	X	-	-
Dependability modeling and analysis for the virtual data center of cloud computing (WEI B.; KONG, 2011) (A5)	-	X	X	-	-	X	-	-	-	X	X	-
An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism (DANTAS et al., 2012) (A6)	X	X	-	-	-	X	-	-	-	X	X	-
Models for dependability analysis of cloud computing architectures for Eucalyptus platform (DANTAS et al., 2012) (A7)	X	X	-	-	-	X	-	-	-	X	X	-
Dependability models for designing disaster tolerant cloud computing systems (SILVA et al., 2013) (A8)	-	X	X	-	-	X	-	-	-	X	X	-
End-to-end performability analysis for infrastructure-as-a-service cloud: an interacting stochastic models approach (GHOSH et al., 2010) (A9)	X	-	-	-	-	X	X	X	-	X	-	-
Performability models for designing disaster tolerant Infrastructure-as-a-Service cloud computing systems (SILVA; MACIEL; ZIMMERMANN, 2013) (A10)	-	X	X	-	-	-	-	X	-	X	X	-
GeoClouds Modcs: A perfomability evaluation tool for disaster tolerant IaaS clouds (SILVA et al., 2014) (A11)	-	X	X	-	-	-	-	X	-	X	X	X
The method and tool of cost analysis for cloud computing (LI et al., 2009) (A12)	-	-	-	-	X	-	-	-	-	-	-	X

3.6 Considerações Finais

Este capítulo apresentou os principais trabalhos correlatos ao estudo proposto. Embora existam vários trabalhos na literatura que proporcionam a avaliação de desempenho, a avaliação de dependabilidade ou a avaliação de performabilidade de nuvens computacionais por meio de modelos analíticos, nenhum desses trabalhos foca na avaliação de aspectos de desempenho, dependabilidade, performabilidade e custo através de modelos analíticos e expressões matemáticas. Alguns trabalhos apresentam uma estratégia de modelagem hierárquica e heterogênea para avaliação de dependabilidade de nuvens computacionais, mas poucos trabalhos avaliam o impacto da atribuição de mecanismos de redundância aos componentes da nuvem computacional. Embora existam trabalhos que apresentem uma ferramenta para avaliação de performabilidade ou de custo de nuvens computacionais, nenhum trabalho proporciona uma ferramenta para avaliação de desempenho, dependabilidade, performabilidade e custo de nuvens computacionais.

4

Metodologia e Métodos para Geração e Seleção de Cenários de Computação em Nuvem

Este capítulo apresenta a metodologia e os métodos que compõem a solução integrada proposta para o planejamento de infraestruturas de nuvens privadas. A metodologia proposta consiste de métodos para Geração de Cenários de Computação em Nuvem, Geração de Modelo de Desempenho, Geração de Modelo de Disponibilidade, Geração de Modelo de Custo, Avaliação de Cenários de Computação em Nuvem e de um método de Seleção de Cenários de Computação em Nuvem. Os métodos propostos permitem a confecção do modelo de otimização para geração de cenários de desempenho e custo e do modelo de otimização para geração de cenários de disponibilidade e custo e dos modelos de representação de desempenho, de disponibilidade e de custo. A Figura 4.1 apresenta uma visão de alto nível da metodologia proposta através de uma notação baseada em UML (*Unified Modeling Language*) (LINS, 2012; GROUP, 2013).

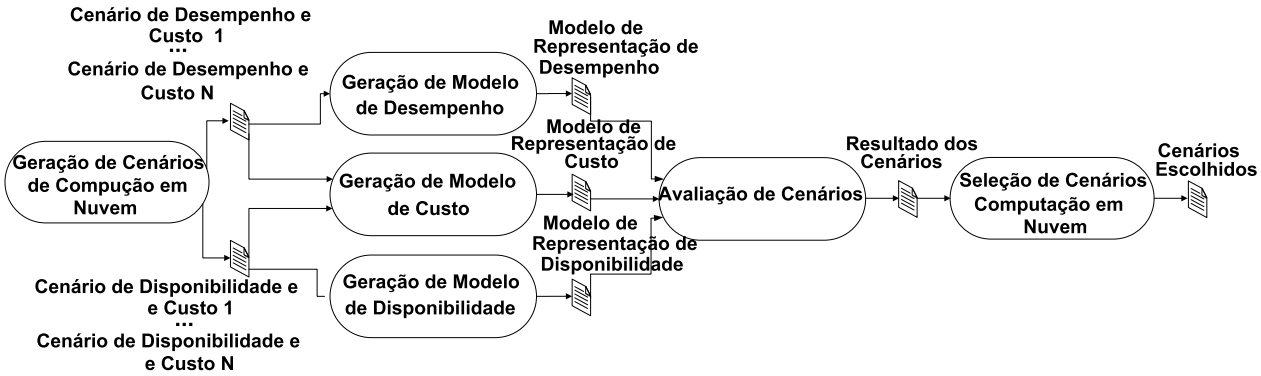


Figura 4.1: Visão de Alto Nível da Metodologia Proposta

A atividade representa a ação que será executada na metodologia e nos métodos. O artefato retrata o resultado da atividade. A decisão controla a sequência de atividades. A conexão provê a ligação entre a atividade e o artefato. A conexão pode indicar que um artefato é gerado por uma atividade ou que um artefato é usado por uma atividade. A Figura 4.2 apresenta as

notações gráficas dos elementos da metodologia e dos métodos propostos.

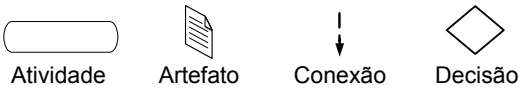


Figura 4.2: Elementos da Metodologia e dos Métodos

A metodologia proposta pode ser usada por projetistas com experiência em técnicas metaheurísticas, modelos combinatoriais e modelos baseados em espaço de estados para o planejamento de infraestruturas de nuvens privadas que atendam aos requisitos dos usuários. Essa metodologia proporciona a geração automática dos modelos de representação de desempenho, de disponibilidade e de custo por meio do Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas (SMG4PCIP) (SOUSA et al., 2013), o qual pode ser usado por projetistas com pouca ou nenhuma experiência em técnicas metaheurísticas e formalismos matemáticos.

O desenvolvimento dessa metodologia deve proporcionar meios para o dimensionamento de recursos computacionais, considerando o tempo de resposta, a utilização de recursos, a disponibilidade, o *downtime* e o custo requeridos.

A Geração de Cenários de Computação em Nuvem elabora diferentes cenários, em que se consideram diferentes conjuntos de *softwares* configurados em várias infraestruturas de nuvens privadas por meio do Modelo de Otimização para Geração de Cenários de Desempenho e Custo. Os cenários gerados são representados através de modelos estocásticos e expressões matemáticas produzidas pela Geração de Modelo de Desempenho e pela Geração de Modelo de Custo. Na Avaliação de Cenários de Computação em Nuvem, esses modelos propiciam a avaliação do efeito de diferentes níveis de carga de trabalho nas métricas de desempenho e custo.

A Geração de Cenários de Computação em Nuvem também propicia a avaliação de cenários em que mecanismos de redundância são atribuídos aos componentes das infraestruturas de nuvens privadas. Esses cenários são gerados através do Modelo de Otimização para Geração de Cenários de Disponibilidade e Custo. Os cenários gerados são representados através de expressões matemáticas e modelos heterogêneos e hierarquicamente combinados que são produzidos na Geração de Modelo de Disponibilidade e na Geração de Modelo de Custo. A Avaliação de Cenários de Computação em Nuvem analisa o impacto de diferentes mecanismos de redundância nas métricas de dependabilidade e custo.

A Seleção de Cenários de Computação em Nuvem combina as métricas de desempenho, de dependabilidade, de performabilidade e de custo para escolher cenários que atendam aos requisitos dos usuários.

4.1 Geração de Cenários

A Geração de Cenários de Computação em Nuvem elabora cenários de infraestruturas de nuvens privadas configuradas com diferentes conjuntos *softwares* e *hardwares* e cenários de

infraestruturas de nuvens privadas com diversos mecanismos de redundância associados aos seus componentes por meio de modelos de otimização baseados na metaheurística GRASP. Os cenários criados são representados através dos modelos concebidos na Geração de Modelo de Desempenho, na Geração de Modelo de Disponibilidade e na Geração de Modelo de Custo.

Esse método é composto de duas atividades: Geração de Cenários de Desempenho e Custo e Geração de Cenários de Disponibilidade e Custo. Todas as atividades desse método devem ser realizadas por projetistas com experiência em técnicas metaheurísticas. A Figura 4.3 apresenta o método de Geração de Cenários.

Figura 4.3: Geração de Cenários

Geração de Cenários de Desempenho e Custo provê a geração de cenários de infraestruturas de nuvens privadas com diferentes configurações de *hardware* e de *software* através do Modelo para Geração de Cenários de Desempenho e Custo (ver Seção 5.2.1). Como exemplo temos cenários configurados com diferentes *softwares* (plataformas de nuvem, sistemas clientes, sistemas operacionais, bancos de dados e servidores *web*) e vários *hardwares* (diferentes dimensionamentos de processamento, memória e memória secundária).

A Tabela 4.1 apresenta quatro exemplos de cenários de desempenho e custo considerando a atribuição de diferentes conjuntos de *software* a um conjunto de *hardware*. A segunda coluna dessa tabela mostra o dimensionamento do processador, memória e memória secundária do conjunto de *hardware* e a terceira coluna apresenta os tipos de plataforma de nuvem, sistema cliente, banco de dados, sistema operacional e servidor *web* dos diferentes conjuntos de *software*.

Tabela 4.1: Cenários de Desempenho e Custo

Cenário	Conjunto de *Hardware*	Conjunto de *Software*
1	2 *Cores*, 2GB, 80GB	*Eucalyptus* (EUCALYPTUS, 2013), *Moodle* (MOODLE, 2013), *MySQL* (MYSQL, 2013), *Ubuntu* (UBUNTU, 2013), *Apache* (APACHE, 2013)
2	2 *Cores*, 2GB, 80GB	*Eucalyptus* (EUCALYPTUS, 2013), *Moodle* (MOODLE, 2013), *MySQL* (MYSQL, 2013), *Ubuntu* (UBUNTU, 2013), *Lighttpd* (LIGHTTPD, 2013)

3	2 *Cores*, 2GB, 80GB	*Eucalyptus* (EUCALYPTUS, 2013), *Moodle* (MOODLE, 2013), *MySQL* (MYSQL, 2013), *CentOS* (CENTOS, 2013), *Apache* (APACHE, 2013)
4	2 *Cores*, 2GB, 80GB	*Eucalyptus* (EUCALYPTUS, 2013), *Moodle* (MOODLE, 2013), *MySQL* (MYSQL, 2013), *CentOS* (CENTOS, 2013), *Lighttpd* (LIGHTTPD, 2013)

Geração de Cenários de Disponibilidade e Custo corresponde a geração de cenários de infraestruturas de nuvens privadas com diferentes mecanismos de redundância atribuídos aos seus componentes por meio do Modelo para Geração de Cenários de Disponibilidade e Custo (ver Seção 5.2.2). Como exemplo temos cenários com os mecanismos de redundância (*hot standby*, *cold standby*, *warm standby*, $N+1$ e nenhum mecanismo de redundância) associados (RUPE, 2003) aos componentes (controlador de nuvem, controlador de *cluster*, controlador de nó, máquina virtual, *switch* e roteador para a plataforma *Eucalyptus*) (EUCALYPTUS, 2013) da nuvem computacional.

A Tabela 4.2 apresenta cinco exemplos de cenários de disponibilidade e custo considerando a atribuição de mecanismos de redundância aos componentes da plataforma *Eucalyptus* (EUCALYPTUS, 2013). A primeira coluna dessa tabela mostra os componentes da nuvem computacional (controlador de nuvem - CLC, controlador de *cluster* - CC, controlador de nó - NC, máquina virtual - VM, *switch* - SW e roteador - RT) e a segunda coluna apresenta os tipos de mecanismos de redundância (*cold standby* - *cold*, *hot standby* - *hot*, *warm standby* - *warm*, Ativo-Ativo - AA e nenhum mecanismo de redundância - nen) atribuídos sequencialmente aos componentes da nuvem computacional.

Tabela 4.2: Cenários de Disponibilidade e Custo

Cenário	Componentes	Mecanismos de Redundância
1	CLC, CC, NC, VM, RT, SW	Nen, *Warm*, *Cold*, *Hot*, Nen, AA
2	CLC, CC, NC, VM, RT, SW	AA, *Warm*, *Cold*, *Hot*, Nen, *Warm*
3	CLC, CC, NC, VM, RT, SW	*Warm*, *Warm*, *Cold*, *Hot*, Nen, *Warm*
4	CLC, CC, NC, VM, RT, SW	*Hot*, AA, *Cold*, *Hot*, Nen, *Warm*
5	CLC, CC, NC, VM, RT, SW	AA, Nen, *Cold*, *Hot*, Nen, *Warm*

4.2 Geração de Modelo de Desempenho

A Geração de Modelo de Desempenho é responsável pela medição e modelagem dos cenários concebidos na atividade de Geração de Cenários de Desempenho e Custo do método de Geração de Cenários de Computação em Nuvem. Na fase de medição, esses cenários são entendidos, seus componentes de *software* são configurados, eles são submetidos a diferentes níveis de carga de trabalho e o impacto dessa carga de trabalho é medido. Na fase de modelagem, esses cenários são representados por meio de modelos SPN (GERMAN, 2000). Esses modelos SPN são analisados, refinados e validados. Os modelos SPN concebidos para avaliação de infraestruturas de nuvens privadas serão apresentados na Seção 5.1.1.

Esse método é composto de onze atividades: Entendimento do Sistema, Preparação do Ambiente, Geração de Carga de Trabalho, Medição, Obtenção das Métricas de Desempenho, Geração de Modelo Abstrato, Análise Qualitativa do Modelo Abstrato, Geração de Modelo Refinado, Análise Qualitativa do Modelo Refinado, Mapeamento de Métricas de Desempenho e Análise Quantitativa do Modelo Refinado. As atividades de Entendimento do Sistema, Preparação do Ambiente, Geração de Carga de Trabalho, Medição e Obtenção das Métricas de Desempenho compreendem a fase de medição e devem ser realizadas por projetistas com experiência em ferramentas de medição de desempenho. Já as atividades de Geração de Modelo Abstrato, Análise Qualitativa do Modelo Abstrato, Geração de Modelo Refinado, Análise Qualitativa do Modelo refinado, Mapeamento de Métricas de Desempenho e Análise Quantitativa do Modelo Refinado compreendem a fase de modelagem e podem ser realizadas por projetistas com experiência em modelos baseados em espaço de estados. A Figura 4.4 apresenta o método de Geração de Modelo de Desempenho.

Entendimento do Ambiente compreende a identificação das características da plataforma de nuvem, dos seus componentes (como exemplos temos as plataformas *Eucalyptus* (EUCALYPTUS, 2013), *Nimbus* (PROJECT, 2013a), *OpenNebula* (PROJECT, 2013b) e *OpenStack* (PROJECT, 2013c)), e dos componentes do sistema cliente. Essa atividade também considera os critérios de desempenho que serão examinados no processo de avaliação. Dentre esse critérios temos os requisitos de desempenho que devem ser analisadas e o impacto de determinados níveis de solicitações nos referidos requisitos de desempenho.

Obtenção das Métricas de Desempenho compreende a obtenção de métricas de desempenho (ex: tempo de resposta) da nuvem computacional através de dados históricos. Quando não há dados históricos, as métricas de desempenho são obtidas através das atividades Preparação do Ambiente, Geração de Carga de Trabalho e Medição.

Preparação do Ambiente proporciona a configuração da plataforma de nuvem e do sistema cliente com base na identificação das características dos seus componentes e dos requisitos de desempenho que devem ser considerados no processo de avaliação.

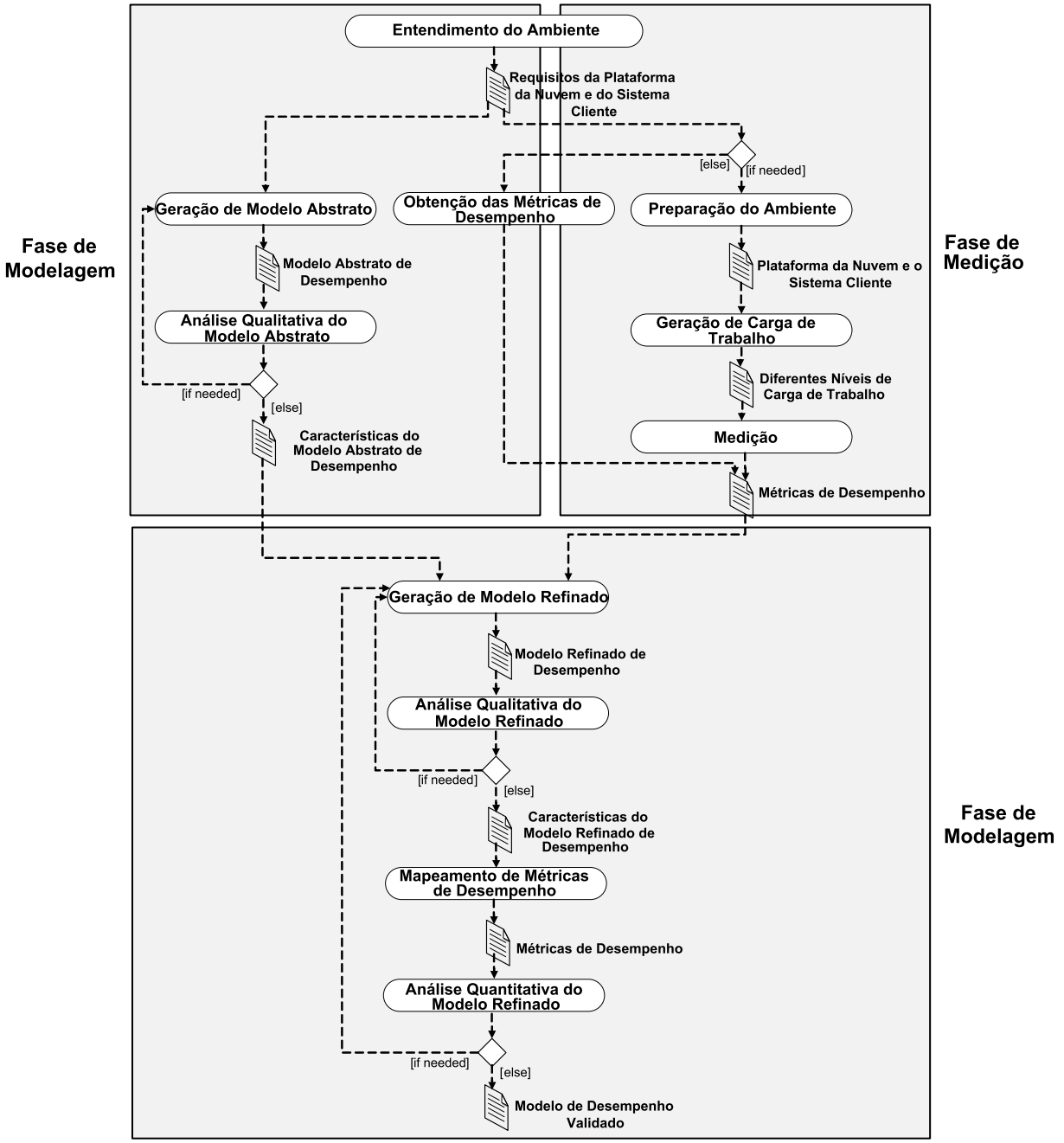


Figura 4.4: Geração de Modelo de Desempenho

Geração de Carga de Trabalho define os tipos e os níveis de carga de trabalho que devem ser considerados para avaliação de infraestruturas de nuvem.

Medição define o processo de medição, coleta de dados, análise de dados e tratamento estatístico dos dados (JAIN, 1991).

- O processo de medição examina quais os componentes da nuvem computacional e respectivos recursos computacionais devem ser analisados (ex: CLC, CC, NC da plataforma *Eucalyptus* (EUCALYPTUS, 2013)) e as métricas de desempenho (ex: tempo de resposta) que são representativas para avaliação do ambiente de nuvem computacional.

- O processo de coleta de dados define a ferramenta de medição (ex: *Sysstat* para *Linux* (GODARD, 2013), *Microsoft Performance Monitor* para *Windows* (FRIEDMAN, 2005)), o quantitativo de métricas de desempenho que devem ser avaliadas concomitantemente, o tempo de coleta dos dados, a frequência de coleta dos dados, a forma de armazenamento do dados (ex: blg, csv, xml) (GODARD, 2013; FRIEDMAN, 2005) e o local de armazenamento dos dados.

- O processo de análise de dados corresponde à aplicação de métodos estatísticos nos dados coletados com o objetivo de tratar os *outliers* e fornecer informações precisas a respeito do ambiente de nuvem computacional avaliado. As principais estatísticas de interesse são a média (μ_D), o desvio padrão (σ_D) e o inverso do coeficiente de variação.

- O processo de tratamento estatístico dos dados utiliza o *moment matching* por meio do cálculo do inverso do coeficiente de variação da distribuição empírica (dados coletados) para seleção de uma distribuição expolinomial (ex: hipoexponencial, hiperexponencial e *Erlang*) que melhor se adapta à distribuição empírica (DESROCHERS; AL-JAAR, 1995).

Geração do Modelo Abstrato corresponde a geração do modelo de desempenho (ver Seção 5.1.1) que é utilizado para avaliar o comportamento da nuvem computacional e estimar o seu desempenho quando submetido a diferentes níveis de carga de trabalho ou variações na infraestrutura de nuvem computacional. Os modelos podem ser expressos em diferentes níveis de representação. Esses níveis de detalhamento na representação dos componentes da nuvem computacional determinam as análises que podem ser realizadas (MENASCÉ; ALMEIDA, 2005). Esse trabalho modela o envio da carga de trabalho, os módulos de gerenciamento da nuvem computacional e as infraestruturas de processamento e armazenamento da nuvem computacional.

Análise Qualitativa do Modelo Abstrato corresponde a análise das propriedades qualitativas do modelo de desempenho através da ferramenta INA (INA, 2013). As propriedades de interesse são a alcançabilidade, a limitação, e se o modelo é livre de *deadlocks* (MURATA, 1989). No processo de análise, pode-se observar a necessidade de ajustes no modelo de desempenho. Após os ajustes, esse modelo deve ser novamente analisado. Após a atividade de Análise Qualitativa do Modelo Abstrato e da atividade de Medição ou da atividade de Obtenção das Métricas de Desempenho, o modelo refinado de desempenho deve ser concebido, e posteriormente, analisado qualitativamente.

Geração do Modelo Refinado proporciona o refinamento do modelo de desempenho de forma que este represente os dados medidos ou obtidos do sistema configurado. Esta atividade corresponde à geração do modelo refinado de desempenho em função do modelo de desempenho e das estatísticas obtidas na atividade de Medição ou das métricas de desempenho coletadas na atividade de Obtenção das Métricas de Desempenho. Essas estatísticas sugerem o tipo

de distribuição expolinomial que melhor representa a distribuição empírica (dados coletados). Essa adequação é realizada com o auxílio do *moment matching* (DESROCHERS; AL-JAAR, 1995) através do cálculo do inverso do coeficiente de variação da distribuição empírica. Esse cálculo provê a seleção da distribuição expolinomial e a obtenção dos parâmetros numéricos da distribuição expolinomial escolhida.

Análise qualitativa do modelo refinado corresponde a análise das propriedades qualitativas do modelo refinado de desempenho através da ferramenta INA (INA, 2013). Esta atividade possibilita a análise qualitativa do modelo de desempenho após a modificação estrutural causada pelo seu refinamento. As propriedades de interesse são a alcançabilidade, a limitação, e se o modelo é livre de *deadlocks* (MACIEL; LINS; CUNHA, 1996; MURATA, 1989). No processo de análise, pode-se observar a necessidade de ajustes no modelo refinado de desempenho. Após os ajustes, esse modelo deve ser novamente analisado.

Mapeamento das métricas de Desempenho corresponde ao processo de representação do conjunto de critérios de desempenho em métricas através de referências aos elementos do modelo refinado de desempenho concebido. Como exemplo dessas métricas temos o tempo de resposta.

Análise quantitativa do modelo refinado analisa se os resultados das métricas de desempenho calculadas pelo modelo refinado de desempenho são comparáveis às métricas de desempenho obtidas através de medições na nuvem computacional, considerando um erro de exatidão aceitável. Se os resultados obtidos das medições e do modelo refinado de desempenho forem comparáveis, esse modelo será usado para representação dos cenários de infraestruturas de nuvens. Caso contrário, o modelo refinado de desempenho precisará de ajustes.

4.3 Geração de Modelo de Disponibilidade

A Geração de Modelo de Disponibilidade proporciona a modelagem dos cenários concebidos na atividade de Geração de Cenários de Disponibilidade e Custo do método de Geração de Cenários de Computação em Nuvem. Na Geração de Modelo de Disponibilidade, os cenários gerados são entendidos, e os parâmetros de dependabilidade dos equipamentos e as métricas de dependabilidade são obtidas. Esse método utiliza uma modelagem hierárquica e heterogênea baseada em redes de *Petri* estocásticas (SPNs) (BOLCH et al., 2006; GERMAN, 2000) e diagramas de bloco de confiabilidade (RBDs) (XIE; DAI; POH, 2004). Modelos estocásticos de baixo nível baseados em RBD representam os componentes da infraestrutura da nuvem privada. Esses modelos são agrupados de forma a representar os subsistemas da infraestrutura da nuvem privada. E os modelos estocásticos de alto nível baseados em SPNs e RBDs representam os sistemas da infraestrutura da nuvem privada. Os modelos SPN podem ser analisados e refinados. A estratégia de modelagem é validada. Os modelos RBD e SPN concebidos para avaliação de infraestruturas de computação em nuvem serão apresentados na Seção 5.1.2.

A Geração de Modelo de Disponibilidade é composta de nove atividades: Entendimento

do Ambiente, Obtenção dos Parâmetros e Métricas de Dependabilidade, Agrupamento de Componentes, Geração de Modelo do Subsistema, Geração de Modelo RBD do Sistema, Geração de Modelo SPN do Sistema, Análise Qualitativa do Modelo SPN do Sistema, Mapeamento de Métricas de Dependabilidade e Análise Quantitativa da Estratégia de Modelagem. Todas as atividades desse método compreendem a fase de modelagem e devem ser realizadas por projetistas com experiência em modelos combinatoriais e modelos baseados em espaço de estados. A Figura 4.5 apresenta o método de Geração de Modelo de Disponibilidade de infraestruturas de computação em nuvem.

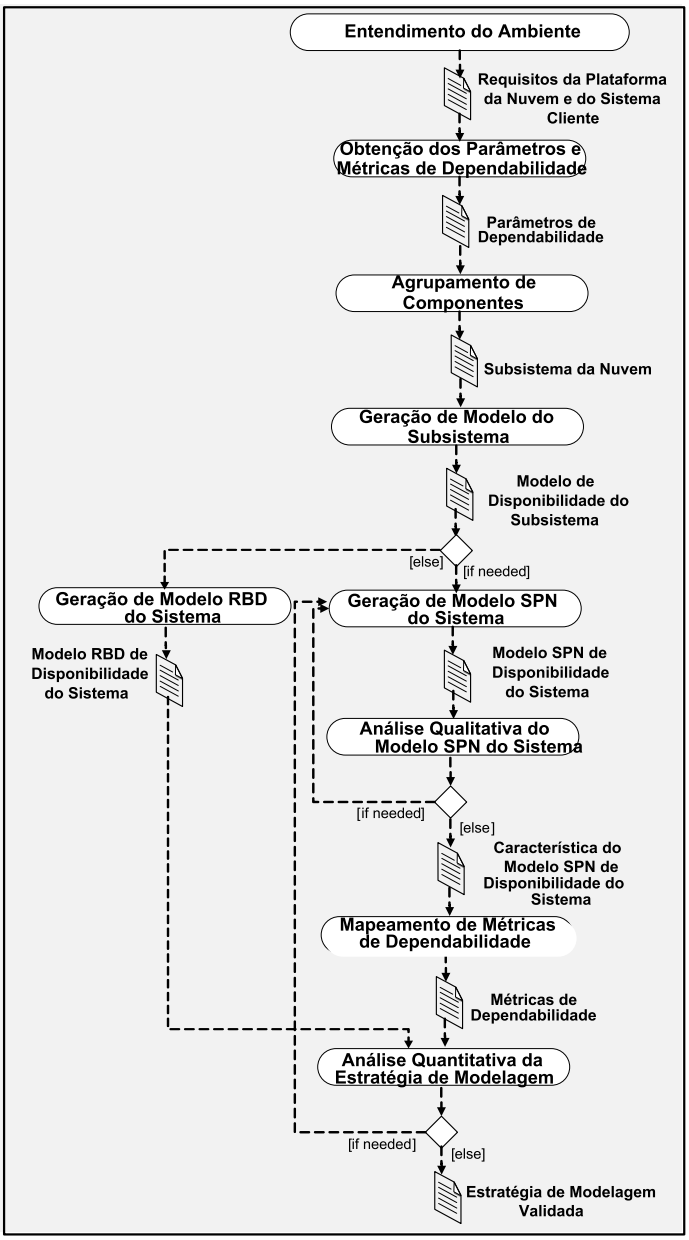


Figura 4.5: Geração de Modelo de Disponibilidade

Entendimento do Ambiente identifica as características das plataformas de nuvem, seus componentes e os do sistema cliente. Essa atividade também define os requisitos de dependabilidade que devem ser estudados no processo de avaliação. Esses requisitos de dependabilidade são

o estudo dos componentes críticos da plataforma da nuvem e do sistema cliente, e os mecanismos de redundância que devem ser usados para maximizar a dependabilidade do sistema cliente hospedado na nuvem computacional (ex: mecanismo de redundância do tipo ativo-ativo, *hot standby*, *cold standby* e *warm standby*) (RUPE, 2003).

Obtenção dos Parâmetros e Métricas de Dependabilidade corresponde à obtenção dos parâmetros e das métricas de dependabilidade dos mecanismos de redundância, dos componentes da nuvem computacional e do sistema cliente por meio de dados históricos ou de dados de fabricantes. Os parâmetros de dependabilidade são o tempo médio para a falha do equipamento e o tempo médio para o acionamento dos mecanismos de redundância. Já as métricas de dependabilidade são a disponibilidade e o *downtime*.

Agrupamento de Componentes corresponde à combinação dos componentes da nuvem computacional e do sistema cliente em subsistemas, com objetivo de minimizar a complexidade do sistema modelado. Um exemplo da combinação de componentes da plataforma *Eucalyptus* em subsistemas é a geração do sistema computacional através da combinação dos componentes processador, memória e memória secundária.

Geração de Modelo do Subsistema proporciona a geração de modelos de disponibilidade de baixo nível para representar os componentes da infraestrutura de computação em nuvem e do sistema cliente, com base em diagramas de bloco de confiabilidade (XIE; DAI; POH, 2004). Esses modelos são agrupados para representar os subsistemas da nuvem computacional. Os resultados dos modelos de subsistemas são usados como parâmetros de dependabilidade para os modelos abstratos do sistema e para os modelos do sistema.

Geração de Modelo RBD do Sistema corresponde à geração de modelos de disponibilidade de alto nível para representar os sistemas da nuvem computacional, com base em diagramas de bloco de confiabilidade (XIE; DAI; POH, 2004). Esses modelos RBD (ver Seção 5.1.2) são utilizados para estimar a disponibilidade e o *downtime* da nuvem computacional quando há pouca relação de dependência entre os componentes desse ambiente e os mecanismos de redundância empregados. Caso haja a necessidade de representar uma maior relação de dependência entre os componentes da nuvem computacional e os mecanismos de redundância usados, modelos SPN são usados para representar os sistemas da nuvem computacional.

Geração de Modelo SPN do Sistema corresponde à geração de modelos de disponibilidade de alto nível para representar os sistemas da nuvem computacional, com base em redes de *Petri* estocásticas (ver Seção 5.1.2) (GERMAN, 2000).

Análise Qualitativa de Modelo SPN do Sistema corresponde à análise das propriedades qualitativas do modelo SPN do sistema. As propriedades analisadas são a alcançabilidade, a limitação, e se o modelo é livre de *deadlocks* (MACIEL; LINS; CUNHA, 1996; MURATA, 1989). Essa atividade ocorre após a geração do modelo SPN do sistema. No processo de análise, pode-se observar a necessidade de ajustes no modelo SPN do sistema. Após os ajustes, esse modelo deve ser novamente analisado.

Mapeamento de Métricas de Dependabilidade corresponde ao processo de representa-

ção do conjunto de critérios de dependabilidade em métricas através de referências aos elementos do modelo SPN do sistema. Como exemplo dessas métricas temos a disponibilidade.

Análise Quantitativa da Estratégia de Modelagem analisa se os resultados das métricas de dependabilidade calculados usando a estratégia de modelagem hierárquica e heterogênea proposta são comparáveis às métricas de dependabilidade obtidas através da atividade de Obtenção dos Parâmetros e Métricas de Dependabilidade, considerando um erro de exatidão aceitável. Essa análise quantitativa também pode ser realizada através da comparação entre os resultados das métricas de dependabilidade calculados usando a estratégia de modelagem proposta e uma estratégia de modelagem já validada, de acordo com um erro de exatidão aceitável. Se os resultados obtidos nas duas estratégias de modelagem forem comparáveis, a estratégia de modelagem de disponibilidade proposta será usada para representação dos cenários de infraestruturas de nuvens. Caso contrário, o modelo SPN do sistema precisará de ajustes.

4.4 Geração de Modelo de Custo

A Geração de Modelo de Custo provê a modelagem dos cenários concebidos na atividade de Geração de Cenários de Desempenho e Custo e na atividade de Geração de Cenários de Disponibilidade e Custo do método de Geração de Cenários de Computação em Nuvem. Os modelos de custo dos cenários gerados são concebidos através da criação de expressões matemáticas e do mapeamento de métricas de custo nos modelos de desempenho e nos modelos de disponibilidade. Os modelos de custo concebidos para avaliação de infraestruturas de computação em nuvem serão apresentados na Seção 5.1.3.

Esse método é composto de três atividades: Entendimento do Ambiente, Mapeamento de Métricas de Custo e Geração de Modelo de Custo. O Mapeamento de Métricas de Custo deve ser realizada por um projetistas com experiência em modelagem e a atividade Geração de Modelo de Custo deve ser realizado por projetistas com experiência em baseados em espaço de estados. A Figura 4.6 apresenta o método de Geração de Modelo de Custo.

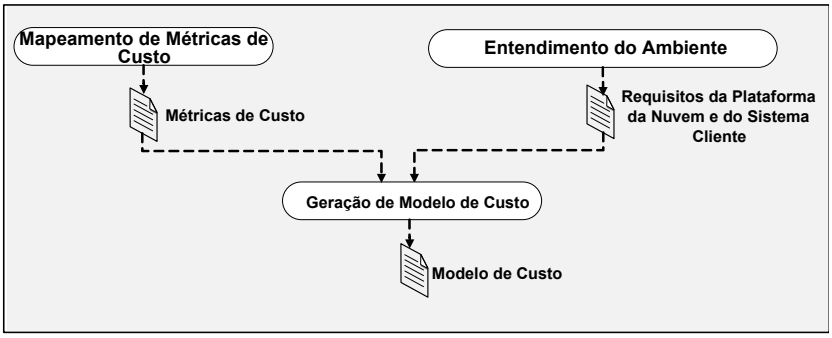


Figura 4.6: Geração de Modelo de Custo

Entendimento do Ambiente identifica as características das plataformas de nuvem, dos seus componentes e dos componentes do sistema cliente. Essa atividade também define os

requisitos de custo que devem ser estudados no processo de avaliação. Esses requisitos de custo estão relacionados à aquisição de equipamentos do sistema de TI, à aquisição de equipamentos e *softwares* redundantes, à substituição de equipamentos, à aquisição de licenças de *software*, à equipe técnica e à equipe de manutenção.

Mapeamento de Métricas de Custo corresponde à representação dos critérios de custo em métricas através de referências aos elementos do modelo refinado de desempenho e do modelo SPN de disponibilidade do sistema.

Geração do Modelo de Custo corresponde à criação de expressões matemáticas e métricas de custo (ver Seção 5.1.3) para calcular o custo da aquisição de equipamentos do sistema de TI, da aquisição de equipamentos e *softwares* redundantes, da substituição de equipamentos, da aquisição de licenças de *software*, da equipe técnica e da equipe de manutenção.

4.5 Avaliação de Cenários de Computação em Nuvem

A Avaliação de Cenários de Computação em Nuvem provê a avaliação dos modelos de desempenho, disponibilidade e custo que representam os cenários gerados pelo método de Geração de Cenários de Computação em Nuvem. Os resultados da avaliação desses modelos são normalizados e usados para seleção dos cenários gerados.

Esse método é composto de sete atividades: Análise de Cenários de Desempenho e Custo, Análise de Cenários de Disponibilidade e Custo, Avaliação de Performabilidade e Custo, Estratégia de Decomposição e Composição, Normalização dos Resultados dos Cenários de Desempenho e Custo, Normalização dos Resultados dos Cenários de Disponibilidade e Custo e Normalização dos Resultados de Performabilidade e Custo. Todas as atividades desse método devem ser realizadas por projetistas com experiência em modelos combinatoriais e modelos baseados em espaço de estados. A Figura 4.7 apresenta o método de Avaliação de Cenários de Computação em Nuvem.

Avaliação de Cenários de Desempenho e Custo corresponde à análise dos modelos de desempenho e de custo providos pelo método de Geração de Modelo de Desempenho e pelo método de Geração de Modelo de Custo. Essa análise provê as métricas tempo de resposta (RT), utilização de processador (UP), utilização de memória (UM), custo da aquisição de equipamentos do sistema de TI (CSTI), custo da aquisição de licenças de *software* (CLS) e custo da equipe técnica (CET).

Avaliação de Cenários de Disponibilidade e Custo corresponde à análise dos modelos de disponibilidade e de custo providos pelo método de Geração de Modelo de Disponibilidade e pelo método de Geração de Modelo de Custo. Essa análise provê as métricas disponibilidade (A), *downtime* (D), custo da aquisição de equipamentos e *softwares* redundantes (CESR), custo da substituição de equipamentos (CSE) e custo da equipe de manutenção (CEM).

Estratégia de Decomposição e Composição combina o resultado de um modelo de disponibilidade de alto nível, o qual representa os cenários de disponibilidade e custo, aos

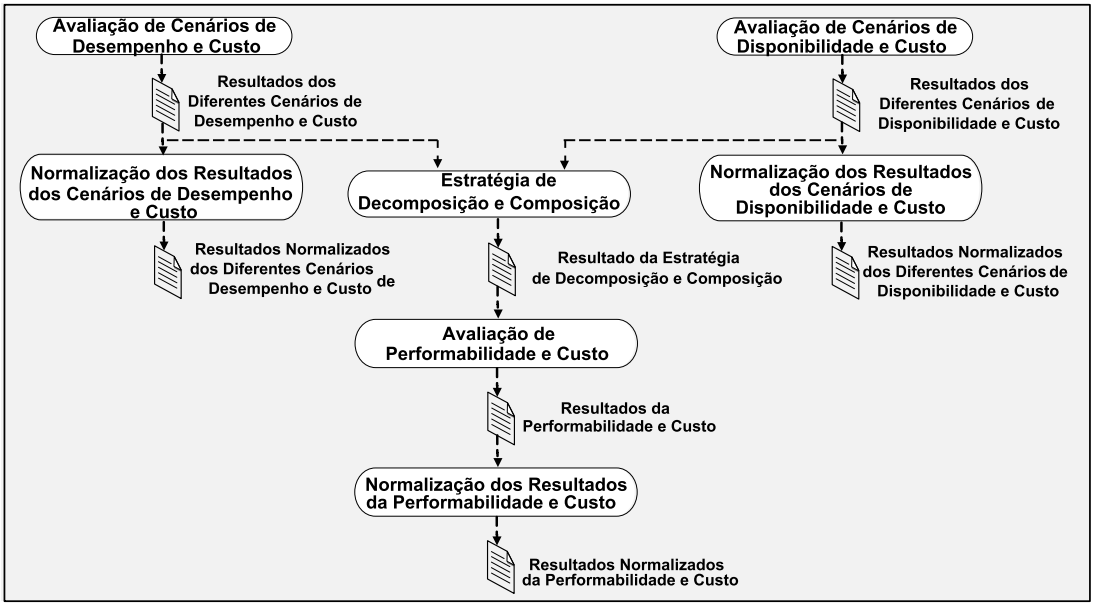


Figura 4.7: Avaliação de Cenários de Computação em Nuvem

resultados de um conjunto de modelos de desempenho de baixo nível que representam os cenários de desempenho e custo.

Avaliação de Performabilidade e Custo corresponde à composição dos resultados da avaliação de cenários de desempenho e custo e dos resultados da avaliação de cenários de disponibilidade e custo. Essa composição provê as métricas de performabilidade, as quais são calculadas, independentemente, a partir dos modelos de desempenho e de disponibilidade e posteriormente combinadas para mostrar o efeito da dependabilidade no desempenho da nuvem computacional. Essas métricas de performabilidade são o tempo de resposta (RT_p), a utilização de processador (UP_p) e a utilização de memória (UM_p).

Normalização dos Resultados dos Cenários de Desempenho e Custo corresponde à normalização das métricas de desempenho e custo por meio da Equação 4.1 (GUIMARÃES, 2013; MONTGOMERRY; GEORGE, 2009). Essas métricas normalizadas são o tempo de resposta (RT_i), a utilização de processador (UP_i), a utilização de memória (UM_i), o custo da aquisição de equipamentos do sistema de TI ($CSTI_i$),o custo da aquisição de licenças de *software* (CLS_i) e o custo da equipe técnica (CET_i).

$$m_{ni} = (m_{sni} - m_{mn})/(m_{mx} - m_{mn}) \tag{4.1}$$

onde:

m_{ni} é a i-ésima medida normalizada, tal que $0 \leq m_{ni} \leq 1$;

m_{sni} é a i-ésima medida obtida sem normalização, tal que $m_i \in R$;

m_{mn} é o valor mínimo da medida, tal que $m_{mn} \in R$;

m_{mx} é o valor máximo da medida, tal que $m_{mx} \in R$.

Normalização dos Resultados dos Cenários de Disponibilidade e Custo corresponde à normalização das métricas disponibilidade e custo por meio da Equação 4.1. Essas métricas normalizadas são a disponibilidade (A_i), *downtime* (D_i), custo da aquisição de equipamentos e *softwares* redundantes ($CESR_i$), custo da substituição de equipamentos (CSE_i) e custo da equipe de manutenção (CEM_i).

Normalização dos Resultados de Performabilidade e Custo corresponde à normalização das métricas de performabilidade e custo por meio da Equação 4.1 (GUIMARÃES, 2013; MONTGOMERRY; GEORGE, 2009). Essas métricas normalizadas são o tempo de resposta (RT_{pi}), utilização de processador (UP_{pi}), utilização de memória (UM_{pi}), custo da aquisição de equipamentos do sistema de TI (CSTI), custo da aquisição de licenças de *software* (CLS), custo da equipe técnica (CET), custo da aquisição de equipamentos e *softwares* redundantes ($CESR_i$), custo da substituição de equipamentos (CSE_i) e custo da equipe de manutenção (CEM_i).

A normalização das métricas de desempenho, dependabilidade, performabilidade e custo proporcionam a uniformidade aos resultados das avaliações dos cenários das infraestruturas de nuvens computacionais.

4.6 Seleção de Cenários de Computação em Nuvem

A Seleção de Cenários de Computação em Nuvem provê a escolha de cenários de infraestruturas de nuvens privadas conforme os resultados das métricas normalizadas dos modelos de desempenho, de disponibilidade e de custo que representam infraestruturas de nuvens privadas configuradas com diferentes *softwares* e *hardwares* e infraestruturas de nuvens privadas com diversos mecanismos de redundância associados aos seus componentes.

Esse método é composto de duas atividades: Seleção de Cenários de Desempenho e Custo e Seleção de Cenários de Disponibilidade e Custo. Todas as atividades desse método devem ser realizadas por projetistas com experiência em modelos combinatoriais e modelos baseados em espaço de estados e projetistas com experiência em técnicas metaheurísticas. Estes projetistas devem selecionar os cenários de desempenho e custo e os cenários de dependabilidade e custo que sejam viáveis e atendam aos requisitos dos usuários. A Figura 4.8 apresenta o método de Seleção de Cenários de Computação em Nuvem.

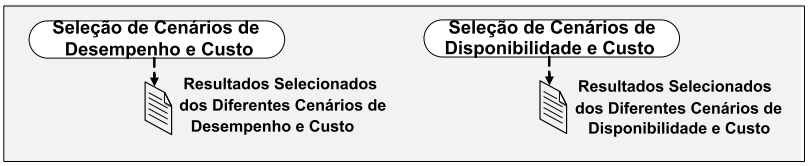


Figura 4.8: Seleção de Cenários de Computação em Nuvem

Seleção de Cenários de Desempenho e Custo apresenta os cenários de infraestruturas de nuvem escolhidos com base nos resultados normalizados das avaliações dos modelos de

desempenho e de custo. Os valores das métricas normalizadas desses modelos devem atender aos requisitos estabelecidos previamente com os clientes da nuvem computacional, as quais são as métricas de desempenho, de performabilidade e de custo.

Seleção de Cenários de Disponibilidade e Custo apresenta os cenários de infraestruturas de nuvem escolhidos conforme os resultados normalizados das avaliações dos modelos de disponibilidade e de custo. Os valores das métricas normalizadas desses modelos devem estar de acordo com os requisitos estabelecidos previamente.

4.7 Considerações Finais

Esse capítulo apresentou a metodologia e os métodos que compõem a solução integrada proposta para o planejamento de infraestruturas de nuvens privadas. Os métodos apresentados proporcionam insumos para geração de cenários de infraestruturas de nuvens através de modelos de otimização. Esses cenários são representados através de modelos de representação. Os modelos de representação de desempenho, de disponibilidade e de custo e os modelos de otimização de desempenho e custo e de disponibilidade e custo serão apresentados no próximo capítulo. Os resultados da avaliação dos modelos de representação são usados pelos modelos de otimização para seleção das infraestruturas de nuvens que atendem aos requisitos de desempenho, dependabilidade e custo. O Capítulo 6 apresentará a ferramenta proposta para a implementação da metodologia apresentada neste capítulo.

5

Modelos de Representação e Modelos de Otimização

Este capítulo apresenta os modelos de representação e os modelos de otimização que compõem a solução integrada proposta para o planejamento de infraestruturas de nuvens privadas. A solução integrada adotada propicia o planejamento de infraestruturas de computação em nuvem por meio de modelos hierárquicos e heterogêneos baseados em redes de *Petri* estocásticas (SPNs) (GERMAN, 2000), diagramas de bloco de confiabilidade (RBDs) (XIE; DAI; POH, 2004) e equações de custo. Inicialmente, os modelos de representação de desempenho, de disponibilidade e de custo adotados na solução integrada proposta serão apresentados. Finalmente, os modelos de otimização propostos para o planejamento de infraestruturas de nuvens computacionais serão introduzidos.

5.1 Modelos de Representação

Esta seção apresenta os modelos de desempenho, de disponibilidade e de custo.

5.1.1 Modelo de Desempenho

Esta seção apresenta o modelo SPN concebido para avaliação de desempenho de infraestruturas de nuvens privadas e os submodelos que compõem esse modelo (ver Seção 4.2) (SOUSA et al., 2014a). Os submodelos Cliente, Memória e Infraestrutura de Processamento descrevem os envios de requisições dos clientes e as infraestruturas de processamento e armazenamento das nuvens privadas que atenderão as requisições desses clientes.

O modelo de desempenho proposto permite a análise do comportamento de infraestruturas de nuvens privadas com diferentes configurações de *hardware* e *software*. A variação no nível da carga de trabalho pode ser ocasionada por uma variação no número de usuários, no tipo e no número de atividades que são solicitadas ao sistema. Esse modelo de desempenho também possibilita a análise do impacto da submissão de diferentes níveis de carga de trabalho às infraestruturas de processamento e de armazenamento da nuvem privada.

Modelo Cliente

O Modelo Cliente (ver Figura 5.1) representa o envio das requisições dos usuários ao serviço ofertado pelo sistema hospedado na nuvem privada. Esse modelo pode representar diferentes tipos e níveis de cargas de trabalho gerados para a infraestrutura de nuvem.

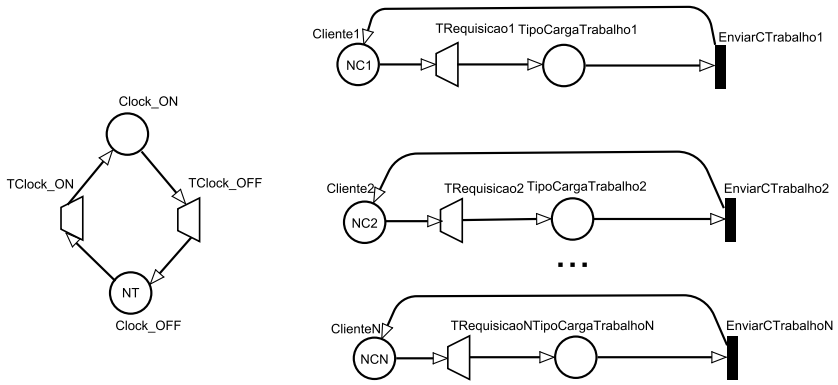


Figura 5.1: Modelo Cliente

As marcações $NC1$, $NC2$ e NCN são atribuídas aos lugares $Cliente1$, $Cliente2$ e $ClienteN$, respectivamente, para representar o número de usuários, e as transições temporizadas $TRequisicao1$, $TRequisicao2$ e $TRequisicaoN$ representam os tempos entre envios de requisições dos usuários ($T_{Requisicao}$) de um determinado tipo e os lugares $TipoCargaTrabalho1$, $TipoCargaTrabalho2$ e $TipoCargaTrabalhoN$ representam os diferentes tipos de requisições dos usuários. As transições temporizadas $TRequisicao1$, $TRequisicao2$ e $TRequisicaoN$ apresentam uma forma trapezoidal para mostrar que os tempos ($T_{Requisicao}$) entre envios de requisições dos usuários podem seguir o comportamento de qualquer distribuição de probabilidade. Como exemplo dos diferentes tipos de requisições dos usuários para o ambiente virtual de aprendizagem *Moodle* (MOODLE, 2013), temos as requisições referentes às atividades *chat*, fórum, glossário e *quiz*.

As transições temporizadas $TClock\_ON$ e $TClock\_OFF$ representam os intervalos de tempo TON_{Clock} em que as requisições dos usuários são enviadas ao serviço ofertado pelo sistema hospedado na nuvem privada e os intervalos de tempo $TOFF_{Clock}$ em que as requisições não são enviadas, respectivamente. As transições temporizadas $TClock\_ON$ e $TClock\_OFF$ apresentam uma forma trapezoidal para mostrar que os intervalos de tempo em que as requisições dos usuários são enviadas TON_{Clock} e os intervalos de tempo em que as requisições dos usuários não são enviadas $TOFF_{Clock}$ podem seguir o comportamento de qualquer distribuição de probabilidade.

A marcação NT atribuída ao lugar $Clock\_OFF$ representa o acionamento do intervalo de tempo TON_{Clock} em que as requisições dos usuários são enviadas e marcação NT atribuída ao lugar $Clock\_OFF$ representa a interrupação desse intervalo de tempo.

A função de habilitação ($\{\#Clock\_ON = 1\}$) atribuída às transições imediatas $EnviarCTrabalho1$, $EnviarCTrabalho2$ e $EnviarCTrabalhoN$ proporciona o envio das requisições durante o intervalo de tempo TON_{Clock} atribuído à transição temporizada $TClock\_ON$.

Os pesos atribuídos às transições imediatas $EnviarCTrabalho1$, $EnviarCTrabalho2$ e $EnviarCTrabalhoN$ permitem a escolha dos tipos de requisições que serão enviadas para o serviço hospedado na nuvem privada com determinada probabilidade. As transições imediatas que tiverem maiores pesos serão disparadas com uma maior probabilidade. Os atributos das transições do Modelo Cliente (ver Figura 5.1) são apresentados na Tabela 5.1. Essas transições são imediatas (im) ou exponenciais (exp) e apresentam um grau de concorrência com a semântica *single server* (SS).

Tabela 5.1: Atributos das Transições - Modelo Cliente

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
$EnviarCTrabalho1$	im	-	1	1	-
$EnviarCTrabalho2$	im	-	1	1	-
$EnviarCTrabalhoN$	im	-	1	1	-
$TRequisicao1$	exp	$T_{Requisicao1}$	-	-	SS
$TRequisicao2$	exp	$T_{Requisicao2}$	-	-	SS
$TRequisicaoN$	exp	$T_{RequisicaoN}$	-	-	SS
$TClock\_ON$	exp	TON_{Clock}	-	-	SS
$TClock\_OFF$	exp	$TOFF_{Clock}$	-	-	SS

Modelo Memória

O Modelo Memória (ver Figura 5.2) representa a infraestrutura de armazenamento usada para instanciação de máquinas virtuais com várias configurações na nuvem privada. A marcação *NMP* atribuída ao lugar *TamanhoMemoria* representa a quantidade de memória total destinada à instanciação da máquina virtual. Cada marcação do lugar *Memoria* representa a quantidade de memória usada para o atendimento de uma requisição dos usuários. Como um exemplo, o valor 4 atribuído à *NMP* significa que a memória necessária para execução da máquina virtual é 4GB. Neste caso, cada marca representa 1GB. Os atributos das transições do Modelo Memória (ver Figura 5.2) são apresentados na Tabela 5.2.

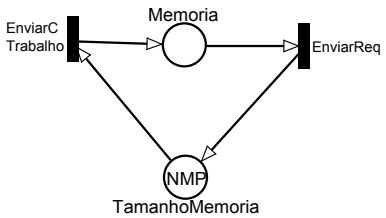


Figura 5.2: Modelo Memória

Tabela 5.2: Atributos das Transições - Modelo Memória

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
EnviarCTrabalho	im	-	1	1	-
EnviarReq	im	-	1	1	-

Modelo Infraestrutura de Processamento

O Modelo Infraestrutura de Processamento (ver Figura 5.3) representa a infraestrutura de processamento usada para instanciação de máquinas virtuais com diferentes configurações na nuvem privada.

A marcação *NP* atribuída ao lugar $InfraestruturaProcessamento$ representa o número total de *cores* fornecido pela infraestrutura de nuvem computacional para a instanciação de uma determinada máquina virtual. Como exemplo, quando *NP* recebe o valor 2 significa que a máquina virtual necessita de 2 *cores* para sua execução. A transição temporizada $TResposta$ representa os tempos necessários para o atendimento das requisições dos usuários ($T_{Resposta}$), que podem seguir o comportamento de qualquer distribuição de probabilidade. Os atributos das transições do Modelo Infraestrutura de Processamento (ver Figura 5.3) são apresentados na Tabela 5.3.

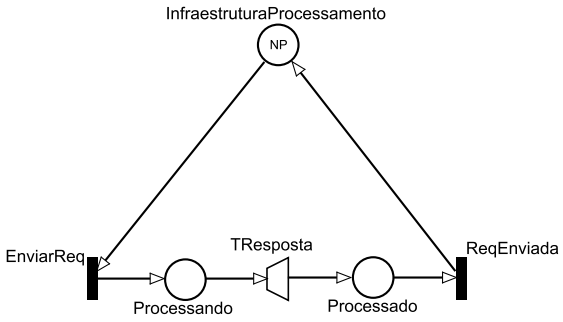


Figura 5.3: Modelo Infraestrutura de Processamento

Tabela 5.3: Atributos das Transições - Modelo Infraestrutura de Processamento

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
$TResposta_i$	exp	$T_{Resposta}$	-	-	SS
EnviarReq	im	-	1	1	-
ReqEnviada	im	-	1	1	-

O modelo de desempenho proposto (ver Figura 5.4) é composto pelos submodelos Cliente, Memória e Infraestrutura de Processamento. Esse modelo de desempenho representa vários

tipos e níveis de requisições aos serviços configurados nas máquinas virtuais das plataformas de nuvens.

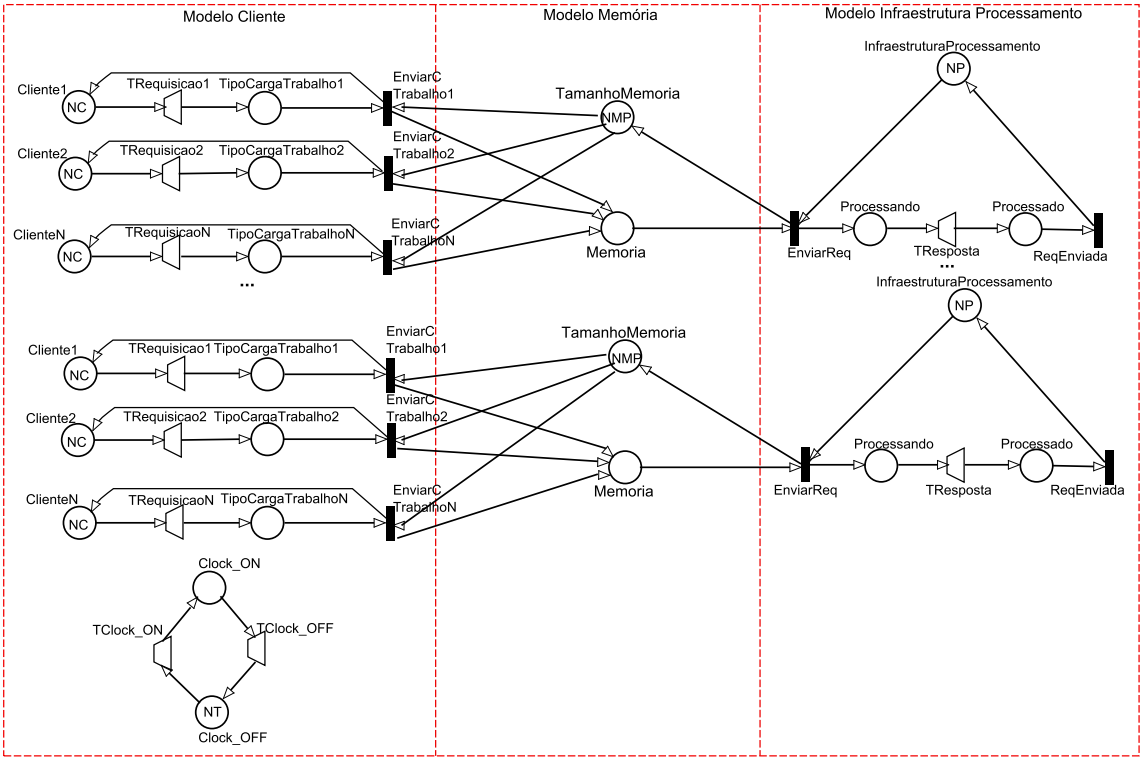


Figura 5.4: Modelo de Desempenho

Esse modelo de desempenho avalia o impacto de diferentes tipos e níveis de requisições de usuários na infraestrutura da nuvem privada através das métricas tempos de resposta e utilização de recursos. Essa avaliação proporciona meios para o planejamento de infraestruturas que atendam a determinados níveis de carga de trabalho com a qualidade de serviço requerida.

Métricas de Desempenho

O modelo de desempenho (ver Figura 5.4) proporciona o cálculo das métricas tempo de resposta e utilização de recursos das infraestruturas de nuvens privadas.

Tempo de resposta (TR) representa o tempo decorrido entre o pedido da realização de um serviço pelo usuário e a chegada da resposta do pedido ao usuário requisitante. Essa métrica é representada em unidade de tempo.

O tempo de resposta é calculado através da lei de *Little* (BAKOUCH, 2011; TRIVEDI, 2008) conforme a Equação 5.1.

$$T = \frac{N}{\lambda} \tag{5.1}$$

Onde T representa o tempo de permanência no sistema, N indica o número de usuários no sistema e λ representa a taxa de chegada de usuários ao sistema. Assim, $N = E\{\#TipoCarga\ Trabalho\} + E\{\#Memoria\} + E\{\#Processando\}$ é o número de requisições dos usuários

aguardando para serem atendidas pelo serviço hospedado na nuvem privada e $\lambda = P\{\#Cliente > 0\} \times W(TRequisicao)$ (ZIMMERMANN et al., 2006) é a taxa de chegada de requisições a nuvem computacional (ver Figura 5.4) (BAKOUCH, 2011; TAVARES et al., 2012).

Mais especificamente, E representa o número de marcações no lugar indicado entre chaves, P representa a probabilidade da ocorrência da expressão entre chaves. $P\{\#Cliente > 0\}$ indica a probabilidade de haver mais do que zero marcações no lugar $Cliente$ e $W(TRequisicao)$ (ZIMMERMANN et al., 2006) representa o tempo associado à transição $TRequisicao$ (ZIMMERMANN et al., 2006) (tempo entre envio de requisições dos usuários) (BAKOUCH, 2011; TAVARES et al., 2012).

Utilização de memória (UM) representa a razão entre a memória usada para o atendimento das requisições dos usuários e a memória total da máquina virtual. Essa métrica é calculada através da Equação 5.2 sendo representada como porcentagem (%) da utilização da memória total.

$$UM = 100 \times \frac{QMU}{QMT} \tag{5.2}$$

Onde QMU representa a memória usada para o atendimento das requisições dos usuários e QMT representa a memória total da máquina virtual. Assim, $QMU = E\{\#Memoria\} \times QR$ (ZIMMERMANN et al., 2006) (ZIMMERMANN et al., 2006) é a quantidade média de memória usada para o atendimento das requisições dos usuários.
$E\{\#Memoria\}$ representa o número médio de marcações no lugar $Memoria$ e QR representa a quantidade de memória de cada marcação.

Utilização de processador (UP) representa a fração de tempo que o processador permanece ocupado atendendo às requisições feitas pelos usuários. Essa métrica é representada como porcentagem (%) da utilização da infraestrutura de processamento total.

A utilização do processador UP é descrita pela expressão $P\{\#Infraestrutura\ Processamento = 0\}$ (ZIMMERMANN et al., 2006), onde $InfraestruturaProcessamento$ representa o total de recursos de processamento da máquina virtual. Essa expressão indica a probabilidade de não haver marcações no lugar $InfraestruturaProcessamento$ quando há apenas um processador na máquina virtual.

Tabela 5.4: Métricas de Desempenho

Métrica	Expressão	Significado
TR	$(E\{\#TipoCargaTrabalho\} + E\{\#Memoria\} + E\{\#Processando\})/\lambda$	Representa o tempo decorrido entre o pedido da realização de um serviço pelo usuário e a chegada da resposta do pedido ao usuário requisitante.
UM	$100 \times (E\{\#Memoria\} \times QM)/QMT$	Representa a razão entre a memória usada para atender as requisições feitas pelos usuários e a memória total.
UP	$P\{\#Infraestrutura\ Processamento = 0\}$	Representa a fração de tempo que a infraestrutura de processamento permanece ocupada atendendo às requisições feitas pelos usuários.

Modelo Refinado de Desempenho

O refinamento dos modelos de desempenho é realizado através do cálculo das médias e desvios-padrões do tempo de envio de requisições (TER) dos usuários e o tempo de resposta da nuvem computacional (TR). As médias e desvios-padrões dessas métricas sugerem o tipo de distribuição expolinomial que melhor representa os dados medidos. A seleção da distribuição expolinomial é proporcionada pelo cálculo do inverso do coeficiente de variação dos dados medidos através da técnica de aproximação por fases (DESROCHERS; AL-JAAR, 1995). O inverso do coeficiente de variação dos dados medidos provê a seleção da distribuição expolinomial e obtenção dos seus parâmetros, conforme mostrado a seguir.

- Se $1/CV = 1$, a distribuição exponencial representa o tempo de envio de requisições (TER) dos usuários e o tempo médio de resposta da nuvem computacional (TR) associados às transições $TRequisicao$ e $TResposta$. As Figuras 5.5 e 5.6 apresentam os modelos que representam a distribuição exponencial para o tempo de envio de requisições (TER) dos usuários e o tempo de resposta (TR).

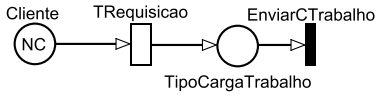


Figura 5.5: Modelo Cliente Exponencial

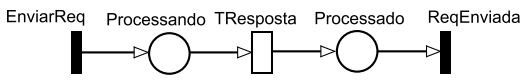


Figura 5.6: Modelo Infraestrutura de Processamento Exponencial

- Se $1/CV \in \mathbb{Z}$ *e* $1/CV \neq 1$, a distribuição *Erlang* representa o tempo de envio de requisições (TER) dos usuários e o tempo de resposta da nuvem computacional (TR) associados às transições $TRequisicao$ e $TResposta$. Os parâmetros dessa distribuição são o número de fases γ e a taxa λ. As Figuras 5.7 e 5.8 apresentam os modelos que representam a distribuição *Erlang* para o tempo de envio de requisições (TER) dos usuários e o tempo de resposta (TR).

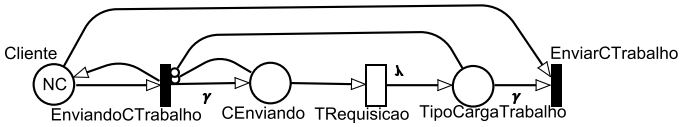


Figura 5.7: Modelo Cliente *Erlang*

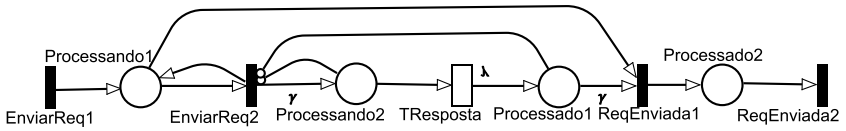


Figura 5.8: Modelo Infraestrutura de Processamento *Erlang*

- Se $1/CV > 1$ *e* $1/CV \notin \mathbb{Z}$, a distribuição hipoexponencial representa o tempo de envio de requisições (TER) dos usuários e tempo de resposta da nuvem computacional (TR) associados às transições $TRequisicao_1$, $TRequisicao_2$, $TResposta_1$ e $TResposta_2$. Os parâmetros dessa distribuição são o número de fases γ e as taxas λ_1 e λ_2. As Figuras 5.9 e 5.10 apresentam os modelos que representam a distribuição hipoexponencial para o tempo de envio de requisições (TER) dos usuários e o tempo de resposta (TR).

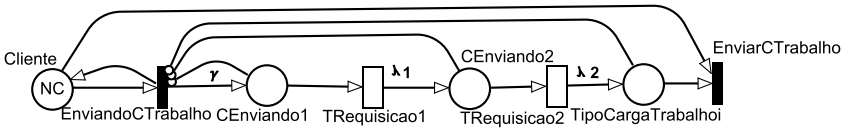


Figura 5.9: Modelo Cliente Hipoexponencial

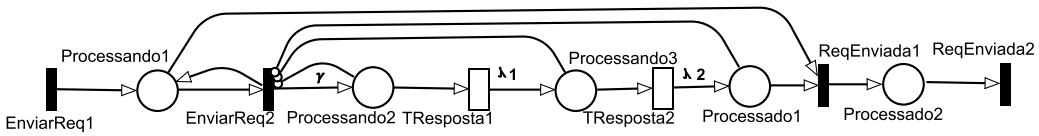


Figura 5.10: Modelo Infraestrutura de Processamento Hipoexponencial

- Se $1/CV < 1$, a distribuição hiperexponencial representa o tempo de envio de requisições (TER) dos usuários e tempo de resposta da nuvem computacional (TR) associados às transições $TRequisicao$ e $TResposta$. Os parâmetros dessa distribuição são os pesos ω_1, ω_2 e a taxa λ_h. As Figuras 5.11 e 5.12 apresentam os modelos que representam a distribuição hiperexponencial para o tempo de envio de requisições (TER) dos usuários e o tempo de resposta (TR).

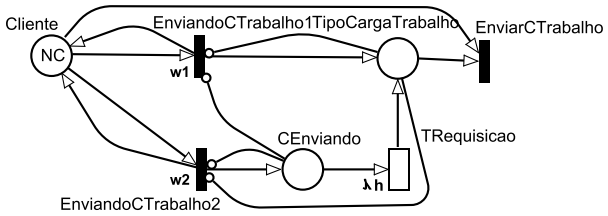


Figura 5.11: Modelo Cliente Hiperexponencial

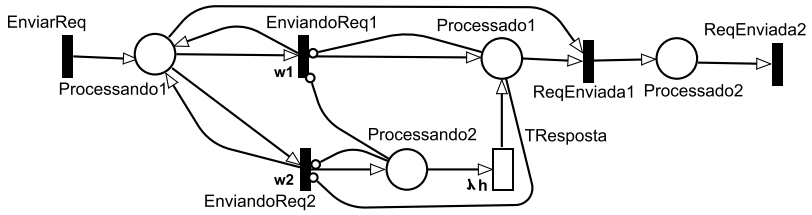


Figura 5.12: Modelo Infraestrutura de Processamento Hiperexponencial

Validação do Modelo de Desempenho

O cenário adotado para validação do modelo de desempenho é composto pelo *Moodle* (MOODLE, 2013) configurado na máquina virtual da plataforma *Eucalyptus* (EUCALYPTUS, 2013). Nesse cenário, duas máquinas físicas foram usadas para configuração da plataforma *Eucalyptus* e uma máquina física foi usada para configuração do gerador de carga *JMeter* (JMETER, 2013).

O controlador de nuvem (CLC) e o controlador de *cluster* da plataforma *Eucalyptus* foram configurados em uma máquina física e o controlador de nó (NC) também foi configurado em uma máquina física. O *Moodle* foi instalado em uma máquina virtual instanciada na máquina física onde o serviço do NC é executado. Essas máquinas físicas têm um processador de dois núcleos, uma memória de 2GB e uma memória secundária de 80GB.

O gerador de carga *JMeter* foi instalado em uma máquina física. Essa ferramenta proporciona a criação de requisições de usuários para as atividades fornecidas pelo *Moodle*. No cenário adotado, o *JMeter* gerou requisições de 6, 7, 8, 9 e 10 usuários para o *chat* do *Moodle*. A Tabela 5.5 apresenta os resultados da medição dos tempos de processamento para as requisições de 6, 7, 8, 9 e 10 usuários destinadas ao *chat* do *Moodle*.

O modelo de desempenho foi refinado com base nos resultados da medição dos tempos de processamento referentes às requisições de 6, 7, 8, 9 e 10 usuários para o *chat* do *Moodle*. A escolha de qual distribuição expolinomial é mais adequada aos tempos de processamento

Tabela 5.5: Resultados da Medição

Número de Usuários	Tempo de Processamento (s)
6	0,20
7	0,24
8	0,28
9	0,32
10	0,37

medidos foi realizada através do cálculo da média (μ_D) e desvio-padrão (σ_D) dos resultados da medição.

O cálculo do inverso do coeficiente de variação $1/CV = \mu_D/\sigma_D$ proporcionou a seleção da distribuição expolinomial e obtenção dos parâmetros numéricos dessa distribuição expolinomial. A Tabela 5.6 mostra a média (μ_D) e o desvio-padrão (σ_D) dos tempos de processamento medidos das requisições de 6, 7, 8, 9 e 10 usuários para o *chat* do *Moodle*.

Tabela 5.6: Média e Desvio-Padrão

Transição	μ_D (s)	σ_D (s)	Distribuição de Probabilidade
$TRequisicao$	0,14	0,008	Hipoexponencial
$TResposta$	0,28	0,066	Hipoexponencial

Após a definição da distribuição expolinomial mais adequada aos tempos de processamento medidos conforme a Tabela 5.6, deve-se calcular os parâmetros dessa distribuição. Os parâmetros μ_1, μ_2 e γ da distribuição hipoexponencial foram calculados e apresentados na Tabela 5.7.

Tabela 5.7: Parâmetros da Distribuição de Probabilidade

Transição	μ_1 (s)	μ_2 (s)	γ
$TRequisicao_i$	0,00014	0,14	3
$TResposta_i$	0,025	0,25	17

As utilizações de processador medidas e obtidas do modelo refinado de desempenho são mostradas na Figura 5.13. O teste t emparelhado foi aplicado às utilizações de processador medidas e obtidas do modelo de desempenho. Considerando um nível de significância de 5%, o teste t emparelhado gerou um intervalo de confiança de (-4,414, 0,508). Como o intervalo de

confiança contém 0, não há evidências estatísticas para rejeitar a hipótese de equivalência entre as utilizações do processador medidas e obtidas do modelo de desempenho (BAKOUCH, 2011).

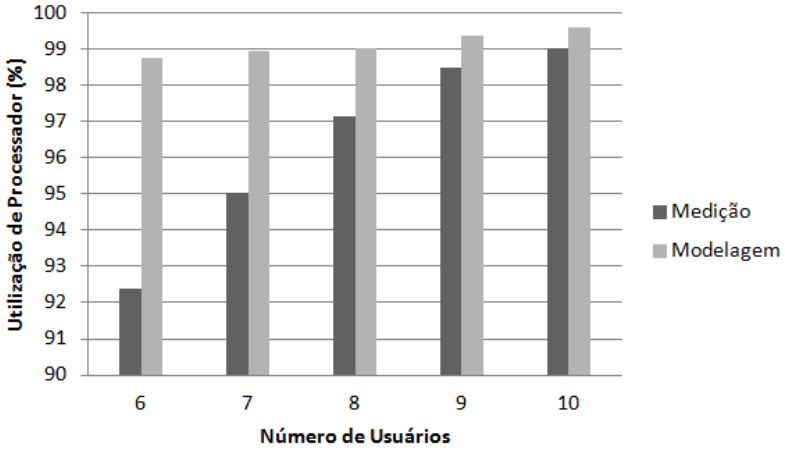


Figura 5.13: Análise Quantitativa do Modelo de Desempenho

5.1.2 Modelos de Disponibilidade

As métricas de dependabilidade podem ser calculadas através de modelos baseados em espaço de estados (ex: SPN) e de modelos combinatoriais (ex: RBD). Os RBDs apresentam uma vantagem em relação ao fornecimento dos resultados, pois apresentam cálculos mais rápidos por meio de suas fórmulas do que as simulações e as análises numéricas das SPNs. Entretanto, as SPNs têm um maior poder de representação (BALBO, 2001; MARSAN et al., 1998; KUO; ZUO, 2003).

Os modelos baseados em espaço de estados podem descrever dependências que possibilitam a representação de mecanismos de redundância complexos. Entretanto, esses modelos podem gerar um número muito grande ou mesmo infinito de estados quando representam sistemas de alta complexidade (BALBO, 2001; MARSAN et al., 1998; KUO; ZUO, 2003).

A combinação de modelos baseados em espaço de estados e de modelos combinatoriais propicia a redução da complexidade na representação dos sistemas.

A Figura 5.14 mostra um modelo SPN básico que permite uma representação de alto nível da nuvem privada. Nesse modelo SPN, os lugares ON e OFF representam a nuvem computacional em funcionamento ou em estado de falha. Os atributos das transições desse modelo SPN são apresentados na Tabela 5.8.

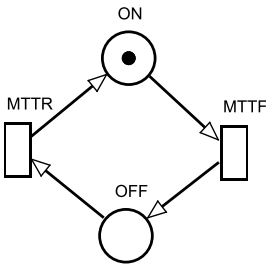


Figura 5.14: Modelo SPN

Tabela 5.8: Atributos das Transições do Modelo SPN

Transição	Tipo	Tempo	Peso	Concorrência
MTTF	exp	X_{MTTF}	-	SS
MTTR	exp	X_{MTTR}	-	SS

A Figura 5.15 mostra um modelo RBD básico que permite a representação de sistemas de alto nível e de subsistemas de baixo nível da infraestrutura da nuvem privada. Os parâmetros do modelo RBD são apresentados na Tabela 5.9.

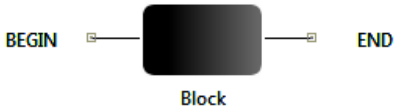


Figura 5.15: Modelo RBD

Tabela 5.9: Parâmetros do Modelo RBD

Parâmetros	Descrição
$MTTF_{Block}$	Tempo Médio para Defeito do *Block*
$MTTR_{Block}$	Tempo Médio de Reparo do *Block*

Esta seção apresenta os modelos SPN e RBD concebidos para avaliação da disponibilidade da infraestrutura da nuvem privada (ver Seção 4.3). Os modelos são: modelos das plataformas de nuvens, Modelo do Sistema Computacional, Modelo da Máquina Virtual, Modelo do Gerenciador de Recursos, Modelo de Redundância Ativo-Ativo, os modelos de redundância ativo-passivo e Modelo de Manutenção.

Os modelos RBD e SPN são combinados conforme a estratégia de modelagem apresentada no método de Geração de Modelo de Disponibilidade (ver Seção 4.3). Essa estratégia de modelagem hierárquica proporciona uma redução na complexidade da representação da nuvem privada. Modelos RBD representam os componentes da infraestrutura da nuvem. Como exemplo

temos os modelos do sistema computacional e da máquina virtual. Esses modelos são agrupados de forma a representar os subsistemas da infraestrutura da nuvem. Os resultados dos modelos de subsistemas são usados como parâmetros de dependabilidade para o modelo dos sistemas das infraestruturas de nuvens privadas. Os modelos SPN e RBD representam os sistemas da infraestrutura da nuvem. Como exemplo temos os modelos das plataformas de nuvens.

A Figura 5.16 apresenta um modelo RBD dos recursos de processamento e de armazenamento de um sistema computacional e apresenta um modelo SPN do sistema computacional com a atribuição de um mecanismo de redundância *cold standby*.

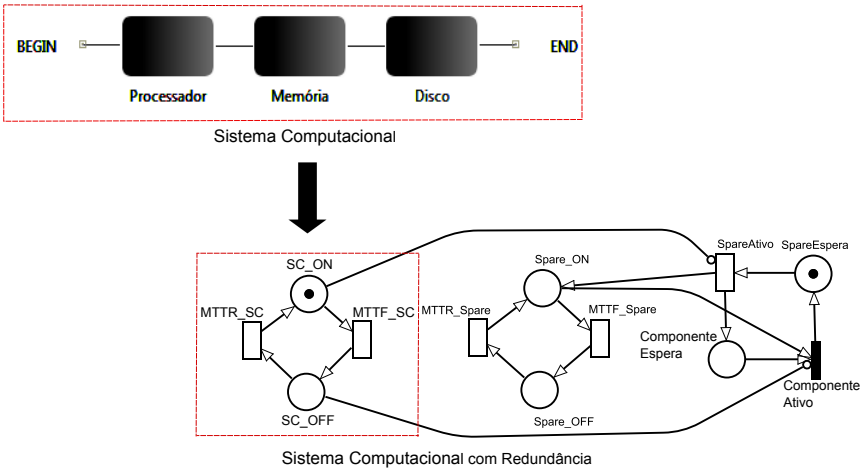


Figura 5.16: Sistema Computacional com um Mecanismo de Redundância *Cold Standby*

O modelo do sistema computacional poderia ser um RBD se não houvesse a necessidade da representação da dependência entre o sistema computacional e o mecanismos de redundância *cold standby*. Maiores detalhes sobre o Modelo *Cold Standby* serão apresentados nesta seção.

Modelos das Plataformas de Nuvem

Os modelos das plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack* representam os componentes dessas plataformas através de diagramas de blocos de confiabilidade. Esses diagramas de blocos de confiabilidade determinam a disponibilidade dessas infraestruturas de nuvem por meio da disponibilidade dos seus componentes.

Modelo da Plataforma *Eucalyptus*

A plataforma *Eucalyptus* (EUCALYPTUS, 2013) é composta pelo Controlador de Nuvem (CLC), Controlador de Cluster (CC), Controlador de Nó (NC) e Controlador de Armazenamento (SC). A Figura 5.17 apresenta uma arquitetura da plataforma *Eucalyptus*, na qual o CLC e o SC estão configurados na mesma máquina física, o CC está configurado em uma máquina física e o NC também está configurado em uma máquina física, possibilitando a instanciação de uma máquina virtual. As máquinas físicas onde os componentes da plataforma *Eucalyptus* estão

configurados são conectadas por meio de um *switch* e um roteador.

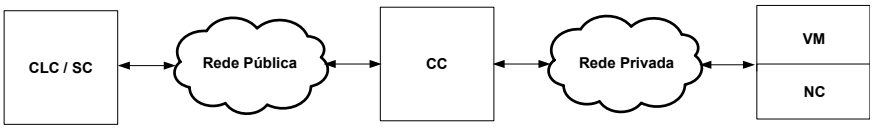


Figura 5.17: Arquitetura da Plataforma *Eucalyptus*

Todos os componentes da arquitetura da plataforma *Eucalyptus* devem estar operacionais para que a nuvem computacional esteja operacional. Esses componentes podem ser descritos como CLC, CC, NC, máquina virtual (VM), *switch* (SW) e roteador (RT). Desta forma, o modo operacional desse ambiente de nuvem é $OM_{PE} = (CLC \wedge CC \wedge NC \wedge VM \wedge SW \wedge RT)$. A Figura 5.18 mostra o modelo RBD adotado para estimar a disponibilidade e o *downtime* da infraestrutura da plataforma *Eucalyptus* por meio dos diagramas de bloco de confiabilidade de CLC, CC, NC, VM, SW e RT (SOUSA et al., 2014b), (SOUSA et al., 2014), (SOUSA et al., 2013). Os parâmetros do Modelo da Plataforma *Eucalyptus* são apresentados na Tabela 5.10.

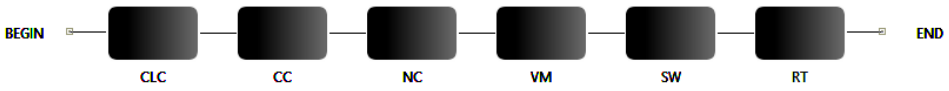


Figura 5.18: Modelo da Plataforma *Eucalyptus*

Tabela 5.10: Parâmetros do Modelo da Plataforma *Eucalyptus*

Parâmetros	Descrição
$MTTF_{CLC}$, $MTTF_{CC}$, $MTTF_{NC}$, $MTTF_{VM}$, $MTTF_{SW}$, $MTTF_{RT}$	Tempo Médio para Defeito de CLC, CC, NC, VM, SW e RT
$MTTR_{CLC}$, $MTTR_{CC}$, $MTTR_{NC}$, $MTTR_{VM}$, $MTTR_{SW}$, $MTTR_{RT}$	Tempo Médio de Reparo de CLC, CC, NC, VM, SW e RT

Modelo da Plataforma *Nimbus*

A plataforma *Nimbus* (PROJECT, 2013a) é composta pelo *Service Node* e *VMM Node* conforme mostrado na Figura 5.19. Esses componentes são configurados em máquinas físicas que são conectadas por meio de um *switch* e um roteador.

Figura 5.19: Arquitetura da Plataforma *Nimbus*

Os componentes da arquitetura da plataforma *Nimbus* devem estar operacionais para que esse ambiente de nuvem esteja operacional. O modo operacional desse ambiente de nuvem é

$OM_{PN} = (SN \wedge NO \wedge VM \wedge SW \wedge RT)$, onde SN, NO, VM, SW e RT são *Service Node*, *Node*, máquina virtual, *switch* e roteador, respectivamente. A Figura 5.20 mostra o modelo RBD adotado para estimar a disponibilidade e o *downtime* da infraestrutura dessa plataforma. Os parâmetros desse modelo são apresentados na Tabela 5.11.

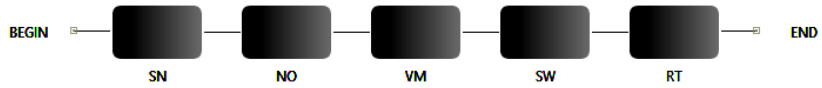


Figura 5.20: Modelo da Plataforma *Nimbus*

Tabela 5.11: Parâmetros do Modelo da Plataforma *Nimbus*

Parâmetros	Descrição
$MTTF_{SN}$, $MTTF_{NO}$, $MTTF_{VM}$, $MTTF_{SW}$, $MTTF_{RT}$	Tempo Médio para Defeito de SN, NO, VM, SW e RT
$MTTR_{SN}$, $MTTR_{NO}$, $MTTR_{VM}$, $MTTR_{SW}$, $MTTR_{RT}$	Tempo Médio de Reparo de SN, NO, VM, SW e RT

Modelo da Plataforma *OpenNebula*

A plataforma *OpenNebula* (PROJECT, 2013b) é composta pelo *Front-end* e *Cluster Node* (ver Figura 5.21). Cada componente dessa plataforma está configurado em uma máquina física e essas máquinas estão conectadas por meio de um *switch* e um roteador.

Figura 5.21: Arquitetura da Plataforma *OpenNebula*

O modo operacional para a arquitetura da plataforma *OpenNebula* é $OM_{PON} = (FE \wedge CN \wedge VM \wedge SW \wedge RT)$, significando que o *Front-End*, o *Cluster Node*, a máquina virtual, o *switch* e o roteador devem estar operacionais para que a nuvem computacional também esteja. A Figura 5.22 representa a plataforma *OpenNebula* por meio dos diagramas de bloco de confiabilidade de FE, CN, VM, SW e RT. Os parâmetros do Modelo da Plataforma *OpenNebula* são apresentados na Tabela 5.12.

Figura 5.22: Modelo da Plataforma *OpenNebula*

Tabela 5.12: Parâmetros do Modelo da Plataforma *OpenNebula*

Parâmetros	Descrição
$MTTF_{FE}$, $MTTF_{CN}$, $MTTF_{VM}$, $MTTF_{SW}$, $MTTF_{RT}$	Tempo Médio para Defeito de FE, CN, VM, SW e RT
$MTTR_{FE}$, $MTTR_{CN}$, $MTTR_{VM}$, $MTTR_{SW}$, $MTTR_{RT}$	Tempo Médio de Reparo de FE, CN, VM, SW e RT

Modelo da Plataforma *OpenStack*

A plataforma *OpenStack* (PROJECT, 2013c) é constituída por três componentes: *OpenStack Object Storage*, *OpenStack Imaging Service* e *OpenStack Compute*. O *Controller Node* executa os serviços do *OpenStack Compute* e do *OpenStack Imaging Service* configurado em uma máquina física e o *Compute Nodes* executa o serviço do *Openstack Object Storage* configurado em uma máquina física (ver Figura 5.23).

Figura 5.23: Arquitetura da Plataforma *OpenStack*

O modo operacional para a arquitetura da plataforma *OpenStack* é $OM_{POS} = (CLN \wedge CPN \wedge VM \wedge SW \wedge RT)$, onde CLN, CPN, VM, SW e RT são o *Controller Node*, o *Compute Node*, a máquina virtual, o *switch* e o roteador. Esses componentes devem estar operacionais para que a nuvem computacional também esteja. A Figura 5.24 representa o modelo da plataforma *OpenStack*. Os parâmetros desse modelo são apresentados na Tabela 5.13.

Figura 5.24: Modelo da Plataforma *OpenStack*

Tabela 5.13: Parâmetros do Modelo da Plataforma *OpenStack*

Parâmetros	Descrição
$MTTF_{CLN}$, $MTTF_{CPN}$, $MTTF_{VM}$, $MTTF_{SW}$, $MTTF_{RT}$	Tempo Médio para Defeito de CLN, CPN, VM, SW e RT
$MTTR_{CLN}$, $MTTR_{CPN}$, $MTTR_{VM}$, $MTTR_{SW}$, $MTTR_{RT}$	Tempo Médio de Reparo de CLN, CPN, VM, SW e RT

Modelo do Sistema Computacional

O Modelo RBD do Sistema Computacional (SOUSA et al., 2014b), (SOUSA et al., 2014), (SOUSA et al., 2013) representa a infraestrutura de processamento (PR), a memória (MP) e a memória secundária (MS) da máquina física (ver Figura 5.25). Esse modelo RBD é adotado para estimar o MTTF e o MTTR do sistema computacional. O modo operacional desse modelo é $OM_{CS} = (PR \wedge MP \wedge MS)$. Os parâmetros do Modelo do Sistema Computacional são apresentados na Tabela 5.14.

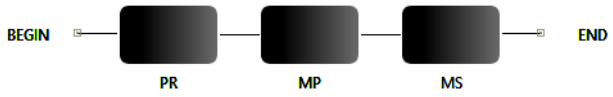


Figura 5.25: Modelo do Sistema Computacional

Tabela 5.14: Parâmetros do Modelo do Sistema Computacional

Parâmetros	Descrição
$MTTF_{PR}$, $MTTF_{MP}$, $MTTF_{MS}$	Tempo Médio para Defeito de PR, MP e MS
$MTTR_{PR}$, $MTTR_{MP}$, $MTTR_{MS}$	Tempo Médio de Reparo de PR, MP e MS

Modelo da Máquina Virtual

O modelo RBD da máquina virtual representa os *softwares* necessários para a configuração do sistema que será hospedado na nuvem (ver Figura 5.26) (SOUSA et al., 2014b), (SOUSA et al., 2014), (SOUSA et al., 2013). Como exemplo temos o sistema cliente (SC), o sistema operacional (SO), o banco de dados (BD), o servidor *web* (SWE), o monitor da máquina virtual (MMV) e o balanceador de carga (BC). Esse modelo RBD é adotado para estimar o MTTF e o MTTR da máquina virtual. O modo operacional desse modelo é $OM_{VM} = (SC \wedge SO \wedge BD \wedge SWE \wedge MMV \wedge BC)$. Os parâmetros do Modelo da Máquina Virtual são apresentados na Tabela 5.15.

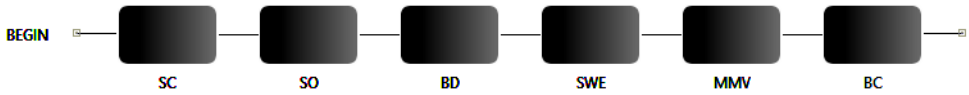


Figura 5.26: Modelo da Máquina Virtual

Tabela 5.15: Parâmetros do Modelo da Máquina Virtual

Parâmetros	Descrição
$MTTF_{SC}$, $MTTF_{SO}$, $MTTF_{BD}$, $MTTF_{SWE}$, $MTTF_{MMV}$, $MTTF_{BC}$	Tempo Médio para Defeito de SC, SO, BD, SWE, MMV e BC
$MTTR_{SC}$, $MTTR_{SO}$, $MTTR_{BD}$, $MTTR_{SWE}$, $MTTR_{MMV}$, $MTTR_{BC}$	Tempo Médio de Reparo de SC, SO, BD, SWE, MMV e BC

Modelo do Gerenciador de Recursos

O modelo RBD do gerenciador de recursos é adotado para estimar o MTTF e o MTTR dos gerenciadores de recursos das plataformas de nuvens através de diagramas de bloco de confiabilidade do sistema computacional (SCL), da plataforma de nuvem (PN) e do sistema operacional (SO) (ver Figura 5.27). O modo operacional desse modelo é $OM_{RG} = (SCL \wedge PN \wedge SO)$. Os parâmetros do Modelo do Gerenciador de Recursos são apresentados na Tabela 5.16.

Figura 5.27: Modelo do Gerenciador de Recursos

Tabela 5.16: Parâmetros do Modelo do Gerenciador de Recursos

Parâmetros	Descrição
$MTTF_{SCL}$, $MTTF_{PN}$, $MTTF_{SO}$	Tempo Médio para Defeito de SCL, PN e SO
$MTTR_{SCL}$, $MTTR_{PN}$, $MTTR_{SO}$	Tempo Médio de Reparo de SCL, PN e SO

Modelo de Redundância Ativo-Ativo

Os mecanismos de redundância do tipo ativo-ativo são empregados quando os componentes primários e secundários atendem às requisições dos usuários do sistema. O Modelo de Redundância Ativo-Ativo proposto representa a plataforma *Eucalyptus* com esse mecanismo de redundância através de redes de *Petri* estocásticas.

O modelo SPN proposto foi baseado em uma cadeia de *Markov* apresentada em (BAUER; ADAMS; EUSTACE, 2011). O modo operacional do Modelo de Redundância Ativo-Ativo pro-

posto é $OM_{N+1} = (CLC\_ON = 0 \wedge CC\_ON = 0 \wedge NC\_ON = 0 \wedge SW\_ON = 0 \wedge RT\_ON = 0 \wedge (VM1\_ON = 0 \vee VM2\_ON = 0))$. Este modo operacional mostra que a nuvem computacional estará disponível quando o CLC, o CC, o NC, o *switch*, o roteador e uma das máquinas virtuais estiverem operacionais. A Figura 5.28 mostra o modelo SPN adotado para estimar a disponibilidade de um sistema com redundância ativo-ativo. As marcações dos lugares CLC_ON, CC_ON, NC_ON, SW_ON, RT_ON, $VM1\_ON$ e $VM2\_ON$ denotam os estados operacionais do CLC, CC, NC, *switch*, roteador, máquina virtual principal e máquina virtual redundante, respectivamente, e as marcações dos lugares CLC_OFF, CC_OFF, NC_OFF, SW_OFF, RT_OFF, $VM1\_OFF$ e $VM2\_OFF$ denotam os estados defeituosos desses componentes.

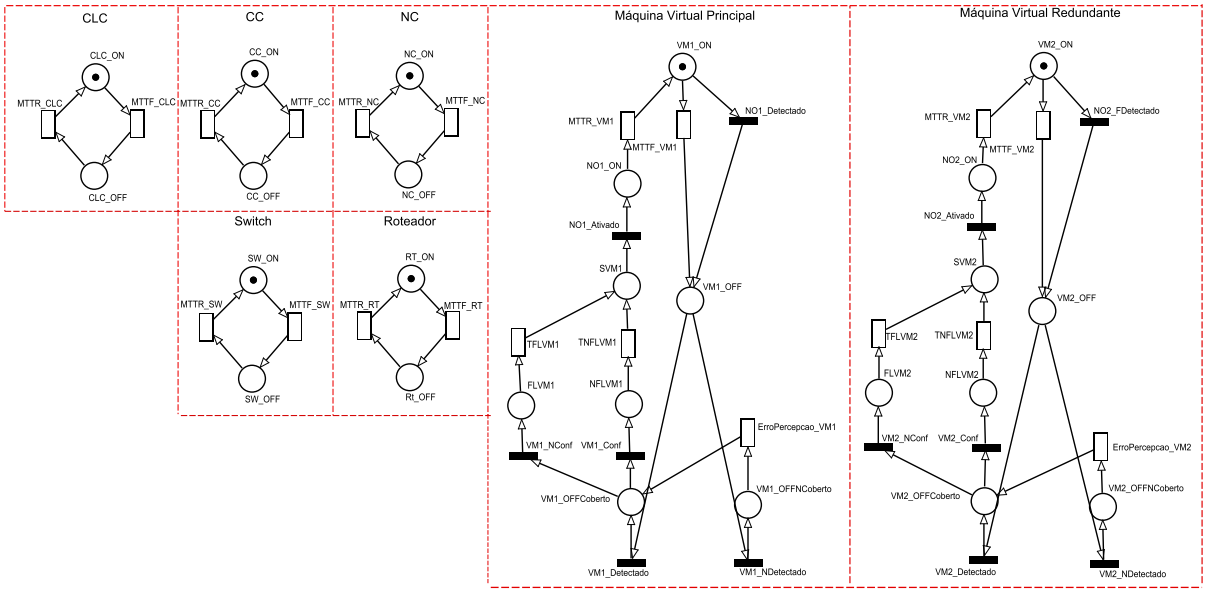


Figura 5.28: Modelo de Redundância Ativo-Ativo

As máquinas virtuais principal e redundante compartilham a carga de trabalho do sistema. Essas máquinas virtuais estão instanciadas na máquina física onde o serviço do NC é executado. Cada uma das máquinas virtuais falha quando há um evento de falha no NC ou na própria máquina virtual. A função de habilitação $(\#NC\_ON = 0)$ atribuída às transições imediatas $NO1\_Detectado$ e $NO2\_Detectado$ representa a falha de cada uma das máquinas virtuais devido a ocorrência de eventos de falhas no NC. As máquinas virtuais serão reparadas quando o NC estiver operacional. Essa condição é representada pela função de habilitação $(\#NC\_ON = 1)$ atribuída às transições imediatas $NO1\_Ativado$ e $NO2\_Ativado$. Os lugares $NO1\_ON$ e $NO2\_ON$ representam que a nuvem está operacional, pois o NC está funcionando.

Quando a máquina virtual principal falha, a máquina virtual redundante assume a carga de trabalho do sistema. As transições temporizadas $MTTF\_VM1$, $MTTF\_VM2$, $MTTR\_VM1$ e $MTTR\_VM2$ representam a ocorrência de um evento de falha e de uma atividade de reparo nas máquinas virtuais principal e redundante, e os tempos associados a essas transições temporizadas representam o MTTF e o MTTR desses componentes.

Esse modelo também representa a probabilidade da detecção e da não detecção do evento

de falha através de pesos associados as transições imediatas $VM1\_Detectado$, $VM2\_Detectado$, $VM1\_NDetectado$ e $VM2\_NDetectado$. Após a ocorrência do evento de falha com os disparos das transições temporizadas $MTTF\_VM1$ e $MTTF\_VM2$, esse evento de falha pode ser detectado com os disparos das transições imediatas $VM1\_Detectado$ e $VM2\_Detectado$ e pode não ser detectado com os disparos das transições imediatas $VM1\_NDetectado$ e $VM2\_NDetectado$. Os lugares $VM1\_OFFCoberto$ e $VM2\_ OFFCoberto$ denotam a detecção do evento de falha e os lugares $VM1\_OFFNCoberto$ e $VM2\_OFFNCoberto$ representam a não detecção do evento de falha.

As máquinas virtuais principal e redundante compartilham a carga de trabalho do sistema. Quando há a detecção de um evento de falha na máquina virtual principal, o sistema pode ser configurado para enviar a carga de trabalho somente para a máquina virtual redundante. A configuração do sistema para o envio de toda a carga de trabalho para a máquina virtual redundante ocorrerá após os disparos das transições $VM1\_Conf$ e $TNFLVM1$ ou das transições $VM2\_Conf$ e $TNFLVM2$. Os lugares $NFLVM1$ e $NFLVM2$ representam o início da configuração do sistema e os lugares $SVM1$ e $SVM2$ representam a conclusão da configuração do sistema.

Quando houver uma falha na configuração do sistema, a carga de trabalho será enviada para as máquinas virtuais principal e redundante. Entretanto, as requisições enviadas para a máquina virtual principal não serão atendidas. Os lugares $FLVM1$ e $FLVM2$ representam a não realização da configuração do sistema. Enquanto o sistema não puder ser configurado para o envio de toda a carga de trabalho para a máquina virtual redundante, as requisições não serão atendidas. Neste caso, a configuração do sistema ocorrerá após o disparo das transições $VM1\_NConf$ e $TFLVM1$ ou $VM2\_NConf$ e $TFLVM2$. A probabilidade de haver ou não a configuração do sistema é modelada através de pesos associados às transições imediatas $VM1\_Conf$, $VM2\_Conf$ e $VM1\_NConf$, $VM2\_NConf$, respectivamente.

A detecção do defeito e a configuração do sistema para o envio da carga de trabalho para a máquina virtual redundante permite o reparo do sistema após o disparo das transições temporizadas $MTTR\_VM1$ e $MTTR\_VM2$.

Quando o defeito não é detectado, há um erro de percepção, pois para o sistema não ocorreu nenhum defeito. As transições temporizadas $ErroPercepcao\_VM1$ e $Erro\ Percepcao\_VM2$ representam esse erro de percepção e os tempos associados a essas transições representam o período de duração desse erro de percepção. Após esse período de tempo, o defeito é percebido. Os atributos das transições do Modelo de Redundância Ativo-Ativo (ver Figura 5.28) são apresentados na Tabela 5.17.

Tabela 5.17: Parâmetros do Modelo de Redundância Ativo-Ativo

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
MTTF_CLC, MTTF_CC, MTTF_NC, MTTF_SW, MTTF_RT, MTTF_VM1, MTTF_VM2	exp	MTTF	-	-	SS
MTTR_CLC, MTTR_CC, MTTR_NC, MTTR_SW, MTTR_RT, MTTR_VM1, MTTR_VM2	exp	MTTR	-	-	SS
ErroPercepcao_VM1, ErroPercepcao_VM2	exp	MTTEP	-	-	SS
VM1_Detectado, VM2_Detectado	im	-	0,8	1	-
VM1_NDetectado, VM2_NDetectado	im	-	0,2	1	-
NO1_Detectado, NO2_Detectado, NO1_Ativado, NO2_Ativado	im	-	1	1	-
NO1_Ativado, NO2_Ativado	im	-	1	1	-
VM1_Conf, VM2_Conf	im	-	0,8	1	-
VM1_Nconf, VM2_NConf	im	-	0,2	1	-
TNFLVM1, TNFLVM2	exp	TNFL	-	-	SS
TFLVM1, TFLVM2	exp	TFL	-	-	SS

O Modelo de Desempenho pode ser adotado para estimar o desempenho da plataforma *Eucalyptus* com o mecanismo de redundância ativo-ativo. A Figura 5.29 apresenta o Modelo de Desempenho adaptado para representar um serviço configurado em duas máquinas virtuais, uma principal e uma redundante.

No modelo de desempenho, os clientes enviando requisições para um serviço hospedado na nuvem computacional são representados pelo Modelo Cliente, as memórias das máquinas virtuais principal e redundante são representadas pelo Modelo Memória e as infraestruturas de processamento das máquinas virtuais principal e redundante são representadas pelo Modelo Infraestrutura de Processamento.

As requisições dos usuários são atendidas pelo serviço configurado nas máquinas virtuais, quando as funções de habilitação $(\#Clock\_ON = 1)\,AND\,(\#VM1\_ON = 1)$ e $(\#Clock\_ON = 1)\,AND\,(\#VM2\_ON = 1)$ atribuídas às transições imediatas
$EnviarCTrabalho1$ e $EnviarCTrabalho2$ são satisfeitas.

As duas máquinas virtuais dividem a carga de trabalho do sistema, mas quando há um defeito em uma delas, a outra máquina virtual assume essa carga de trabalho. Se o defeito em

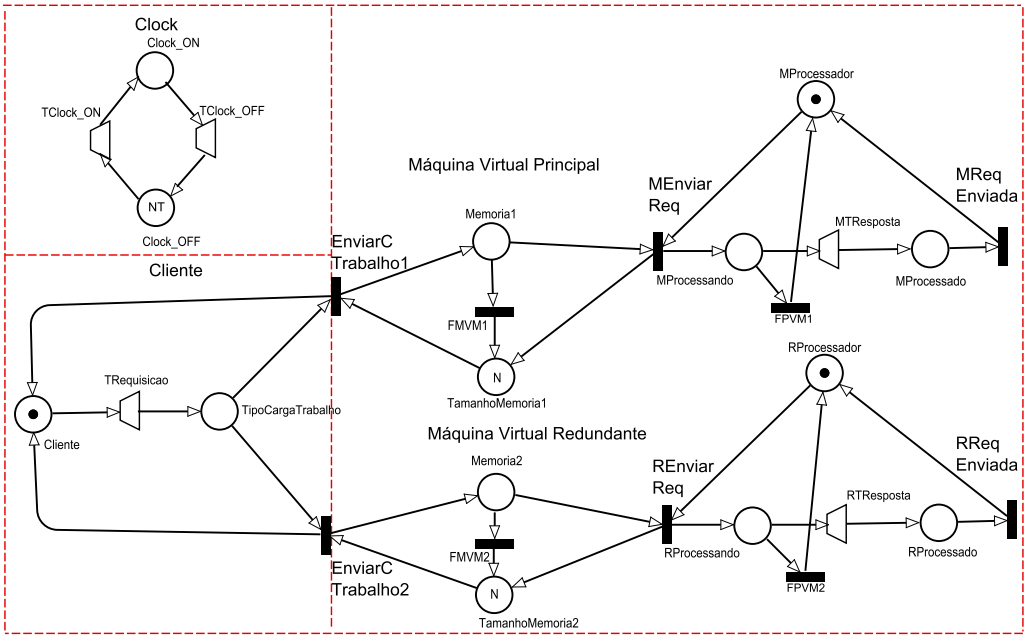


Figura 5.29: Modelo de Desempenho

uma máquina virtual não é detectado, as requisições que são enviadas a essa máquina virtual não são atendidas. A função de habilitação ($\#VM1\_ON = 0$) atribuída às transições imediatas $FPVM1$ e $FMVM1$ e a função de habilitação ($\#VM2\_ON = 0$) atribuída às transições imediatas $FPVM2$ e $FMVM2$ representam o descarte das requisições dos usuários. Os parâmetros do Modelo de Desempenho são apresentados na Tabela 5.18.

Tabela 5.18: Parâmetros do Modelo de Desempenho

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
$TRequisicao$	exp	$T_{Requisicao}$	-	-	SS
$EnviarCTrabalho1$, $EnviarCTrabalho2$, $MEnviarReq$, $REnviarReq$, $MReqEnviada$, $RReqEnviada$	im	-	1	1	-
$TClock\_ON$	exp	TON_{Clock}	-	-	SS
$TClock\_OFF$	exp	$TOFF_{Clock}$	-	-	SS
$MTResposta$	exp	$MT_{Resposta}$	-	-	SS
$RTResposta$	exp	$RT_{Resposta}$	-	-	SS
FVM1, FVM2	im	-	1	1	-

Validação do Modelo de Redundância Ativo-Ativo

O Modelo de Redundância Ativo-Ativo proposto (ver Figura 5.28) foi validado por meio da comparação dos resultados da disponibilidade deste modelo e do Modelo CTMC apresentado em (BAUER; ADAMS; EUSTACE, 2011).

A Figura 5.30 apresenta o modelo CTMC dos módulos CLC, CC e NC da plataforma *Eucalyptus*. O estado CLCCCNC representa que todos os módulos estão operacionais, os estados CLCCC, CLCNC e CCNC representam que apenas os módulos CLC e CC; CLC e NC; CC e NC estão operacionais, respectivamente, e os estados CLC, CC e NC representam que os módulos CLC, CC e NC estão operacionais, respectivamente. O MTTF e o MTTR dos módulos da plataforma *Eucalyptus* são 111111,11 horas e 8 horas, respectivamente.

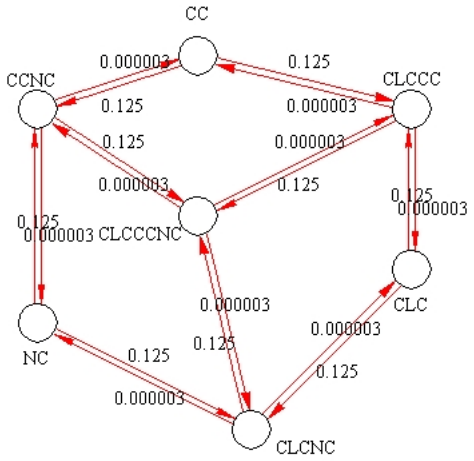


Figura 5.30: Modelo CTMC dos Módulos da Plataforma *Eucalyptus*

O modelo CTMC dos equipamentos de rede (ver Figura 5.31) representa o *switch* e o roteador da nuvem computacional. O estado SWRTUP representa que os equipamentos de rede estão operacionais e os estados SWDOWN e RTDOWN representam que o *switch* e o roteador não estão operacionais, respectivamente. O MTTF e o MTTR dos equipamentos de rede da nuvem computacional são 10752,69 horas e 8 horas, respectivamente.

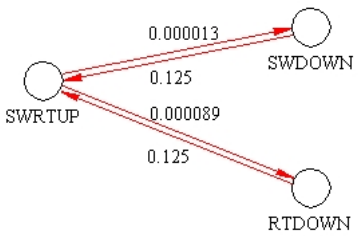


Figura 5.31: Modelo CTMC dos Equipamentos de Rede

A Figura 5.32 apresenta o modelo CTMC das máquinas virtuais com redundância ativo-ativo. Os estados *simplex* e *duplex* representam uma e duas máquinas virtuais operacionais, respectivamente. Os estados *detected failure* e *undetected failure* representam uma falha detectada e não detectada nas máquinas virtuais. O estado *simplex* representa que sistema envia

requisições apenas para a máquina virtual operacional. O estado *failed failover* representa que o sistema não conseguiu configurar o envio das requisições apenas para a máquina virtual operacional. O estado *duplex failure* retrata as duas máquinas virtuais em estado de falha. O MTTF e o MTTR das máquinas virtuais da Plataforma *Eucalyptus* são 293,90 horas e 8 horas, respectivamente.

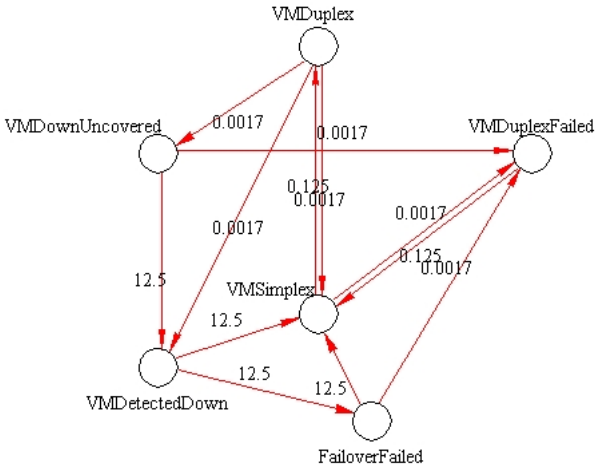


Figura 5.32: Modelo CTMC da Máquina Virtual com Redundância Ativo-Ativo

O modelo CTMC da plataforma *Eucalyptus* (ver Figura 5.33) representa os módulos, equipamentos de rede e máquinas virtuais dessa plataforma de nuvem. O estado MODREDEVM representa que todos componentes da nuvem estão operacionais, os estados MODREDE, MODVM e REDEVM representam que apenas as máquinas virtuais, equipamentos de rede e módulos da nuvem não estão operacionais, respectivamente, e os estados MOD, REDE e VM representam que os módulos, equipamentos de rede e máquinas virtuais estão operacionais, respectivamente.

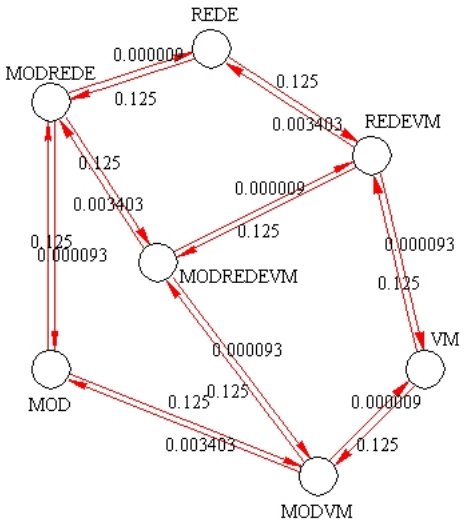


Figura 5.33: Modelo CTMC da Plataforma *Eucalyptus*

A Tabela 5.19 apresenta os parâmetros de dependabilidade usados para validação do

Modelo de Redundância Ativo-Ativo.

Tabela 5.19: Parâmetros de Dependabilidade dos Componentes da Plataforma *Eucalyptus*

Tipo	MTTF (horas)	MTTR (horas)	Tempo (horas)
CC,CLC,NC	329.067,64	8,00	-
RT	80.000,00	8,00	-
SW	11.200,00	8,00	-
VM1,VM2	576,00	8,00	-
ErroPercepcao_VM1,ErroPercepcao_VM2	-	-	0,08
TFLVM1, TFLVM2	-	-	0,16
TNFLVM1, TNFLVM2	-	-	0,08

Os resultados da disponibilidade do Modelo de Redundância Ativo-Ativo proposto e do modelo de redundância ativo-ativo apresentado em (BAUER; ADAMS; EUSTACE, 2011) foram comparados, para tanto foram adotados os mesmos parâmetros de dependabilidade (ver Tabela 5.19). O resultado da disponibilidade dos dois modelos foi 99,89%. Esse resultado mostra que o Modelo de Redundância Ativo-Ativo proposto está validado.

Modelos de Redundância Ativo-Passivo

Os mecanismos de redundância do tipo ativo-passivo são empregados quando os componentes primários atendem às requisições dos usuários do sistema e os componentes secundários estão em espera. O Modelo *Hot Standby* representa o mecanismo de redundância *hot standby* através de diagramas de blocos de confiabilidade. Os Modelos *Cold Standby* e *Warm Standby* representam os mecanismos de redundância *cold standby* e *warm standby* através de redes de *Petri* estocásticas.

Modelo *Hot Standby*

Um componente com redundância *hot standby* é baseado em um módulo redundante ativo. Nessa técnica de redundância, o módulo principal defeituoso pode ser substituído pelo módulo redundante sem atraso. Os módulos principal e redundante estão operacionais, eles recebem e avaliam as informações de entrada, mas os resultados do módulo redundante não são considerados como informações de saída do equipamento. No entanto, quando o módulo principal falhar, o módulo redundante estará ativo. Assim, a principal característica de um componente com uma redundância *hot standby* é a ausência de um tempo de ativação (se comparado com as redundâncias *cold standby* e *warm standby*) (KUO; ZUO, 2003; RUPE, 2003).

O Modelo *hot standby* (SOUSA et al., 2014b), (SOUSA et al., 2014), (SOUSA et al., 2013) mostra um único componente operacional (CP) e um mecanismo de redundância *hot standby* (CR). Desta forma, o modo operacional do componente com redundância *hot standby* é $OM_{HS} = (CP \vee CR)$. A Figura 5.34 mostra o modelo RBD adotado para estimar a disponibilidade de um componente (CP) com redundância *hot standby* (CR). Os parâmetros do Modelo *Hot Standby* são apresentados na Tabela 5.20.

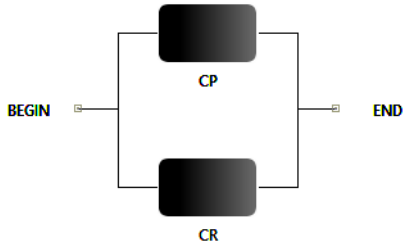


Figura 5.34: Modelo *Hot Standby*

Tabela 5.20: Parâmetros do Modelo *Hot Standby*

Parâmetros	Descrição
$MTTF_{CP}$, $MTTF_{CR}$	Tempo Médio para Defeito de CP e CR
$MTTR_{CP}$, $MTTR_{CR}$	Tempo Médio de Reparo de CP e CR

Modelo *Cold Standby*

Um componente com redundância *cold standby* é baseado em um módulo redundante não-ativo que espera para ser ativado quando o módulo principal ativo falha. Assim, quando o módulo principal falha, a ativação do módulo redundante ocorre em um certo período de tempo. Esse período é chamado de *Mean Time to Active* (MTA) (KUO; ZUO, 2003; RUPE, 2003). O modo operacional do componente com redundância *cold standby* é $OM_{CS} = (Componente\_ON = 0 \vee Spare\_ON = 0)$.

A Figura 5.35 mostra o modelo SPN adotado para estimar a disponibilidade de um componente com redundância *cold standby* (SOUSA et al., 2014b), (SOUSA et al., 2014), (SOUSA et al., 2013). As marcações dos lugares $Componente\_ON$, $Spare\_ON$, $Componente\ \_OFF$ e $Spare\_OFF$ denotam os estados operacionais e falhos de ambos os módulos principal e redundante, respectivamente. O módulo redundante está inicialmente desativado, uma vez que não há *tokens* nos lugares $Spare\_ON$ e $Spare\_OFF$. Quando o módulo principal falha, a transição temporizada $SpareAtivo$ dispara. O tempo associado à transição temporizada $SpareAtivo$ representa o *Mean Time to Active* (MTA) e uma marcação no lugar $SpareEspera$ denota que o módulo reserva não está operacional. As transições temporizadas $MTTF\_Componente$, $MTTF\_Spare$, $MTTR\_Componente$ e $MTTR\_Spare$ representam a ocorrência de um evento de falha e de uma

atividade de reparo nos módulos principal e redundante, e os tempos associados a essas transições temporizadas representam o MTTF e o MTTR desses componentes. O MTTF ($MTTFC$) e o MTTR ($MTTRC$) associados ao módulo principal podem ser diferentes do MTTF ($MTTFS$) e do MTTR ($MTTRS$) associados ao módulo redundante. Os atributos das transições do modelo *Cold Standby* (ver Figura 5.35) são apresentados na Tabela 5.21.

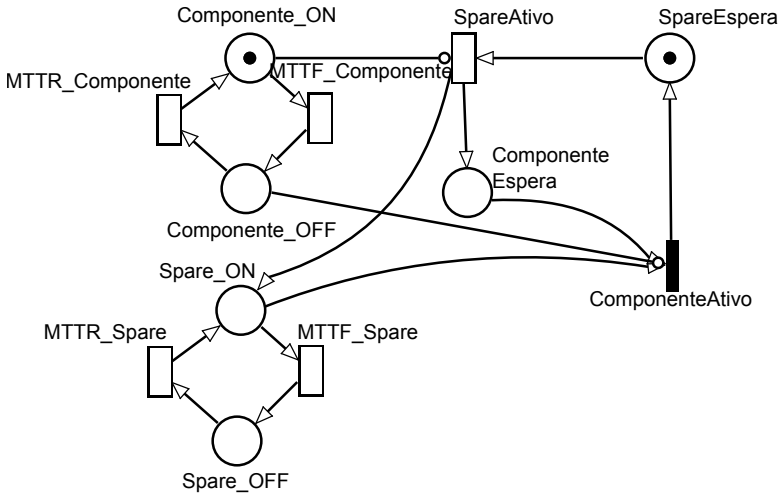


Figura 5.35: Modelo *Cold Standby*

Tabela 5.21: Atributos das Transições - Modelo *Cold Standby*

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
MTTF_Componente	exp	MTTFC	-	-	SS
MTTR_Componente	exp	MTTRC	-	-	SS
MTTF_Spare	exp	MTTFS	-	-	SS
MTTR_Spare	exp	MTTRS	-	-	SS
SpareAtivo	exp	MTA	-	-	SS
ComponenteAtivo	im	-	1	1	-

Modelo *Warm Standby*

Um componente com redundância *warm standby* é baseado em um módulo redundante não-ativo que espera para ser ativado quando o módulo principal ativo falha. A diferença em relação à redundância *cold standby* é que o módulo principal e o módulo redundante possuem uma taxa de falhas λ quando estão em operação, mas o módulo redundante possui uma taxa de falha ϕ quando está desenergizado, considerando que $0 \leq \phi \leq \lambda$ (KUO; ZUO, 2003; RUPE, 2003). O modo operacional do componente com redundância *warm standby* é

$OM_{WS} = (Componente\_ON = 0 \vee OPSpare\_ON = 0)$. A Figura 5.36 apresenta o Modelo *Warm Standby*.

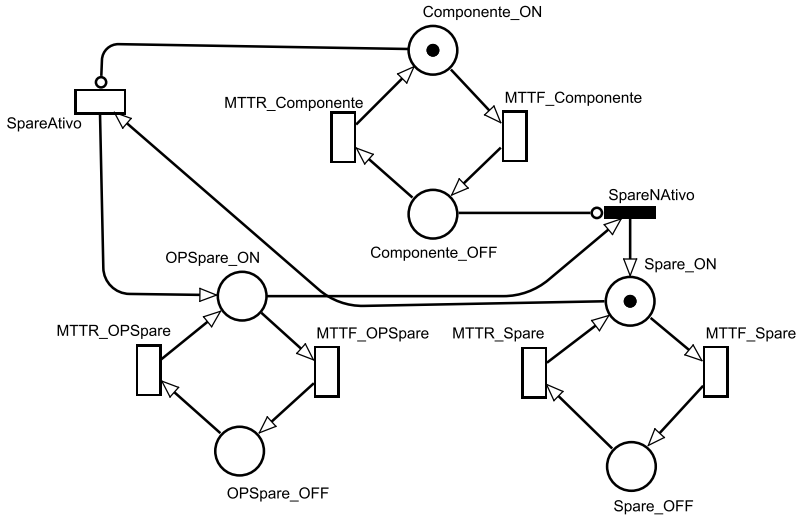


Figura 5.36: Modelo *Warm Standby*

O Modelo *warm standby* (GUIMARÃES; MACIEL; MATIAS JR., 2013) (ver a Figura 5.36) possui seis lugares, os lugares do módulo principal $Componente\_ON$ e $Componente\_OFF$ e os lugares do módulo redundante. Esses lugares do módulo redundante representam o seu estado de atividade e de inatividade. Os lugares $Spare\_ON$ e $Spare\_OFF$ representam o módulo reserva em modo inativo. Já os lugares $OPSpare\_ON$ e $OPSpare\_OFF$ representam o módulo reserva em modo ativo. O módulo redundante começa em modo inativo. Quando o módulo principal falha, a transição temporizada $SpareAtivo$ dispara. Esse disparo representa o começo da operação do módulo redundante. O tempo associado à transição temporizada $SpareAtivo$ representa o *Mean Time to Active* (MTA). Já a transição imediata $SpareNAtivo$ representa o retorno do módulo principal para o modo operacional. As transições $MTTF\_Componente$, $MTTF\_Spare$, $MTTF\_OPSpare$, $MTTR\_Componente$, $MTTR\_Spare$ e $MTTR\_OPSpare$ representam a ocorrência de um evento de falha e de uma atividade de reparo dos módulos principal e reserva. Os tempos associados a essas transições temporizadas representam o MTTF e MTTR do módulo principal ($MTTFC$ e $MTTRC$), o MTTF e MTTR do módulo redundante em modo ativo ($MTTFOPS$ e $MTTROPS$) e o MTTF e MTTR do módulo redundante em modo inativo ($MTTFS$ e $MTTRS$). Os atributos das transições do modelo *Warm Standby* (ver Figura 5.36) são apresentados na Tabela 5.22.

Tabela 5.22: Atributos das Transições - Modelo *Warm Standby*

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
MTTF_Componente	exp	MTTFC	-	-	SS
MTTR_Componente	exp	MTTRC	-	-	SS
MTTF_Spare	exp	MTTFS	-	-	SS
MTTR_Spare	exp	MTTRS	-	-	SS
MTTF_OPSpare	exp	MTTFOPS	-	-	SS
MTTR_OPSpare	exp	MTTROPS	-	-	SS
SpareAtivo	exp	MTA	-	-	SS
SpareNAtivo	im	-	1	1	-

A representação das redundâncias *cold standby* e *warm standby* através de um modelo RBD é complexa, uma vez que a estrutura do sistema é dinâmica e o tempo necessário para ativar o módulo redundante deverá ser representado. Assim, o Modelo *Cold Standby* e o Modelo *Warm Standby* são fornecidos por meio de uma SPN (MACIEL et al., 2012).

Validação do Modelo *Cold Standby* e do Modelo *Warm Standby*

O Modelo *Cold Standby* (ver Figura 5.35) e o Modelo *Warm Standby* (ver Figura 5.36) foram validados através da comparação dos resultados da disponibilidade desses modelos e do Modelo CTMC apresentado em (BAUER; ADAMS; EUSTACE, 2011). A Figura 5.30 apresenta o modelo CTMC dos módulos CLC, CC e NC da plataforma *Eucalyptus*. O MTTF e o MTTR dos módulos da plataforma *Eucalyptus* são 111111,11 horas e 8 horas, respectivamente. O modelo CTMC dos equipamentos de rede (ver Figura 5.31) representa o *switch* e o roteador da nuvem computacional. O MTTF e o MTTR dos equipamentos de rede da nuvem computacional são 10752,69 horas e 8 horas, respectivamente.

A Figura 5.37 apresenta o modelo CTMC das máquinas virtuais com redundância ativo-passivo. Os estados *simplex* e *duplex* representam uma e duas máquinas virtuais operacionais, respectivamente. O estado *ActiveDown* representa que o sistema envia requisições para a máquina virtual redundante após o intervalo de tempo da ocorrência do defeito na máquina virtual principal. O estado *duplex failure* retrata as duas máquinas virtuais em estado de falha. O MTTF e o MTTR das máquinas virtuais com redundância *cold standby* e *warm standby* são 588,24 horas e 8 horas, respectivamente.

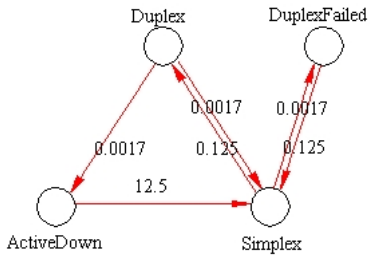


Figura 5.37: Modelo CTMC da Máquina Virtual da Plataforma *Eucalyptus* com Redundância Ativo-Passivo

O modelo CTMC da plataforma *Eucalyptus* (ver Figura 5.33) representa os módulos, equipamentos de rede e máquinas virtuais dessa plataforma de nuvem. A Tabela 5.23 apresenta os parâmetros de dependabilidade usados para validação do Modelo *Cold Standby* e do Modelo *Warm Standby*.

Tabela 5.23: Parâmetros de Dependabilidade dos Componentes da Plataforma *Eucalyptus*

Tipo	MTTF (horas)	MTTR (horas)	Tempo (horas)
CC,CLC,NC	329.067,64	8,00	-
RT	80.000,00	8,00	-
SW	11.200,00	8,00	-
VM, Spare_VM,OPSpare_VM	576,00	8,00	-
SpareAtivo (*Warm Standby*)	-	-	0,08
SpareAtivo (*Cold Standby*)	-	-	0,16

Os resultados da disponibilidade do Modelo *Cold Standby* e do Modelo *Warm Standby* e do modelo de redundância ativo-passivo apresentado em (BAUER; ADAMS; EUSTACE, 2011) foram comparados; para tanto foram adotados os mesmos parâmetros de dependabilidade (ver Tabela 5.19). O resultado da disponibilidade do Modelo *Cold Standby* e modelo ativo-passivo foi 99,87% e o resultado da disponibilidade do Modelo *Warm Standby* foi 99,91%. Esse resultado mostra que o Modelo *Cold Standby* e o Modelo *Warm Standby* estão validados.

Modelo de Manutenção

O Modelo de Manutenção é baseado em redes de *Petri* estocásticas e representa a alocação de equipes de manutenção para manutenção corretiva de um componente da nuvem computacional. O modo operacional do componente é $OM_{MA} = (Componente\_ON = 0)$. A Figura 5.38 mostra o modelo SPN adotado para estimar a disponibilidade de um componente da nuvem computacional. Os atributos das transições do Modelo de Manutenção são apresentados

na Tabela 5.24.

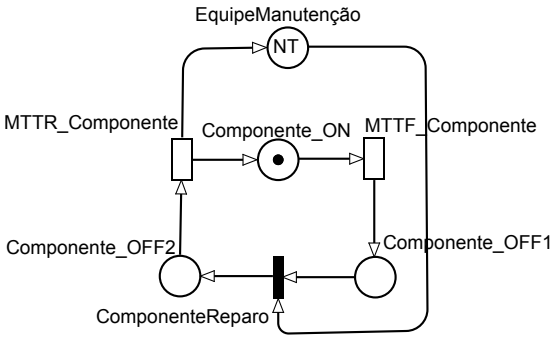


Figura 5.38: Modelo de Manutenção

Tabela 5.24: Atributos das Transições - Modelo de Manutenção

Transição	Tipo	Tempo	Peso	Prioridade	Concorrência
$MTTF\_Componente$	exp	X_{MTTF}	-	-	SS
$MTTR\_Componente$	exp	X_{MTTR}	-	-	SS
$ComponenteReparo$	im	-	1	1	-

Métricas de Dependabilidade

As métricas de dependabilidade empregadas para o planejamento da infraestrutura da nuvem privada são a disponibilidade e o *downtime*.

A métrica disponibilidade (A) representa a probabilidade do ambiente de nuvem estar operacional durante um determinado período de tempo, ou que tenha sido restaurado após a ocorrência de um evento de falha. Essa métrica é representada em porcentagem (%).

A métrica *downtime* (D) representa o período de tempo em que a infraestrutura de nuvem não está operacional devido a ocorrência de um evento de falha ou atividade de reparo. Essa métrica é representada em unidade de tempo.

Os modelos de disponibilidade das plataformas de nuvem quantificam a disponibilidade das infraestruturas das plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack*. A disponibilidade de N blocos conectados em série é obtida através de $P_{PN} = \prod_{i=1}^{n} P_i$. $P_i(t)$ descreve a disponibilidade instantânea $A_i(t)$ e de estado estacionário A_i do bloco P_i (MACIEL et al., 2012). Essa expressão indica que a nuvem computacional estará operacional quando nenhum dos componente das plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack* falhar (KUO; ZUO, 2003; RUPE, 2003). O *downtime* é representado pela expressão $DT_{PN} = (1 - P_{PN}) \times T$, onde P_{PN} descreve a disponibilidade da infraestrutura de nuvem computacional e T representa o período de tempo.

O Modelo de Redundante Ativo-Ativo quantifica a disponibilidade e o *downtime* das infraestruturas de computação em nuvem. A disponibilidade é representada pela expressão

$DP_{N+1} = (P\{exp\})$ onde $exp = ((\#CLC\_ON = 1)\, AND\, (\#CC\_ON = 1)\, AND\, (\#NC\_ON = 1)\, AND\, (\#SW\_ON = 1)\, AND\, (\#RT\_ON = 1)\, AND\, (\#VM1\_ON = 1\, OR\, \#VM2\_ON = 1))$ (ZIMMERMANN et al., 2006). Essa expressão significa que a infraestrutura de nuvem estará em modo operacional quando houver uma marcação nos lugares $CLC\_ON = 1$, CC_ON, NC_ON, SW_ON, RT_ON, $VM1\_ON$ ou CLC_ON, CC_ON, NC_ON, SW_ON, RT_ON, $VM2\_ON$. O *downtime* é representado pela expressão $DT_{N+1} = (1 - DP_{N+1}) \times T$, onde DP_{N+1} descreve a disponibilidade da infraestrutura de nuvem computacional.

O Modelo *Hot Standby* quantifica a disponibilidade das infraestruturas de nuvem computacional. A disponibilidade de dois blocos conectados em paralelo é obtida através de $P_{HS} = 1 - \prod_{i=1}^{n}(1 - P_i)$. $P_i(t)$ descreve a disponibilidade instantânea $A_i(t)$ e de estado estacionário A_i do bloco P_i (MACIEL et al., 2012). Essa expressão indica que o sistema cliente não estará operacional quando os componentes principal e redundante da nuvem computacional falharem (KUO; ZUO, 2003; RUPE, 2003). O *downtime* é representado pela expressão $DT_{HS} = (1 - P_{HS}) \times T$, onde P_{HS} descreve a disponibilidade da infraestrutura de nuvem computacional.

O Modelo *Cold Standby* quantifica a disponibilidade e o *downtime* das infraestruturas de computação em nuvem. A disponibilidade é representada pela expressão $DP_{CS} = (P\{exp\})$ onde $exp = \#Componente\_ON = 1\ OR\ \#Spare\_ON = 1$
(ZIMMERMANN et al., 2006). Essa expressão significa que a infraestrutura de nuvem estará em modo operacional quando houver pelo menos uma marcação nos lugares $Componente\_ON$ ou $Spare\_ON$. O *downtime* é representado pela expressão $DT_{CS} = (1 - DP_{CS}) \times T$, onde DP_{CS} descreve a disponibilidade da infraestrutura de nuvem computacional.

O Modelo *Warm Standby* quantifica a disponibilidade e o *downtime* das infraestruturas de nuvem computacional. A disponibilidade é representada pela expressão $DP_{WS} = (P\{exp\})$ onde $exp = \#Componente\_ON = 1\ OR\ \#OPSpare\_ON = 1$ (ZIMMERMANN et al., 2006). Essa expressão significa que a nuvem computacional estará em modo operacional quando houver pelo menos uma marcação nos lugares $Componente\_ON$ ou $OPSpare\_ON$. O *downtime* é representado pela expressão $DT_{WS} = (1 - DP_{WS}) \times T$, onde DP_{WS} descreve a disponibilidade da infraestrutura de computação em nuvem.

O Modelo de Manutenção quantifica a disponibilidade e o *downtime* das infraestruturas de nuvem computacional. A disponibilidade é representada pela expressão $DP_{MC} = (P\{exp\})$ onde $exp = \#Componente\_ON = 1$ (ZIMMERMANN et al., 2006). Essa expressão significa que a nuvem computacional estará em modo operacional quando houver uma marcação no lugar $Componente\_ON$. O *downtime* é representado pela expressão $DT_{MC} = (1 - DP_{MC}) \times T$, onde DP_{MC} descreve a disponibilidade da infraestrutura de computação em nuvem.

Tabela 5.25: Métricas de Dependabilidade

Modelo	Métrica	Expressão	Significado
Plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack*	P_{PN}	$\prod_{i=1}^{n} P_i$	A disponibilidade (A) de N blocos conectados em série.
Plataformas *Eucalyptus*, *Nimbus*, *OpenNebula* e *OpenStack*	DT_{PN}	$(1-P_{PN}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.
Ativo-Ativo	DP_{N+1}	$P\{((\#CLC\_ON = 1)\ AND\ (\#CC\_ON = 1)\ AND\ (\#NC\_ON = 1)\ AND\ (\#SW\_ON = 1)\ AND\ (\#RT\_ON = 1)\ AND\ (\#VM1\_ON = 1\ OR\ \#VM2\_ON = 1))\}$	O ambiente de nuvem estará em modo operacional quando houver uma marcação nos lugares CLC_ON, CC_ON, NC_ON, SW_ON, RT_ON, $VM1\_ON$ ou CLC_ON, CC_ON, NC_ON, SW_ON, RT_ON, $VM2\_ON$.
Ativo-Ativo	DT_{N+1}	$(1-DP_{N+1}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.
Hot Standby	P_{HS}	$1-\prod_{i=1}^{n}(1-P_i)$	A disponibilidade (A) de N blocos conectados em paralelo.
Hot Standby	DT_{HS}	$(1-P_{HS}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.
Cold Standby	DP_{CS}	$P\{\#Componente\_ON = 1\ OR\ \#Spare\_ON = 1\}$	O ambiente de nuvem estará em modo operacional quando houver pelo menos uma marcação nos lugares $Componente\_ON$ ou $Spare\_ON$.
Cold Standby	DT_{CS}	$(1-DP_{CS}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.

Warm Standby	DP_{WS}	$P\{\#Componente\_ON = 1$ $OR\ \#OPSpare\_ON = 1\}$	A nuvem computacional estará em modo operacional quando houver pelo menos uma marcação nos lugares *Componente_ON* ou *OPSpare_ON*
Warm Standby	DT_{WS}	$(1 - DP_{WS}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.
Manutenção	DP_{MC}	$P\{\#Componente\_ON = 1$	A nuvem computacional estará em modo operacional quando houver uma marcação no lugar *Componente_ON*
Manutenção	DT_{MC}	$(1 - DP_{MC}) \times T$	O período de tempo em que a infraestrutura de nuvem não está operacional.

Validação da Estratégia de Modelagem Proposta

A validação da estratégia de modelagem de disponibilidade adotada nesse trabalho foi baseada na comparação dos resultados da disponibilidade calculados por meio dessa estratégia de modelagem e dos resultados da disponibilidade calculados através da estratégia de modelagem de disponibilidade validada em (WEI B.; KONG, 2011).

A estratégia de modelagem de disponibilidade adotada em (WEI B.; KONG, 2011) propõe a modelagem hierárquica e heterogênea de *data centers* virtuais da computação em nuvem. Diagramas de bloco de confiabilidade (RBD) e redes de *Petri* estocásticas generalizadas (GSPN) são combinados hierarquicamente para avaliação da disponibilidade de *data centers* virtuais da computação em nuvem. Nessa estratégia de modelagem, um modelo de alto nível baseado em RBD descreve toda a infraestrutura dos VDCs para nuvem computacional. Esse modelo RBD descreve *clusters* compostos por M servidores, onde cada servidor contém N máquinas virtuais, um monitor de máquina virtual e o *hardware* da máquina física e cada *cluster* é conectado por um equipamento de rede. O modelo GSPN representa um componente de um VDC em estado de falha ou de reparo.

O cenário usado para validação da estratégia de modelagem de disponibilidade proposta nesse trabalho foi o *Moodle* hospedado na plataforma *Eucalyptus*. A plataforma de nuvem foi

configurada em três máquinas físicas, cada uma destinada ao controlador de nuvem (CLC), ao controlador de *cluster* (CC) e ao controlador de nó (NC).

Uma máquina virtual foi instanciada na máquina física que executa o serviço do NC. Os recursos computacionais dessa máquina física foram modelados através do Modelo do Sistema Computacional (Ver a Figura 5.25). A Tabela 5.26 mostra os MTTF's e MTTR's (KIM; MACHIDA; TRIVEDI, 2009) dos recursos computacionais da máquina física que executa o serviço do NC.

Tabela 5.26: Parâmetros de Dependabilidade da Máquina Física

Tipo	MTTF (horas)	MTTR (horas)
Infraestrutura de Processamento	2.500.000,00	8,00
Memória	480.000,00	8,00
Memória Secundária (Disco)	1.800.000,00	8,00

O Modelo da Máquina Virtual (Ver Figura 5.26) representa uma máquina virtual instanciada na máquina física onde o serviço do NC é executado. O MTTF e o MTTR da máquina virtual são 2.880,00 horas e 8 horas, respectivamente (KIM; MACHIDA; TRIVEDI, 2009).

Os resultados do Modelo do Sistema Computacional e do Modelo da Máquina Virtual são usados como parâmetros de dependabilidade para o Modelo da Plataforma *Eucalyptus*. O Modelo da Plataforma *Eucalyptus* (Ver Figura 5.39) representa a plataforma de nuvem que hospeda o *Moodle*. A Tabela 5.27 mostra os MTTF's e MTTR's (KIM; MACHIDA; TRIVEDI, 2009) dos componentes da plataforma *Eucalyptus*.

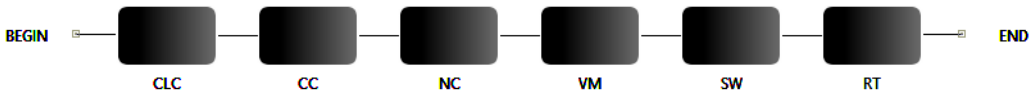


Figura 5.39: Modelo da Plataforma *Eucalyptus*

Tabela 5.27: Parâmetros de Dependabilidade dos Componentes da Plataforma *Eucalyptus*

Tipo	MTTF (horas)	MTTR (horas)
CC/CLC/NC	329.067,64	8,00
Roteador (RT)	80.000,00	8,00
Switch (SW)	11.200,00	8,00
Máquina Virtual (VM)	576,00	8,00

O mecanismo de redundância *hot standby* foi atribuído aos componentes da plataforma *Eucalyptus*. A Figura 5.40 apresenta o modelo dessa plataforma com uma redundância *hot standby* atribuído ao componente CLC. Os MTTF's e MTTR's (KIM; MACHIDA; TRIVEDI, 2009) dos componentes redundantes são apresentados na Tabela 5.27.

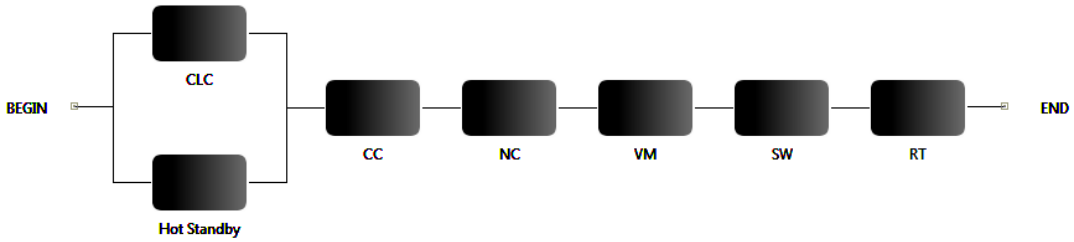


Figura 5.40: Modelo da Plataforma *Eucalyptus* com a Atribuição do Mecanismo de Redundância *Hot Standby*

As métricas $P_S = \prod_{i=1}^{n} P_i\ (A_i)$ e $P_P = 1 - \prod_{i=1}^{n} (1 - P_i)\ (A_i)$ foram usadas para obtenção da disponibilidade da nuvem computacional por meio do Modelo da Plataforma *Eucalyptus* com a atribuição do Mecanismo de Redundância *Hot Standby* (ver Figura 5.40).

A Tabela 5.28 apresenta os resultados da disponibilidade da estratégia de modelagem adotada nesta tese (Estratégia 1) e da estratégia de modelagem validada em (WEI B.; KONG, 2011) (Estratégia 2). O teste t emparelhado foi aplicado as disponibilidades das duas estratégias de modelagem. Considerando um nível de significância de 5%, o teste t emparelhado gerou um intervalo de confiança de (-0,0859, 0,1631). Como o intervalo de confiança contém 0, não há evidências estatísticas para rejeitar a hipótese de equivalência entre as disponibilidades das duas estratégias de modelagem (BAKOUCH, 2011).

Tabela 5.28: Resultados da Disponibilidade das Estratégias de Modelagem de Disponibilidade

Redundância *Hot Standby*	Estratégia 1	Estratégia 2
-	98,5409 (%)	98,5427 (%)
CLC/CC/NC	98,5427 (%)	98,5451 (%)
Máquina Virtual	99,8923 (%)	99,8926 (%)
Switch (Sw)	98,5526 (%)	98,6130 (%)
Roteador (RT)	98,8923 (%)	98,5525 (%)

A estratégia de modelagem de disponibilidade adotada nesta tese foi validada com base na estratégia de modelagem de disponibilidade do artigo (WEI B.; KONG, 2011). Da mesma forma que a estratégia de modelagem apresentada em (WEI B.; KONG, 2011), a estratégia de modelagem proposta combina as vantagens de modelos RBD e SPN para representação de infraestruturas de nuvem. Entretanto, a estratégia de modelagem proposta também permite a

representação de diferentes níveis de dependência entre os sistemas das infraestruturas de nuvem e seus módulos de redundância.

5.1.3 Modelos de Custo

Esta seção apresenta os modelos de custo propostos para o planejamento de infraestruturas de nuvens privadas (ver Seção 4.4). Os modelos propostos são: Modelo de Custo de Aquisição de Equipamentos do Sistema de TI, Modelo de Custo de Aquisição de Equipamentos e *Softwares* Redundantes, Modelo de Custo de Aquisição de Licenças de *Software*, Modelo de Custo da Equipe Técnica, Modelo de Custo da Equipe de Manutenção e Modelo de Custo da Substituição de Equipamentos. Esses modelos de custo são baseados em expressões matemáticas e métricas mapeadas nos modelos de disponibilidade.

Modelo de Custo de Aquisição de Equipamentos do Sistema de TI

Esse modelo é relativo aos gastos com os equipamentos do sistema de TI, que são os servidores, os roteadores e os *switches*. Esses custos são obtidos através da Equação 5.3 (SOUSA et al., 2014b, 2013).

$$\sum_{i=1}^{N} ETIN_i \times ETIC_i \tag{5.3}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de equipamentos do sistema de TI, como o servidor, o roteador e o *switch*.

ETIN: número de determinado tipo de equipamento do sistema de TI.

ETIC: custo unitário de determinado tipo de equipamento do sistema de TI.

Modelo de Custo de Aquisição de Licenças de *Software*

Esse modelo é composto pelo custo com os *softwares* usados na configuração da nuvem privada. Banco de dados, balanceador de carga, monitor de máquina virtual, plataforma de nuvem, sistema cliente, sistema operacional e servidor *web* são os *softwares* usados na configuração da nuvem privada. Os custos de *software* são obtidos por meio da Equação 5.4 (SOUSA et al., 2014b, 2013).

$$\sum_{i=1}^{N} SN_i \times SC_i \tag{5.4}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de *softwares*.

SN: número de determinado tipo de *software*.

SC: custo unitário de determinado tipo de *software*.

Modelo de Custo da Equipe Técnica

Esse modelo é composto pelos gastos com o pagamento da equipe técnica os quais são obtidos por meio da Equação 5.5.

$$\sum_{i=1}^{N} ETN_i \times ETC_i \times ETT_i \tag{5.5}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de especialidades dos componentes da equipe técnica.

ETN: número de componentes da equipe técnica com determinada especialidade.

ETC: custo unitário dos componentes da equipe técnica com determinada especialidade.

ETT: tempo de trabalho dos componentes da equipe técnica com determinada especialidade.

Modelo de Custo de Aquisição de Equipamentos e *Softwares* Redundantes

Esse modelo é composto pelos gastos com os componentes redundantes na nuvem privada, os quais são os mecanismos de redundâncias do tipo ativo-ativo, *cold standby*, *hot standby* e *warm standby* (KUO; ZUO, 2003; RUPE, 2003). Esse custo está associado aos cenários de infraestruturas de nuvens com vários mecanismos de redundância atribuídos aos seus componentes. Os custos dos mecanismos de redundância são obtidos através da Equação 5.6 (SOUSA et al., 2014b, 2013).

$$\sum_{i=1}^{N} ERN_i \times ERC_i \tag{5.6}$$

Os componentes dessa equação são apresentados abaixo:

N: redundâncias do tipo ativo-ativo, *cold standby*, *hot standby* e *warm standby* para os equipamentos e *softwares*.

ERN: número de determinado tipo de redundância para os equipamentos e *softwares*.

ERC: custo unitário de determinado tipo de redundância para os equipamentos e *softwares*.

Modelo de Custo da Equipe de Manutenção

Esse modelo é composto pelos gastos com o pagamento da equipe de manutenção os quais são obtidos por meio da Equação 5.7 (CALLOU et al., 2010; SOUSA et al., 2012).

$$\sum_{i=1}^{N} EMN_i \times EMC_i \times EMT_i \tag{5.7}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de especialidades dos componentes da equipe de manutenção.

EMN: número de componentes da equipe de manutenção com determinada especialidade.

EMC: custo unitário de componentes da equipe de manutenção com determinada especialidade.

EMT: tempo de trabalho de componentes da equipe de manutenção com determinada especialidade.

O custo da equipe de manutenção também pode ser representado pela Equação 5.8. Essa expressão pode ser mapeada no Modelo de Manutenção (ver Figura 5.38).

$$\sum_{i=1}^{N} EMN(m(EquipeManutencao_i)) \times EMC(EquipeManutencao_i) \times EMT \tag{5.8}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de especialidades dos componentes da equipe de manutenção.

$m(EquipeManutencao_i)$: número de marcações do lugar $EquipeManutencao_i$ do Modelo Manutenção (ver Figura 5.38).

$EMN(m(EquipeManutencao_i))$: número esperado de componentes da equipe de manutenção com uma especialidade específica.

$EMC(EquipeManutencao_i)$: custo do tempo de trabalho dos componentes da equipe de manutenção com uma especialidade específica.

EMT: tempo de trabalho de componentes da equipe de manutenção com determinada especialidade.

Modelo de Custo da Substituição de Equipamentos

Esse modelo é composto pelos gastos com a substituição dos equipamentos danificados os quais são obtidos por meio da Equação 5.9 (CALLOU et al., 2010).

$$\sum_{i=1}^{N} SEN_i \times EMT_i \times SEC_i \tag{5.9}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de equipamentos substituídos que podem ser o servidor, o roteador e o *switch*.

SEN: número de determinado tipo de equipamento substituído.

EMT: tempo de manutenção do equipamento.

SEC: custo unitário de determinado tipo de equipamento substituído.

O custo da substituição de equipamentos também pode ser representado pela Equação 5.10. Essa expressão pode ser mapeada no Modelo de Manutenção (ver Figura 5.38).

$$\sum_{i=1}^{N} (tp(ra_i) \times EMT) \times SEC(ra_i) \tag{5.10}$$

Os componentes dessa equação são apresentados abaixo:

N: tipos de equipamentos substituídos que podem ser o servidor, o roteador e o *switch*.

ra_i: transição temporizada $MTTR\_Componente$ que representa as atividades de reparo do componente no Modelo Manutenção (ver Figura 5.38).

$tp(ra_i)$: *throughput* da transição temporizada $MTTR\_Componente$ que representa as atividades de reparo no Modelo Manutenção (ver Figura 5.38).

EMT: tempo de manutenção do equipamento.

$SEC(ra_i)$: custo da substituição de um equipamento específico por atividade de manutenção.

Aplicação dos Modelos de Custo

O cenário apresentado anteriormente na Seção 5.1.2 foi usado para aplicação dos modelos de custo nesse trabalho. Esse cenário é o *Moodle* hospedado na plataforma *Eucalyptus*. Essa plataforma de nuvem foi configurada em três máquinas físicas, cada uma destinada ao controlador de nuvem (CLC), ao controlador de *cluster* (CC) e ao controlador de nó (NC). As máquinas físicas onde os componentes da plataforma *Eucalyptus* estão configurados são conectadas por meio de um *switch* e um roteador.

Uma máquina virtual composta de um processador de dois núcleos, uma memória de 2GB e uma memória secundária de 80GB foi instanciada na máquina física que executa o serviço do NC. Essa máquina virtual foi configurada com a plataforma *Eucalyptus 3.4* (EUCALYPTUS, 2013), o *Moodle 2.6.2* (MOODLE, 2013), o *MySQL 5.5.31* (MYSQL, 2013), o *Ubuntu 13.10* (UBUNTU, 2013) e o servidor *Apache 2.0* (APACHE, 2013). A Tabela 5.29 apresenta os custos unitários e os custos totais da aquisição de equipamentos do sistema de TI e da aquisição de licenças de *software*. O custo unitário do *Apache 2.0* (APACHE, 2013), *Moodle 2.6.2* (MOODLE, 2013) e *Ubuntu 13.10* é $(US\$)$ 0.00. O custo total da aquisição de equipamentos do sistema de TI (ver Equação 5.3) e da aquisição de licenças de *software* (ver Equação 5.4) foi $(US\$)$ 6,000.00.

Tabela 5.29: Parâmetros de Custo de Equipamentos e de *Software*

Componente	Custo Unitário (US$)	Quantidade	Custo Total (US$)
Banco de Dados	2,000.00	1	2,000.00
Máquina Física	500.00	3	1.500.00
Plataforma de Nuvem	1,500.00	1	1,500.00
Roteador, *Switch*	500.00	1	500.00

O custo unitário do dia de trabalho da equipe técnica para o cenário apresentado foi $US\$$ 50.00. O custo anual total da equipe técnica composta de um especialista na plataforma *Eucalyptus* (ver Equação 5.5) foi $US\$$ 18,250.00.

O cenário adotado apresentou uma disponibilidade de 98,5427% quando o mecanismo de redundância *hot standby* foi atribuído ao CLC da plataforma *Eucalyptus*. O custo da aquisição desse equipamento redundante (ver Equação 5.6) foi *US*$ 500.00. O custo unitário da hora de trabalho da equipe de manutenção que é composta por um especialista na plataforma *Eucalyptus* foi *US*$ 20.00. Esse cenário apresentou um *downtime* de 127,66 horas/ano. O custo anual total da equipe de manutenção foi *US*$ 2,553.19. O custo unitário para a substituição do servidor, roteador ou *switch* foi *US*$ 500.00. O custo anual total para substituição desses equipamentos foi *US*$ 485.46.

O custo total do cenário apresentado na Seção 5.1.2 foi composto do custo da aquisição de equipamentos de TI, da aquisição de licenças de *software*, da equipe técnica, da aquisição de equipamentos e *softwares* redundantes, da equipe de manutenção e da substituição de equipamentos. O valor desse custo foi *US*$ 27,788.65.

5.2 Modelo de Otimização

Esta seção apresenta o modelo para geração de cenários de desempenho e custo e o modelo para geração de cenários de disponibilidade e custo.

5.2.1 Modelo para Geração de Cenários de Desempenho e Custo

Esta seção apresenta o modelo proposto para a geração de infraestruturas de nuvens privadas, considerando requisitos de desempenho e custo (ver Seção 4.1).

O Modelo para Geração de Cenários de Desempenho e Custo provê cenários com diferentes configurações de *hardware* e *software* para nuvem privada. Essas configurações de *hardware* e *software* são compostas por conjuntos de *software* (com diferentes balanceadores de carga, bancos de dados, monitores de máquina virtual, plataformas de nuvem, servidores *web* e sistemas operacionais, entre outros) e conjuntos de *hardware* (com diferentes dimensionamentos da infraestrutura de processamento, memória e memória secundária). O modelo proposto proporciona a criação de cenários de infraestruturas de nuvem por meio da atribuição de diferentes conjuntos de *software* a vários conjuntos de *hardware*. Como exemplo, o conjunto de *software* composto pelo balanceador de carga *Linux Virtual Server*, banco de dados *MySQL*, monitor de máquina virtual *KVM*, plataforma *Eucalyptus*, servidor *web Apache* e sistema operacional *Ubuntu* pode ser configurado em uma infraestrutura de nuvem composta por máquinas virtuais com 2 núcleos de processamento, memória de 2GB e memória secundária de 80GB (ver Figura 5.41).

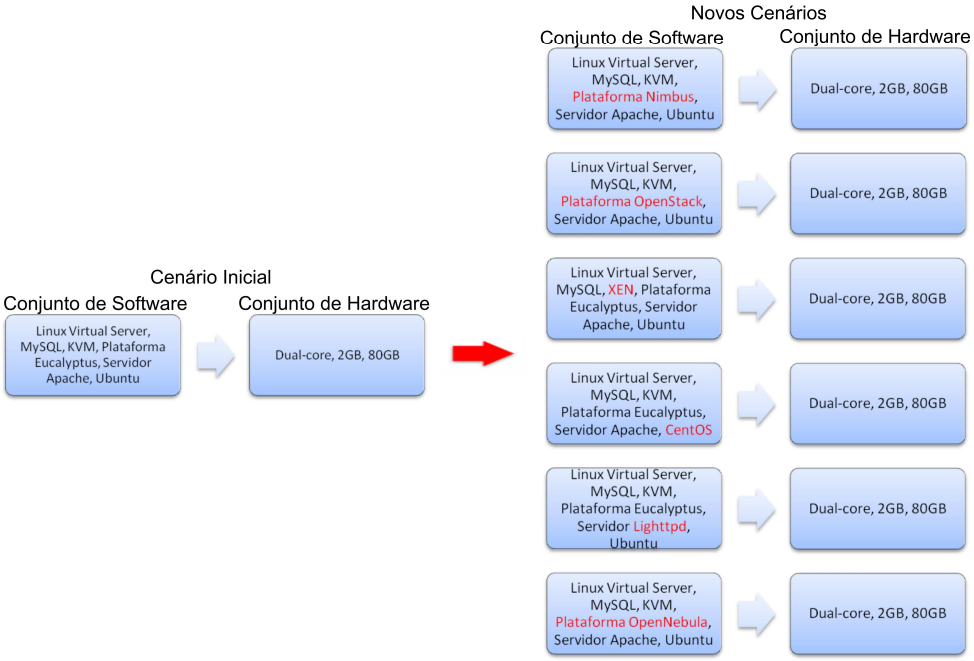


Figura 5.41: Geração do Cenário de Desempenho e Custo

O Modelo para Geração de Cenários de Desempenho e Custo foi baseado na meta-heurística GRASP e portanto, consiste da fase construtiva da solução inicial e da fase de busca local da solução inicial. O Algoritmo 4 mostra o Modelo para Geração de Cenários de Desempenho e Custo. Os dados de entrada do algoritmo são os conjuntos de *software* (*Software Set - SS*) e os conjuntos de *hardware* (*Hardware Set - HS*) e o dado de saída do algoritmo é um vetor de atribuição s^* especificando o conjunto de *software* atribuído a cada conjunto de *hardware*.

O conjunto elite de soluções para infraestrutura de nuvem computacional é inicializado com valor 0 na linha 1. Um número máximo de interações $MaxInter$ é computado da linha 2 a 21. Durante cada iteração, uma solução aleatória e gulosa $s^{'}$ é gerada na linha 3. Se o conjunto elite de soluções $f(s)$ não tiver pelo menos ρ elementos, e se $s^{'}$ é viável e suficientemente diferente de todas as outras soluções do conjunto elite de soluções $f(s)$, $s^{'}$ será adicionado ao conjunto elite na linha 19. Se o conjunto elite de soluções $f(s)$ tem pelo menos ρ elementos, então os passos nas linhas 5 a 17 são computados.

A fase de construção aleatória e gulosa não garante a geração de uma solução viável. Se essa fase retorna uma solução não viável, a solução viável $s^{'}$ é selecionada aleatoriamente do conjunto elite de soluções $f(s)$ na linha 6.

A fase de busca local utiliza a solução $s^{'}$ como ponto de partida na linha 8, resultando no mínimo local aproximado $s^{'}$. Se o conjunto elite de soluções $f(s)$ está completo, e se $s^{'}$ é uma solução melhor que a pior solução elite, e $s^{'} \neq f(s)$, então essa solução será adicionada ao conjunto elite de soluções $f(s)$ na linha 12. Entre todas as soluções elite com um custo pior que o de $s^{'}$, a solução s mais similar a $s^{'}$ é selecionada para ser removida do conjunto elite de soluções de $f(s)$. Entretanto, se o conjunto elite de soluções não está completo, $s^{'}$ é adicionado

ao conjunto elite de soluções na linha 16, se $s^{'} \neq f(s)$.

```
1:  f(s) := 0;
2:  for i = 1 to i = MaxInter do
3:     s' := GreedRandomized();
4:     if elite set f(s) has at least ρ elements then
5:        if s' is not feasible then
6:           Randomly select a new solution s' ∈ f(s);
7:        end if
8:        s' := ApproxLocalSearch(s');
9:        Randomly select a solution s' ∈ f(s);
10:       if elite set f(s) is full then
11:          if c(s') ≤ maxc(s)|s ∈ f(s) and s' ≠ f(s) then
12:             Replace the element most similar to s' among all;
13:             elements with cost worst than s';
14:          end if
15:       else if s' ≠ f(s) then
16:          f(s) := f(s) ∪ s';
17:       end if
18:    else if s' is feasible and s' ≠ f(s) then
19:       f(s) := f(s) ∪ s*;
20:    end if
21: end for
22: return s* = minc(s)|s ∈ f(s);
```

Algoritmo 4: *GRASP(HS,SS,MaxInter,MaxCLS)*

Fase Construção

A fase de construção proporciona a atribuição dos conjuntos de *software* aos conjuntos de *hardware* da infraestrutura da nuvem privada.

O Algoritmo 5 mostra o pseudo-código da fase de construção. O número de conjuntos de *software* (*Software Set Number - SSN*) e o número de conjuntos de *hardware* (*Hardware Set Number - HSN*) são inicializados com valor 0. Já o número máximo de conjuntos de *software* (*Maximum Number of Software Set - MNSS*) e o número máximo de conjuntos de *hardware* (*Maximum Number of Hardware Set - MNHS*) são inicializados com valor *N* na linha 1. Na linha 3, o número de conjuntos de *software* (*SSN*) é gerado aleatoriamente até um número máximo *MNSS*. Cada conjunto de *software* (*SSI*) será adicionado ao conjunto de *software* (*SS*) na linha 4. Na linha 6, o conjunto de *hardware* (*HSI*) é gerado aleatoriamente até um número máximo *MNHS*. Cada conjunto de *hardware* (*HSI*) será adicionado ao conjunto de *hardware* (*Hardware Set - HS*) na linha 7. Nas linhas 8 e 9, o conjunto de *software* (*SSI*) será selecionando aleatoriamente e será atribuído a cada conjunto de *hardware* (*HSI*) gerado.

1: $SSN := 0$; $HSN := 0$; $MNSS := N$; $MNHS := N$;
2: **for** $SSN = 0$ *to* $SSN = MNSS$ **do**
3: *Randomly generate a software set* $SSI \in S$;
4: $SS := SS \cup SSI$;
5: **for** $HSN = 0$ *to* $HSN = MNHS$ **do**
6: *Randomly generate a hardware set* $HSI \in HS$;
7: $HS := HS \cup HSI$;
8: *Randomly select a software set* $SSI \in S$;
9: *Assign a software set SSI to a hardware set HSI*;
10: **end for**
11: **end for**
12: **return** *assignment* $s \in f(s)$;

Algoritmo 5: *GreedyRandomizedConstruction(Seed)*

Fase Busca Local

A fase de busca local provê uma solução que é o mínimo local a partir da solução s produzida pela fase de construção. Essa fase utiliza a estrutura de vizinhança conhecida como 1-*move* (MATEUS; RESENDE; SILVA, 2011). Nessa estrutura de vizinhança, a solução s é obtida pela modificação de uma atribuição (conjunto de *software* $\longrightarrow$ conjunto de *hardware*) na solução s anterior. Se essa mudança resulta em uma primeira melhoria em relação a solução anterior, temos a busca local *first fit*. Mas se todas as modificações da atribuição (conjunto de *software* $\longrightarrow$ conjunto de *hardware*) forem avaliadas a partir de uma solução s com o objetivo de encontrar a melhor solução, temos a busca local *best fit*. Em ambas as variantes, a busca é repetida até que não haja uma solução melhor na vizinhança. Nesse trabalho, ao invés de avaliar todas as soluções da vizinhança, uma lista candidata *CLS* é criada com as melhores soluções. Uma das soluções dessa lista é selecionada aleatoriamente e um movimento é feito para essa solução. Dessa forma, nem todos os vizinhos são avaliados. Consequentemente, a melhor solução encontrada não pode ser um mínimo local. Essa solução é um mínimo local aproximado.

O Algoritmo 6 mostra o pseudo-código da fase de busca local. Essa fase tem como dados de entrada do algoritmo a solução s e os parâmetros $MaxCLS$ e $MaxInter$. As linhas 1 a 13 são repetidas até a produção do mínimo local aproximado. Na linha 2, o contador *count* e a lista candidata *CLS* são inicializados. A cada iteração do laço interno nas linhas 3 a 9, um movimento dos vizinhos de s é realizado sem a substituição da solução anterior através da função $Move(s)$ na linha 4. Se esse vizinho é uma melhor solução, ele é inserido na *CLS* na linha 6. Esse procedimento ocorrerá até a lista candidata tornar-se completa ou haver um número máximo de interações. Nas linhas 10 a 12, se a lista candidata não estiver vazia, uma solução $s \in CLS$ é escolhida aleatoriamente. Se a lista candidata estiver vazia após o processo de amostragem, o procedimento termina retornando a solução s como um mínimo local aproximado na linha 14.

1: *repeat*
2: $count := 0$; $CLS := \emptyset$;
3: *repeat*
4: $s^{'} := Move(s)$;
5: **if** $s^{'}$ *is feasible and* $cost(s^{'}) < cost(s)$ **then**
6: $CLS := CLS \cup s^{'}$;
7: **end if**
8: $count := count + 1$;
9: *until* $|count| \leq MaxCLS$ *or* $count \geq MaxInter$;
10: **if** $CLS \neq 0$ **then**
11: *Randomly Select a solution* $s \in CLS$;
12: **end if**
13: *until* $CLS = 0$;
14: **return** s;
Algoritmo 6: *Procedure ApproxLocalSearch(s,MaxInter,MaxCLS)*

Modelo Matemático

O modelo matemático provê a representação do problema de planejamento de infraestruturas de nuvens que atendam aos requisitos de desempenho e custo. Essas infraestruturas de nuvens privadas são compostas por diferentes configurações de *software* e *hardware*. O modelo matemático deve ser minimizado a fim de obter a melhor solução dentro do espaço de infraestruturas de nuvens privadas candidatas. As variáveis de decisão, objetivos e função objetivo são apresentados a seguir:

Variáveis de Decisão:
$HS = \{hs_1, ..., hs_i\}$ conjunto de *hardware* i da infraestrutura da nuvem privada;
$SS = \{ss_1, ..., ss_j\}$ conjunto de *software* j da infraestrutura da nuvem privada.

Objetivos:
f_1: minimizar o tempo de resposta.
f_2: minimizar a utilização do processador.
f_3: minimizar a utilização da memória.
f_4: minimizar o custo de aquisição de equipamentos do sistema de TI.
f_5: minimizar o custo de aquisição de licenças de *software*.
f_6: minimizar o custo da equipe técnica.

Função Objetivo:
Minimize $f = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 + \lambda_5 f_5 + \lambda_6 f_6$, onde $\lambda_{1...6}$ representa o peso dado para as funções $f_{1...6}$ e $\lambda_{1...6} = 1$.

Os parâmetros da função objetivo são os valores normalizados das métricas de desempenho e custo através da Equação 4.1 (GUIMARÃES, 2013; MONTGOMERRY; GEORGE, 2009).

Essas métricas normalizados são o tempo de resposta, a utilização de processador, a utilização de memória, o custo da aquisição de equipamentos do sistema de TI, o custo da aquisição de licenças de *software* e o custo da equipe técnica. Como restrição, esses valores normalizados devem estar compreendidos entre 0 e 1.

Restrição:
$HS_i > 0$, número de conjuntos de *hardwares* i maior que zero.
$SS_j > 0$, número de conjuntos *softwares* j maior que zero.
$\sum_{j=1}^{J} x_{ji} = 1 \; \forall \; hs$, número de conjunto de *software* j a serem configurados no conjunto de *hardware* i igual a 1.

5.2.2 Modelo para Geração de Cenários de Disponibilidade e Custo

Esta seção apresenta o Modelo para Geração de Cenários de Disponibilidade e Custo proposto para a geração de cenários da nuvem privada, considerando requisitos de dependabilidade e custo (ver Seção 4.1).

O Modelo para Geração de Cenários de Disponibilidade e Custo proporciona a geração de cenários de infraestruturas de nuvem por meio da atribuição de diferentes mecanismos de redundância (ativo-ativo, *cold standby*, *hot standby*, *warm standby* e nenhum mecanismo de redundância) (RUPE, 2003) aos componentes (CLC, CC, NC, VM, Roteador - RT e *Switch* - SW da plataforma *Eucalyptus* (EUCALYPTUS, 2013)).

O Modelo para Geração de Cenários de Disponibilidade e Custo foi baseado na meta-heurística GRASP e portanto, consiste da fase construtiva da solução inicial e da fase de busca local da solução inicial. O Algoritmo 7 mostra o Modelo para Geração de Cenários de Disponibilidade e Custo. Os dados de entrada do algoritmo são os tipos de equipamentos (*Equipment Type - ET*), o número de equipamentos (*Equipment Number - EN*), os tipos de mecanismos de redundância (*Redundancy Type - RT*) e o número máximo de tipos de redundância (*Maximum Number of Redundancy Type - MNRT*). O dado de saída do algoritmo é um vetor de atribuição s^* especificando o tipo de mecanismo de redundância atribuído a cada tipo de equipamento da computação em nuvem.

O conjunto elite de soluções para infraestrutura de nuvem é inicializado com valor 0 na linha 1. Um número máximo de interações *MaxInter* será computado da linha 2 a linha 21. Durante cada iteração, uma solução aleatória e gulosa $s^{'}$ será gerada na linha 3. Se o conjunto elite de soluções $f(s)$ não tiver pelo menos ρ elementos, e se $s^{'}$ é viável e suficientemente diferente de todas as outras soluções do conjunto elite de soluções $f(s)$, $s^{'}$ será adicionado ao conjunto elite na linha 19. Se o conjunto elite de soluções $f(s)$ tem pelo menos ρ elementos, os passos nas linhas 5 a 17 são computados.

A fase de construção aleatória e gulosa não garante a geração de uma solução viável. Se

essa fase retorna uma solução não viável, a solução viável $s^{'}$ é selecionada aleatoriamente do conjunto elite de soluções $f(s)$ na linha 6.

A fase de busca local utiliza a solução $s^{'}$ como ponto de partida na linha 8, resultando no mínimo local aproximado $s^{'}$. Se o conjunto elite de soluções $f(s)$ está completo, e então se $s^{'}$ é uma solução melhor que a pior solução elite, e $s^{'} \neq f(s)$, então essa solução será adicionada ao conjunto elite de soluções $f(s)$ na linha 12. Entre todas as soluções elite com um custo pior que o de $s^{'}$, a solução s mais similar a $s^{'}$ é selecionada para ser removida do conjunto elite de soluções de $f(s)$. Entretanto, se o conjunto elite de soluções não está completo, $s^{'}$ é adicionado ao conjunto elite de soluções na linha 16, se $s^{'} \neq f(s)$.

```
1:  f(s) := 0;
2:  for i = 1 to i = MaxInter do
3:     s' := GreedRandomized();
4:     if elite set f(s) has at least ρ elements then
5:        if s' is not feasible then
6:           Randomly select a new solution s' ∈ f(s);
7:        end if
8:        s' := ApproxLocalSearch(s');
9:        Randomly select a solution s' ∈ f(s);
10:       if elite set f(s) is full then
11:          if c(s') ≤ maxc(s)|s ∈ f(s) and s' ≠ f(s) then
12:             Replace the element most similar to s' among all;
13:             elements with cost worst than s';
14:          end if
15:       else if s' ≠ f(s) then
16:          f(s) := f(s) ∪ s';
17:       end if
18:    else if s' is feasible and s' ≠ f(s) then
19:       f(s) := f(s) ∪ s*;
20:    end if
21: end for
22: return s* = minc(s)|s ∈ f(s);
```

Algoritmo 7: *GRASP(ET, EN, RT, MaxInter, MaxCLS)*

Fase Construção

A fase de construção proporciona a atribuição de mecanismos de redundância aos equipamentos da nuvem privada. O Algoritmo 8 mostra o pseudo-código da fase de construção. O tipo de redundância (*Redundancy Type - RT*) e o tipo de equipamento (*Equipment Type - ET*) são inicializados com valor 0. Já o número máximo de tipos de redundância (*Maximum Number of Redundancy Type - MNRT*) é inicializado com valor *N* e o número máximo de tipos de equipamentos (*Maximum Number Equipment Type - MNET*) é inicializado com valor *M* na linha 1. Na linha 3, o mecanismo de redundância (*RT*) é gerado aleatoriamente até um número máximo *MNRT*. Cada mecanismo de redundância (*RTI*) será adicionado ao conjunto de mecanismos de redundância (*Redundancy Set - RS*) na linha 4. Na linha 7, o equipamento

(ET) é gerado aleatoriamente até um número máximo $MNET$. Cada equipamento (ETI) será adicionado ao conjunto de equipamentos (*Equipment Set - ES*) na linha 8. Na linha 9, mecanismo de redundância (RT) será selecionando aleatoriamente e será atribuído a cada equipamento (ET) da infraestrutura de nuvem computacional gerado.

1: $RT := 0$; $ET := 0$; $MNRT := N$; $MNET := M$;
2: **for** $RT = 0$ *to* $RT = MNRT$ **do**
3: *Randomly generate a redundancy* $RTI \in RS$;
4: $RS := RS \cup RTI$;
5: **end for**
6: **for** $ET = 0$ *to* $ET = MNET$ **do**
7: *Randomly generate a equipment type* $ETI \in ES$;
8: $ES := ES \cup ETI$;
9: *Randomly select a redundancy type* $RTI \in RS$;
10: *Assign redundancy type* RTI *to equipment type* ETI;
11: **end for**
12: **return** *assignment* $s \in f(s)$;

Algoritmo 8: *GreedyRandomizedConstruction(Seed)*

Fase Busca Local

A fase de busca local provê uma solução que é o mínimo local a partir da solução s produzida pela fase de construção. Essa fase utiliza a estrutura de vizinhança conhecida como 1-*move* (MATEUS; RESENDE; SILVA, 2011). Nessa estrutura de vizinhança, a solução s é obtida pela modificação de uma atribuição (mecanismo de redundância $\longrightarrow$ equipamento da nuvem computacional) na solução s anterior. Se essa mudança resulta em uma primeira melhoria em relação a solução anterior, temos a busca local *first fit*. Mas se todas as modificações da atribuição (mecanismo de redundância $\longrightarrow$ equipamento da nuvem computacional) forem avaliadas a partir de uma solução s com o objetivo de encontrar a melhor solução, temos a busca local *best fit*. Em ambas as variantes, a busca é repetida até que não haja uma solução melhor na vizinhança. Nesse trabalho, ao invés de avaliar todas as soluções da vizinhança, uma lista candidata *CLS* é criada com as melhores soluções. Uma das soluções dessa lista é selecionada aleatoriamente e um movimento é feito para essa solução. Dessa forma, nem todos os vizinhos são avaliados. Consequentemente, a melhor solução encontrada não pode ser um mínimo local. Essa solução é um mínimo local aproximado.

O Algoritmo 9 mostra o pseudo-código da fase de busca local. Essa fase tem como dados de entrada do algoritmo a solução s e os parâmetros $MaxCLS$ e $MaxInter$. As linhas 1 a 13 são repetidas até a produção do mínimo local aproximado. Na linha 2, o contador *count* e a lista candidata *CLS* são inicializados. A cada iteração do laço nas linhas 3 a 9, um movimento dos vizinhos de s é realizada sem a substituição da solução anterior através da função $Move(s)$ na linha 4. Se esse vizinho é uma melhor solução, ele é inserido na *CLS* na linha 6. Esse procedimento ocorrerá até a lista candidata tornar-se completa ou haver um número máximo de interações. Nas linhas 10 a 12, se a lista candidata não estiver vazia, uma solução $s \in CLS$ é

escolhida aleatoriamente. Se a lista candidata estiver vazia após o processo de amostragem, o procedimento termina retornando a solução s como um mínimo local aproximado na linha 14.

1: *repeat*
2: $count := 0$; $CLS := \emptyset$;
3: *repeat*
4: $s^{'} := Move(s)$;
5: **if** $s^{'}$ *is feasible and* $cost(s^{'}) < cost(s)$ **then**
6: $CLS := CLS \cup s^{'}$;
7: **end if**
8: $count := count + 1$;
9: *until* $|count| \leq MaxCLS$ *or* $count \geq MaxInter$;
10: **if** $CLS \neq 0$ **then**
11: *Randomly Select a solution* $s \in CLS$;
12: **end if**
13: *until* $CLS = 0$;
14: **return** s;
Algoritmo 9: *Procedure ApproxLocalSearch(s,MaxInter,MaxCLS)*

Como exemplo da geração de cenários de infraestruturas de nuvens temos os mecanismos de redundância *cold standby*, *warm standby*, *cold standby*, *hot standby*, ativo-ativo, ativo-ativo podem ser atribuídos aos componentes CLC, CC, NC, VM, SW, RT da plataforma *Eucalyptus* (ver Figura 5.42).

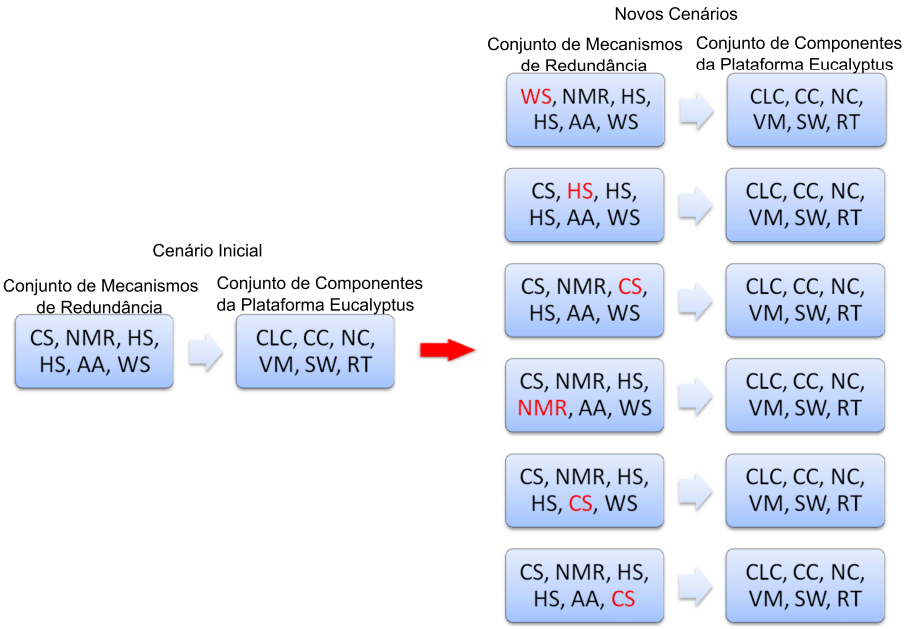


Figura 5.42: Geração do Cenário de Disponibilidade e Custo

Modelo Matemático

O modelo matemático provê a representação do problema de planejamento de infraestruturas de nuvens que atendam aos requisitos de dependabilidade e custo. Essas infraestruturas de

nuvens privadas são compostas por vários mecanismos de redundância. O modelo matemático deve ser minimizado a fim de obter a melhor solução dentro do espaço de infraestruturas de nuvens privadas candidatas. As variáveis de decisão, objetivos e função objetivo são apresentados a seguir:

Variáveis de Decisão:

$ET = \{et_1, ..., et_n\}$ tipos de equipamentos n (ex: CLC, CC, NC, VM, SW e RT) da nuvem privada;

$RT = \{rt_1, ..., rt_m\}$ tipos de mecanismos de redundância m (ex: ativo-ativo, *hot standby*, *cold standby*, *warm standby* e nenhum mecanismo de redundância).

Objetivos:

f_1: minimizar a indisponibilidade.

f_2: minimizar o *downtime*.

f_3: minimizar o custo de aquisição de equipamentos e *softwares* redundantes.

f_4: minimizar o custo da equipe de manutenção.

f_5: minimizar o custo da substituição de equipamentos.

Função Objetivo:

Minimizar $f = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 + \lambda_5 f_5$, onde $\lambda_{1...5}$ representa o peso dado para as funções $f_{1...5}$ e $\lambda_{1...5} = 1$.

Os parâmetros da função objetivo são os valores normalizados das métricas de dependabilidade e custo através da Equação 4.1 (GUIMARÃES, 2013; MONTGOMERRY; GEORGE, 2009). Essas métricas normalizados são a disponibilidade, o *downtime*, o custo da aquisição de equipamentos e *softwares* redundantes, o custo da substituição de equipamentos e o custo da equipe de manutenção. Como restrição, esses valores normalizadas devem estar compreendidos entre 0 e 1.

Restrição:

$ET_n > 0$, número de equipamentos n maior que zero.

$RT_m > 0$, número de mecanismos de redundâncias m maior que zero.

$RT_m \leq 5$, número de mecanismos de redundâncias m menor ou igual a 5.

$\sum_{m=1}^{5} x_{mn} = 1 \; \forall \; et$, número mecanismos de redundâncias m a serem atribuídos ao equipamento n igual a 1.

5.3 Considerações Finais

Este capítulo apresentou os modelos de otimização e os modelos de representação da solução integrada proposta para o planejamento de infraestruturas de nuvens privadas. Os modelos de otimização foram obtidos conforme o método de Geração de Cenários e os modelos de representação foram obtidos conforme os métodos de Geração de Modelo de Desempenho, Geração de Modelo de Disponibilidade e Geração de Modelo de Custo da metodologia proposta. Os modelos de otimização permitem a geração de soluções de infraestruturas de nuvens e os modelos de representação possibilitam a avaliação dessas soluções.

6

Gerador de Cenários e Modelos para o Planejamento de Infraestruturas de Nuvens Privadas

Este capítulo apresenta o Gerador de Cenários e Modelos para o Planejamento de Infraestruturas de Nuvens Privadas (*Scenarios and Models Generator for Private Cloud Infrastructure Planning - SMG4PCIP*) (SOUSA et al., 2013), uma ferramenta utilizada para implementar a metodologia de planejamento de infraestruturas de nuvens privadas proposta. A Figura 6.1 mostra a arquitetura do Gerador de Cenários e Modelos para o Planejamento de Infraestruturas de Nuvens Privadas.

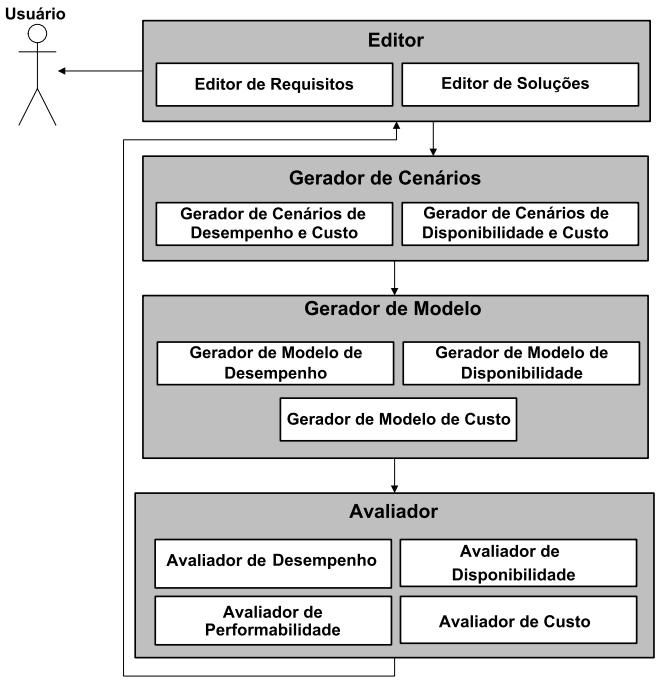


Figura 6.1: Gerador de Cenários e Modelos para o Planejamento de Infraestruturas de Nuvens Privadas

O SMG4PCIP (SOUSA et al., 2013) é composto de quatro módulos que são o Editor, o

Gerador de Cenários, o Gerador de Modelos e o Avaliador, e as seções a seguir apresentam esses módulos. O Editor permite que o usuário forneça as informações necessárias para o planejamento de infraestruturas de nuvens privadas. Essas informações são usadas pelo Gerador de Cenários para criação de cenários de infraestruturas de nuvens com diferentes configurações de *software* e cenários de infraestruturas de nuvens com vários mecanismos de redundância atribuídos aos seus componentes. O Gerador de Modelos proporciona a representação dos cenários gerados através de expressões matemáticas e modelos estocásticos. O Avaliador é responsável pela análise dos modelos que representam os cenários gerados e de acordo com os resultados dessa análise, este módulo escolhe os cenários. O usuário do SMG4PCIP pode ser um projetista com pouca ou nenhuma experiência em técnicas metaheurísticas e formalismos matemáticos.

6.1 Editor

O Editor é o portal *web* do Gerador de Cenários e Modelos para o Planejamento de Infraestruturas de Nuvens Privadas (SMG4PCIP) (SOUSA et al., 2013). Este módulo permite que o usuário forneça informações através de um navegador *web*. Esse módulo é composto de dois submódulos que são o Editor de Requisitos e o Editor de Soluções.

O Editor de Requisitos permite que os clientes criem uma conta para prover informações sobre o negócio, o número de determinado tipo de componente (ex: a máquina física, o roteador e o *switch*), os tipos de *softwares* (ex: o balanceador de carga, o banco de dados, o monitor de máquina virtual, a plataforma de nuvem, o servidor *web*, o sistema de monitoramento e o sistema operacional) e os requisitos de desempenho (ex: utilização de recursos e tempo de resposta), de dependabilidade (ex: disponibilidade e *downtime*) e de custo (ex: custo dos equipamentos do sistema de TI e custo dos equipamentos redundantes) impostos pelos usuários. A Figura 6.2 apresenta as telas dos requisitos de desempenho, de dependabilidade e de custo do Editor de Requisitos do SMG4PCIP. Essas telas possibilitam o fornecimento desses requisitos.

Figura 6.2: Editor de Requisitos - Telas dos Requisitos de Desempenho, de Dependabilidade e de Custo

O Editor de Soluções permite que os usuários do sistema cliente obtenham sugestões de infraestruturas de nuvens privadas de acordo com as informações sobre os componentes dessas infraestruturas e os requisitos de desempenho, de dependabiliade e de custo. Essas sugestões são os cenários de infraestruturas de nuvens que foram escolhidos conforme os requisitos estabelecidos.

6.2 Gerador de Cenários

O Gerador de Cenários é responsável pela criação de cenários que representam aspectos de desempenho, de dependabilidade e de custo da nuvem privada. Os cenários são gerados conforme os dados fornecidos pelo Editor de Requisitos. Esse módulo é composto de dois submódulos que são o Gerador de Cenários de Desempenho e Custo e o Gerador de Cenários de Disponibilidade e Custo.

O Gerador de Cenários de Desempenho e Custo proporciona a criação de cenários de infraestruturas de nuvens com diferentes configurações de *software* por meio do Modelo para Geração de Cenários de Desempenho e Custo conforme a Seção 5.2.1. Esses cenários são gerados através da atribuição de diferentes conjuntos de *software* a várias infraestruturas de nuvens.

Os tipos de componentes do conjunto de *software* são estabelecidos pelo cliente no Editor de Requisitos. Como exemplo, os tipos de balanceadores de carga podem ser o *Network Load Balancing* (NLB) (MICROSOFT, 2014) e o *Linux Virtual Server* (LVS) (PROJECT, 2013d); os tipos de gerenciadores de banco de dados podem ser o *Oracle* (ORACLE, 2014) e o *MySQL* (MYSQL, 2013); os tipos de monitores de máquina virtual (MMV) podem ser o KVM (KVM, 2014) e o Xen (XEN, 2014); os tipos de plataformas de nuvem podem ser o *Eucalyptus* (EUCALYPTUS, 2013), o *Nimbus* (PROJECT, 2013a), o *OpenNebula* (PROJECT, 2013b) e o *OpenStack* (PROJECT, 2013c); os tipos de servidores *web* podem ser o *Apache* (APACHE, 2013) e o *Lighttpd* (LIGHTTPD, 2013); o tipo de sistema cliente pode ser o *Moodle* (MOODLE, 2013); e os tipos de sistemas operacionais podem ser o *Ubuntu* (UBUNTU, 2013) e o *CentOS* (CENTOS, 2013).

Os tipos de componentes do conjunto de *hardware* também são estabelecidos pelo cliente no Editor de Requisitos. Como exemplo, as capacidades da infraestrutura de processamento (PR) podem ser *dual-core* e *quad-core*; as capacidades da memória (MP) podem ser 2GB e 4GB; e as capacidades da memória secundária (MS) podem ser 80GB e 160GB. A Figura 6.3 apresenta a tela da escolha do *hardware* e do *software* do Editor de Requisitos do SMG4PCIP.

Como exemplo de cenário de desempenho e custo temos o conjunto de *software* formado pelo balanceador de carga (BC) - *Linux virtual server* (LVS), banco de dados (BD) - *MySQL*, monitor de máquina virtual - KVM, plataforma de nuvem (PN) - *Eucalyptus*, servidor *web* (SWE) - *Apache*, sistema cliente (SC) - *Moodle* e sistema operacional (SO) - *Ubuntu* atribuído ao conjunto de *hardware* formado por um processador - *dual-core*, memória - 2GB e memória

Figura 6.3: Editor de Requisitos - Tela da Escolha dos *Hardware* e *Software*

secundária - 80GB.

Vários cenários podem ser gerados a partir do cenário anterior por meio da modificação de uma atribuição (conjunto de *software* $\longrightarrow$ conjunto de *hardware*). Esses novos cenários podem ser gerados quando modificamos um dos componentes do conjunto de *software* do cenário anterior e o atribuímos ao mesmo conjunto de *hardware* do cenário anterior.

Um novo cenário pode ser criado através da alteração do tipo de plataforma de nuvem do conjunto de *software* do cenário anterior. Essa plataforma pode ser alterada para plataforma *Nimbus* e o novo conjunto de *software* pode ser atribuído ao conjunto de *hardware* do cenário anterior. A Figura 6.4 apresenta o cenário inicial e o novo cenário gerado após a alteração de um componente do conjunto de *software*.

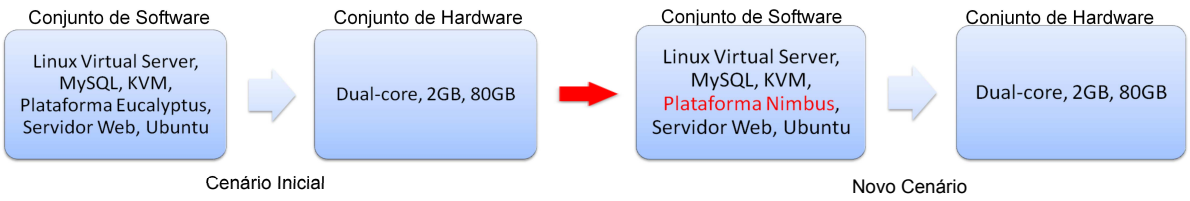


Figura 6.4: Cenário de Desempenho e Custo após Alteração no Conjunto de *Software*

Um novo cenário também pode ser criado por meio da atribuição do conjunto de *software* do cenário anterior ao conjunto de *hardware* com a capacidade de processamento alterada de *dual-core* para *quad-core*. A Figura 6.5 apresenta o cenário inicial e o novo cenário gerado após a alteração de um componente do conjunto de *hardware*.

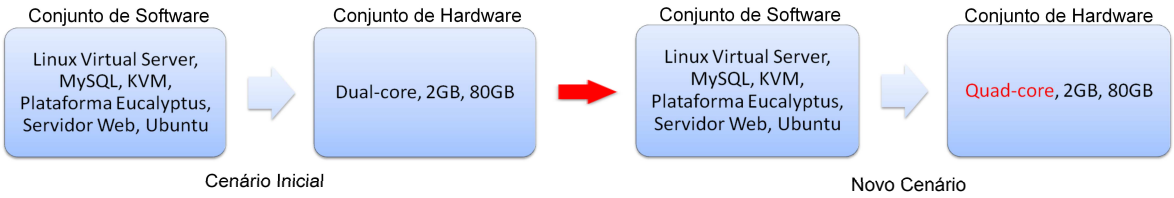


Figura 6.5: Cenário de Desempenho e Custo após Alteração no Conjunto de *Hardware*

O Gerador de Cenários de Disponibilidade e Custo proporciona a criação de cenários da nuvem privada com vários mecanismos de redundância por meio do Modelo para Geração de Cenários de Disponibilidade e Custo conforme a Seção 5.2.2. Esses cenários são gerados através da atribuição de diferentes mecanismos de redundância (RUPE, 2003) aos componentes dessa nuvem computacional.

Os tipos de componentes do conjunto de mecanismos de redundância são o ativo-ativo (AA), o *cold standby* (CS), o *hot standby* (HS), o *warm standby* (WS) e nenhum mecanismo de redundância (NMR).

O tipos de componentes do conjunto da nuvem computacional são o controlador de nuvem (CLC), o controlador de *cluster* (CC), o controlador de nó (NC), a máquina virtual (VM), o roteador (RT) e o *switch* (SW) para plataforma *Eucalyptus*; o *service node* (SN), o *node* (NO), a máquina virtual (VM), o roteador (RT) e o *switch* (SW) para plataforma *Nimbus*; o *front-end* (FE), o *cluster node* (CN), a máquina virtual (VM), o roteador (RT) e o *switch* (SW) para plataforma *OpenNebula*; e o *controller node* (CLN), o *compute node* (CPN), a máquina virtual (VM), o roteador (RT) e o *switch* (SW) para plataforma *OpenStack*.

A quantidade de determinado tipo de componente da infraestrutura da nuvem privada é estabelecida pelo cliente através do Editor de Requisitos. A Figura 6.6 apresenta a tela da escolha do número de componentes do Editor de Requisitos do SMG4PCIP.

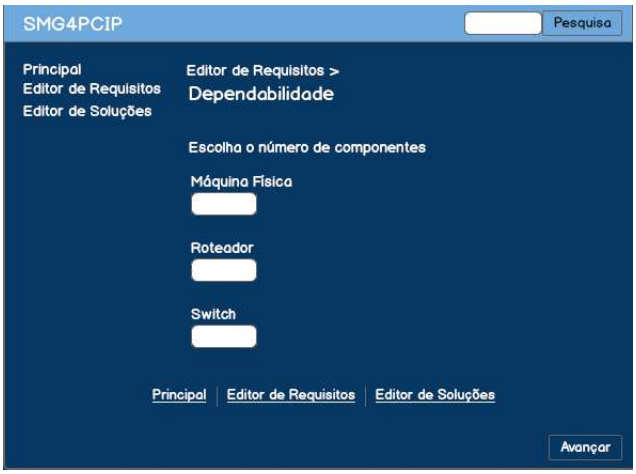


Figura 6.6: Editor de Requisitos - Tela da Escolha do Número de Componentes

Como exemplo de cenário de disponibilidade e custo temos um conjunto de mecanismos de redundância formado pelo *cold standby* (CS), nenhum mecanismo de redundância (NMR),

hot standby (HS), *hot standby* (HS), *warm standby* (WS) e ativo-ativo (AA) atribuído a um conjunto de componentes da nuvem computacional formado pelo controlador de nuvem (CLC), controlador de *cluster* (CC), controlador de nó (NC), máquina virtual (VM), roteador (RT) e *switch* (SW) para plataforma *Eucalyptus*.

Vários cenários podem ser gerados a partir do cenário anterior por meio da modificação de uma atribuição (conjunto de mecanismos de redundância $\longrightarrow$ conjunto de componentes da nuvem computacional). Esses novos cenários podem ser gerados quando modificamos um dos componentes do conjunto de mecanismos de redundância do cenários anterior e atribuímos ao mesmo conjunto de componentes da nuvem computacional.

Um novo cenário pode ser criado através da alteração do tipo de mecanismo de redundância atribuído ao roteador no cenário anterior. Esse mecanismo de redundância pode ser alterado para *cold standby*. A Figura 6.7 apresenta o cenário inicial e o novo cenário gerado após a alteração de um componente do conjunto de mecanismos de redundância.

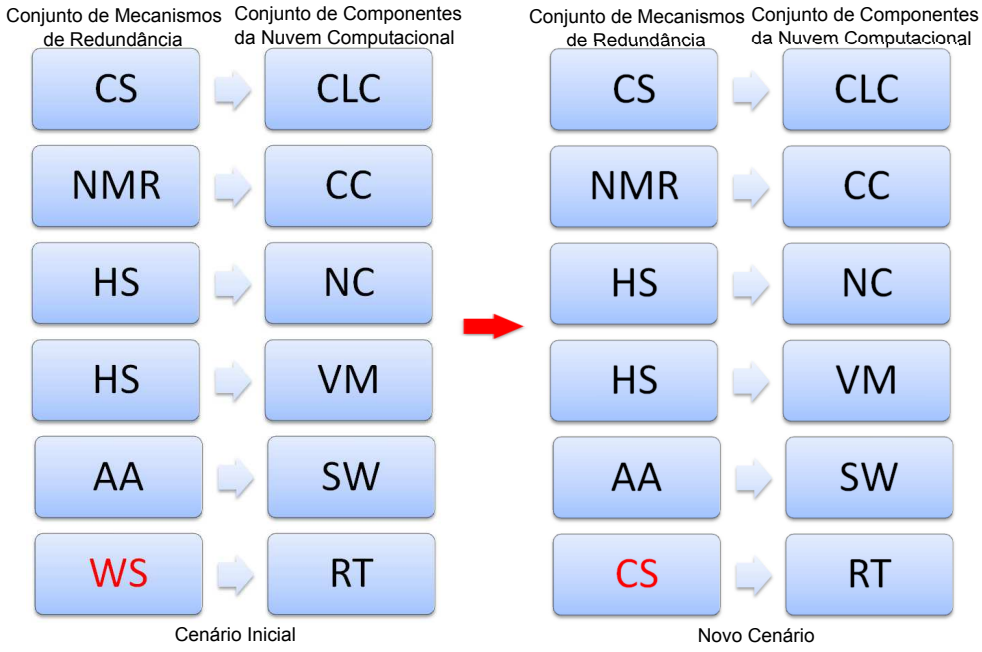


Figura 6.7: Cenário de Disponibilidade e Custo

6.3 Gerador de Modelos

O Gerador de Modelos é responsável pela geração automática de modelos para representação dos cenários criados pelo Gerador de Cenários. Esses modelos são baseados em redes de *Petri* Estocásticas (SPN) (BALBO, 2001; MARSAN et al., 1998), diagramas de blocos de confiabilidade (RBD) (KUO; ZUO, 2003) e expressões de custo.

Esse módulo é composto de três submódulos que são o Gerador de Modelo de Desempenho, o Gerador de Modelo de Disponibilidade e o Gerador de Modelo de Custo (ver Figura

6.1). Esses submódulos geram os modelos de desempenho, os modelos de disponibilidade e os modelos de custo.

Os modelos de desempenho, de disponibilidade e de custo são gerados e avaliados na ferramenta *Mercury* (SILVA et al., 2012; MERCURY, 2013). O *Mercury* (SILVA et al., 2012) é uma ferramenta do grupo de pesquisa MoDCS (Modeling of Distributed and Concurrent Systems) (MACIEL, 2015), que provê a geração e análise de modelos RBD, cadeias de *Markov* e redes de *Petri* estocásticas.

Essa ferramenta é o *kernel* de modelagem e avaliação dos modelos de alto nível criados pela ferramenta ASTRO (CALLOU et al., 2013). A ferramenta ASTRO é um ambiente integrado para a avaliação de disponibilidade, sustentabilidade e custo de ambientes de *data center*. A ferramenta ASTRO permite que usuários modelem sistemas energéticos, sistemas de resfriamento e sistemas de TI.

O Gerador de Modelo de Desempenho provê modelos (ver Seção 5.1.1) para representação dos cenários de infraestruturas de nuvens com diferentes configurações de *software* conforme a Seção 5.2.1.

Os modelos de desempenho são gerados conforme as fases de medição e de modelagem do método de Geração de Modelo de Desempenho (ver Seção 4.2) da metodologia proposta. Na fase de medição, os conjuntos de *software* são configurados na infraestrutura de nuvem. Essa infraestrutura é submetida a uma carga de trabalho e medições são realizadas para análise do impacto dessa carga de trabalho na infraestrutura de nuvem. Os resultados da medição são as médias, os desvio-padrões e os inversos dos coeficientes de variação das métricas de desempenho. Essas atividades não são realizadas pelo SMG4PCIP, mas por um projetista com experiência em ferramentas de medição.

O SMG4PCIP é responsável por algumas atividades da fase de modelagem do método de Geração de Modelo de Desempenho (ver Seção 4.2). O SMG4PCIP utiliza os valores das médias, dos desvios-padrões e dos inversos dos coeficientes de variação das métricas de desempenho para concepção do modelo de desempenho refinado. De acordo com o valor do inverso do coeficiente de variação fornecido pelo projetista, uma distribuição de probabilidade é escolhida para representar os dados medidos. A Figura 6.8 apresenta a tela das estatísticas de medição. A escolha de uma distribuição de probabilidade implica no cálculo dos seus parâmetros e no refinamento do modelo de desempenho. A Figura 6.8 também apresenta a tela dos parâmetros das distribuições. O SMG4PCIP é responsável pela escolha da distribuição de probabilidade, cálculo dos seus parâmetros e o refinamento do modelo de desempenho.

Após o refinamento do modelo de desempenho, as métricas são mapeadas neste modelo. O modelo refinado será validado através da comparação dos valores das métricas de desempenho medidos e calculados no modelo. O SMG4PCIP não é responsável pela atividade de validação do modelo e da confirmação dos novos parâmetros da distribuição de probabilidade escolhida. O projetista com experiência em modelos baseados em espaço de estados deve validar o modelo de desempenho e confirmar os parâmetros da distribuição de probabilidade. O modelo de

desempenho validado será usado para concepção de cenários de desempenho e custo.

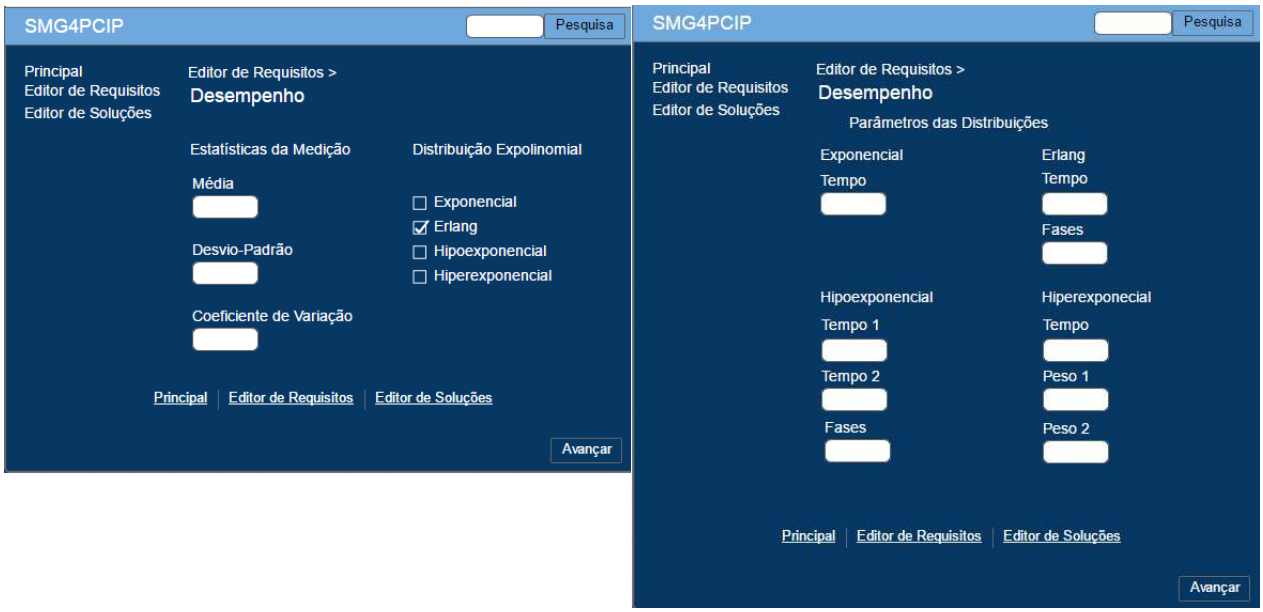


Figura 6.8: Editor de Requisitos - Tela das Estatísticas de Medição e Tela dos Parâmetros das Distribuições

O Gerador de Modelo de Disponibilidade proporciona os modelos (ver Seção 5.1.2) para representação dos cenários de infraestruturas de nuvens com a atribuição de vários mecanismos de redundância aos seus componentes conforme a Seção 5.2.2.

O SMG4PCIP provê a modelagem hierárquica e heterogênea da infraestrutura da nuvem privada conforme o método de Geração de Modelo de Disponibilidade (ver Seção 4.3) da metodologia proposta. Nessa estratégia de modelagem, os componentes da nuvem privada são modelados através de modelos RBD de baixo nível. Os modelos do sistema computacional e da máquina virtual são os modelos de baixo nível que representam os componentes dessa infraestrutura. Esses modelos são agrupados de forma a representar os subsistemas da infraestrutura da nuvem privada. Os modelos estocásticos de alto nível baseados em SPNs e RBDs representam os sistemas da infraestrutura da nuvem privada. Os modelos RBD das plataformas de nuvens são os modelos de alto nível dessas infraestruturas.

O modelo de disponibilidade será baseado em RBD quando não precisar representar dependência com os mecanismos de redundância atribuídos aos componentes da nuvem computacional e será baseado em SPN quando precisar representar essa dependência. O modelo do mecanismo de redundância *hot standby* é baseado em RBD e os modelos dos mecanismos de redundância *cold standby*, *warm standby* e o ativo-ativo são baseados em SPN.

Após a modelagem do cenário, as métricas de dependabilidade são mapeadas no modelo de disponibilidade concebido. A estratégia de modelagem proposta será validada através da comparação dos valores das métricas de dependabilidade calculados nessa estratégia e em uma estratégia já validada. O SMG4PCIP não é responsável pela atividade de validação da estratégia de modelagem. O projetista com experiência em modelos baseados em espaço de estados

e modelos combinatoriais deve validar a estratégia de modelagem de disponibilidade. Essa estratégia de modelagem validada será usada para concepção de cenários.

O Gerador de Modelo de Custo provê os modelos de custo que representam os cenários de desempenho e custo e os cenários de disponibilidade e custo conforme a Seção 5.1.3. Esses modelos de custos são baseados em expressões matemáticas e métricas mapeadas nos modelos de desempenho e de disponibilidade.

6.4 Avaliador

O Avaliador é responsável pela análise dos modelos providos pelo Gerador de Modelos. Esse módulo é composto de quatro submódulos que são o Avaliador de Desempenho, o Avaliador de Disponibilidade, o Avaliador de Performabilidade e o Avaliador de Custo (ver Figura 6.1).

Os modelos concebidos pelo Gerador de Modelos são serializados para um arquivo com formato XML. Esse documento é enviado para uma instância remota da ferramenta *Mercury*. O *Mercury* (SILVA et al., 2012) provê a análise de modelos SPN por meio de métricas com uma sintaxe semelhante à utilizada pela ferramenta *TimeNET* (TIMENET, 2013; TIMENET4.1, 2013; ZIMMERMANN; KNOKE, 2007; ZIMMERMANN et al., 2006).

O Avaliador de Desempenho analisa os modelos criados para representar os cenários de infraestruturas de nuvens com diferentes configurações de *software* (ver Seção 5.2.1) por meio da ferramenta *Mercury*. Esse submódulo permite a avaliação do efeito de diferentes níveis e tipos de carga de trabalho na infraestrutura da nuvem computacional (BALBO, 2001; MARSAN et al., 1998) por meio de métricas de desempenho (ex: utilização de processador (UP), utilização de memória (UM) e tempo de resposta (RT)).

O Avaliador de Disponibilidade analisa os modelos criados para representar os cenários de infraestruturas de nuvens com vários mecanismos de redundância atribuídos aos seus componentes (ver Seção 5.2.2) através da ferramenta *Mercury*. Esse submódulo possibilita a avaliação do impacto da atribuição de diferentes mecanismos de redundância na disponibilidade (A) e no *downtime* (D) da infraestrutura de computação em nuvem.

O Avaliador de Performabilidade combina as métricas de desempenho e as métricas de dependabilidade. Esse submódulo permite a avaliação do impacto da ocorrência de eventos de falhas e atividades de reparo no desempenho da infraestrutura da nuvem computacional por meio de métricas de performabilidade (ex: utilização de processador (UP_p), utilização de memória (UM_p) e tempo de resposta (RT_p)).

O Avaliador de Custo analisa as expressões matemáticas dos cenários gerados. Esse submódulo permite a provisão dos custos da aquisição de equipamentos do sistema de TI, da aquisição de equipamentos e *softwares* redundantes, da substituição de equipamentos, da aquisição de licenças de *software*, da equipe técnica e da equipe de manutenção.

O SMG4PCIP aguarda os resultados da simulação estacionária que está sendo realizada na instância remota da ferramenta *Mercury*. Quando os resultados das métricas de desempenho,

de dependabilidade e de custo retornam, as informações são coletadas e classificadas conforme os modelos de otimização (ver Seções 5.2.1 e 5.2.2).

Os resultados das métricas dos cenários gerados são agrupados para sugestão de infraestruturas de nuvens privadas que atendam os requisitos dos usuários. As sugestões das infraestruturas de nuvens privadas são fornecidas através do Editor de Soluções conforme mostra a Figura 6.9.

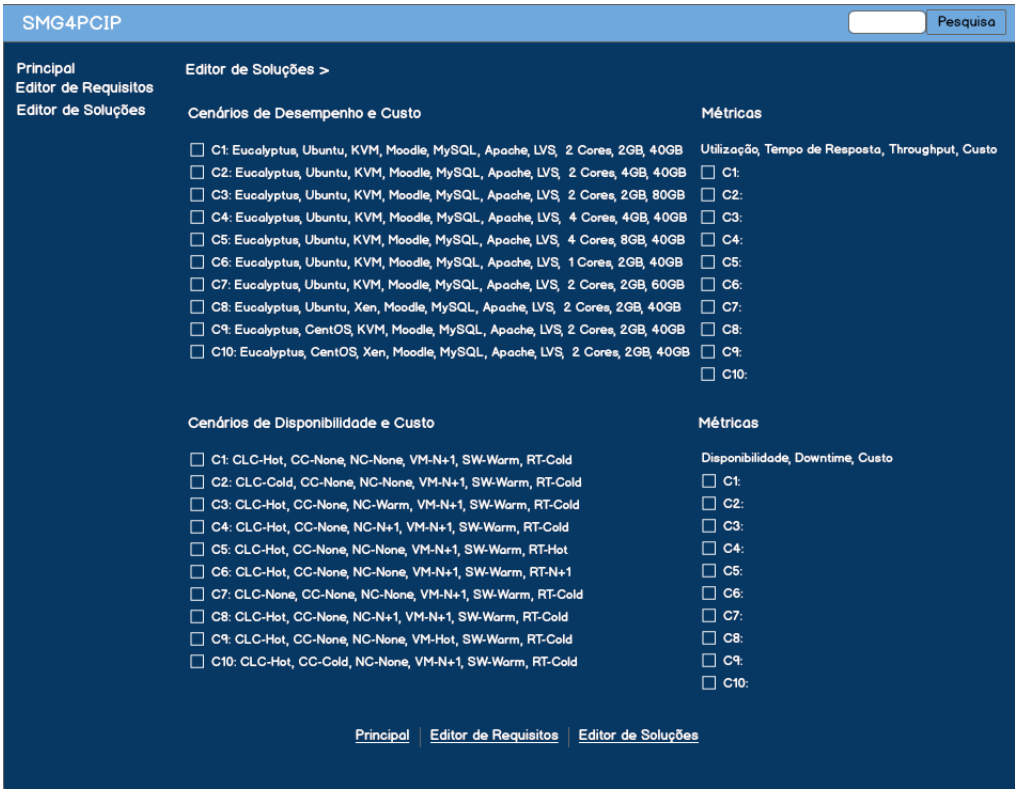


Figura 6.9: Editor de Soluções

6.5 Considerações Finais

Este capítulo apresentou a ferramenta proposta para o planejamento de infraestruturas de nuvens privadas, considerando aspectos de desempenho, dependabilidade e custo. Essa ferramenta automatiza a metodologia proposta e proporciona a geração de cenários e a geração e avaliação de modelos que representam os cenários gerados.

7

Estudo de Caso

O objetivo principal deste capítulo é avaliar se a metodologia e o ferramental propostos nesta tese podem ser aplicados para o planejamento de infraestruturas de nuvens. Baseado no fato que diversos critérios são usados para fazer esse planejamento, são apresentados três estudos de caso com contextos e objetivos diferenciados. Esses estudos de caso consideram um sistema que é composto de um ambiente virtual de aprendizagem (AVA) configurado na plataforma *Eucalyptus*. Inicialmente, este capítulo apresenta o Estudo de Caso 1 que avalia o efeito da atribuição de mecanismos de redundância aos componentes da plataforma *Eucalyptus*. Em seguida, este capítulo mostra o Estudo de Caso 2 com a análise do impacto da variação da carga de trabalho empregada nessa infraestrutura de nuvem quando há diferentes configurações de *software* e de *hardware*. Finalmente, este capítulo apresenta o Estudo de Caso 3 com a avaliação do efeito da ocorrência de defeitos e de atividades de reparo no desempenho da infraestrutura da nuvem computacional.

7.1 Introdução

O sistema adotado para avaliar se a metodologia e o ferramental propostos podem ser usados para o planejamento de infraestruturas de nuvens consiste em um ambiente virtual de aprendizagem (AVA) configurado na plataforma *Eucalyptus*. As instituições de ensino (IES) que adotam um AVA em seus cursos à distância e presenciais necessitam que a infraestrutura desse sistema atenda a uma série de requisitos. O primeiro requisito está relacionado ao tempo de resposta desse sistema. A infraestrutura do AVA deve suportar uma carga de trabalho que pode variar conforme o número de turmas e professores e o uso desse sistema pode sofrer picos durante o prazo de entrega de atividades avaliativas. Outro requisito é a minimização da ocorrência de defeitos na infraestrutura do AVA, garantido que o material didático oferecido esteja sempre disponível. A infraestrutura do AVA também deve ser planejada, de tal forma que os requisitos de custo sejam atendidos.

Esse sistema adotado consiste no AVA da UFRPE hospedado na plataforma *Eucalyptus*. Esse AVA oferece suporte *online* aos 9 cursos de nível superior, distribuídos em 38 pólos que

atendem a 2000 usuários (EADUFRPE, 2014). A plataforma adotada para o gerenciamento desses 38 pólos é o *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*) (MOODLE, 2013).

O *Moodle* é um AVA amplamente utilizado no meio acadêmico, tanto nos cursos a distância quanto nos presenciais. O *Moodle* é um sistema *open source* de gerenciamento de aprendizagem e de trabalho colaborativo em ambiente virtual que permite a criação e administração de cursos *online*, grupo de trabalho e comunidades de aprendizagem (MOODLE, 2013). Esse AVA também fornece ferramentas de avaliação específicas, como por exemplo: discussões de fórum, lições com questões, entradas de glossário, entre outros. As principais funcionalidades são: fórum, *chat*, glossário, lição, questionário e *Wiki*.

No Brasil, o *Moodle* foi homologado pelo MEC como plataforma para educação à distância, o qual pode ser adotado por quaisquer instituições que queiram aplicar essa modalidade de ensino (MOODLE, 2013). A Figura 7.1 mostra uma visão geral do *Moodle* (MOODLE, 2013).

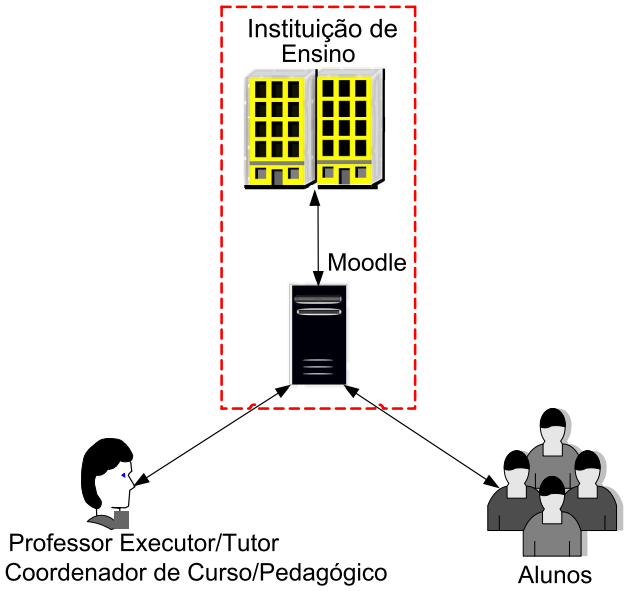


Figura 7.1: *Moodle*

O ambiente de nuvem usado nesse sistema é composto de um controlador de nuvem (CLC), um controlador de *cluster* (CC) e onze controladores de nós (NC1 - NC11). A Figura 7.2 mostra o sistema descrito.

O CLC foi configurado em um servidor, o CC foi configurado em um servidor e os NCs foram configurados em 11 servidores. O *Moodle* hospedado na plataforma *Eucalyptus* foi instalado em diferentes máquinas virtuais providas pela plataforma *Eucalyptus*. Essas máquinas virtuais foram instanciadas nos servidores onde os serviços dos NC's (NC1 - NC11) são executados. Além do EAD da UFRPE, outras instituições ensino podem empregar a configuração apresentada no sistema adotado.

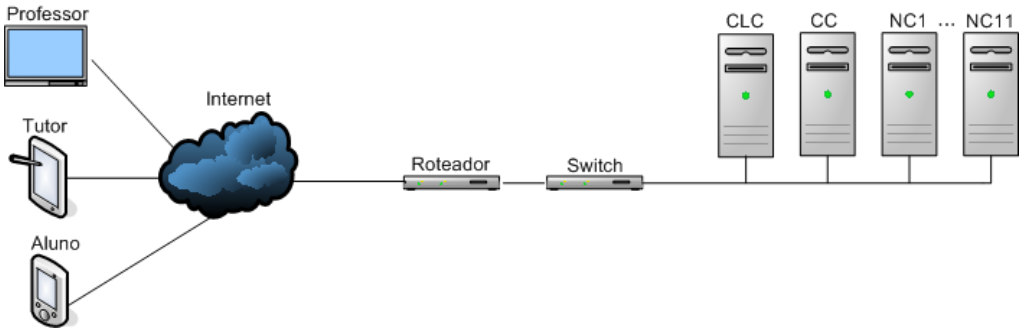


Figura 7.2: *Moodle* Hospedado na Plataforma *Eucalyptus*

7.2 Estudo de Caso 1

Esta seção apresenta um estudo cujo objetivo é avaliar se a metodologia proposta (ver Figura 4.1) pode ser adotada para o planejamento de infraestruturas de nuvens com diferentes mecanismos de redundância atribuídos aos componentes dessa infraestrutura, tendo como base o sistema descrito. Esse estudo de caso avalia o impacto da atribuição desses mecanismos de redundância aos componentes da infraestrutura de nuvem através das métricas disponibilidade, *downtime* e custo. Os resultados dessa avaliação proporcionam a sugestão de infraestruturas de nuvens que atendem aos requisitos de disponibilidade e custo. As próximas seções apresentarão todas as atividades necessárias para a geração, modelagem e avaliação das soluções de infraestruturas de nuvens.

7.2.1 Geração e Representação de Infraestruturas de Nuvens

As soluções de infraestruturas de nuvens são geradas conforme a metodologia proposta (ver Seção 4.1). Cada solução de infraestrutura de nuvem é concebida através da atribuição dos mecanismos de redundância ativo-ativo, *cold standby*, *hot standby*, *warm standby* e nenhum mecanismo de redundância aos componentes CLC, CC, NC, VM do balanceador de carga, VM do AVA, VM do banco de dados, roteador e *switch* da plataforma *Eucalyptus*.

A plataforma *Eucalyptus* foi configurada em 13 servidores, cada um destinado ao controlador de nuvem (CLC), ao controlador de *cluster* (CC) e aos controladores de nós (NC). Diferentes máquinas virtuais foram configuradas nos servidores que executam os serviços dos NC's.

As soluções de infraestruturas de nuvens concebidas são selecionadas conforme o resultado da avaliação da disponibilidade, *downtime* e custo. Neste estudo, os critérios para escolha são no máximo 10 soluções de infraestruturas de nuvens com a disponibilidade maior que 93,50%, o *downtime* menor que 560,00 Horas/Ano e a soma dos custos menor que US$ 68,000.00. Esses critérios foram inspirados nos requisitos de disponibilidade e custo do sistema adotado.

A estratégia de modelagem proposta (ver Seção 5.1.2) foi adotada para representação e

avaliação da disponibilidade e *downtime* das soluções de infraestruturas de nuvens concebidas. Nessa estratégia de modelagem, modelos RBD representam os componentes da nuvem computacional. Esses modelos são agrupados de forma a representar os subsistemas da infraestrutura da nuvem. O Modelo do Sistema Computacional (ver Figura 7.5), o Modelo da Máquina Virtual (ver Figura 7.6) e o Modelo do Gerenciador de Recursos (Ver Figura 7.7) representam esses subsistemas.

Os modelos SPNs e RBDs representam os sistemas da nuvem computacional. O Modelo da Plataforma *Eucalyptus* (ver Figura 5.18) representa esses sistemas. Quando o mecanismo de redundância *hot standby* (ver Figura 5.34) é atribuído aos componentes da plataforma *Eucalyptus* adotamos o modelo RBD, pois não há necessidade de representação da dependência entre o componente principal e o redundante. Mas quando os mecanismos de redundância ativo-ativo (ver Figura 5.28), *cold standby* (ver Figura 5.35) e *warm standby* (ver Figura 5.36) são atribuídos aos componentes da plataforma *Eucalyptus*, adotamos o modelo SPN. Esse modelo permite a representação da dependência entre o componente principal e o redundante.

Como exemplo, a Figura 7.3 apresenta o modelo RBD da plataforma *Eucalyptus* com um mecanismo de redundância *hot standby* atribuído ao componente CLC.

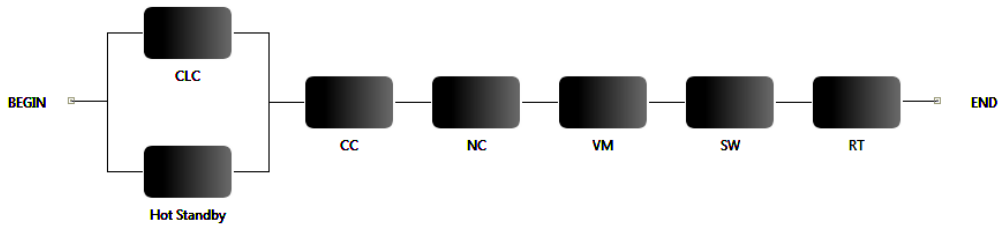


Figura 7.3: Modelo da Plataforma *Eucalyptus* com o Mecanismo de Redundância *Hot Standby*

Um outro exemplo apresenta o modelo SPN dessa plataforma com um mecanismo de redundância *cold standby* atribuído ao componente CLC (ver Figura 7.4).

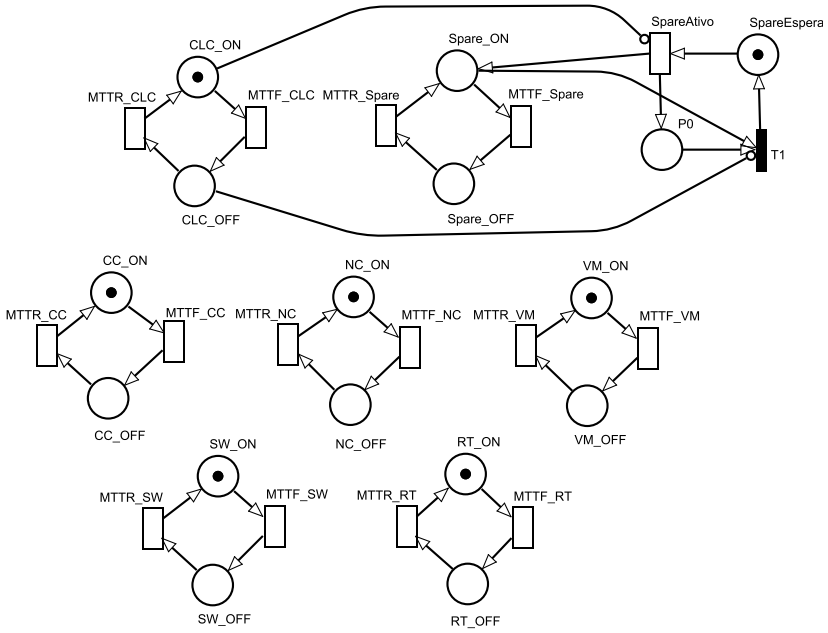


Figura 7.4: Modelo da Plataforma *Eucalyptus* com o Mecanismo de Redundância *Cold Standby*

Os recursos computacionais dos servidores da plataforma *Eucalyptus* foram representados através do Modelo do Sistema Computacional (Ver Figura 7.5). A Tabela 7.1 mostra os MTTF's e os MTTR's dos recursos computacionais do servidor que executa o serviço do CLC, CC ou NC conforme (KIM; MACHIDA; TRIVEDI, 2009; HANDY, 2014). Os recursos computacionais são a infraestrutura de processamento (PR), a memória (MP) e a memória secundária (MS). Os valores do MTTF e do MTTR do sistema computacional são 329.067,64 horas e 8,00 horas, respectivamente.

Figura 7.5: Modelo do Sistema Computacional

Tabela 7.1: Parâmetros de Dependabilidade dos Recursos do Sistema Computacional

Tipo	MTTF (horas)	MTTR (horas)
Infraestrutura de Processamento	2.500.000,00	8,00
Memória	480.000,00	8,00
Memória Secundária (Disco)	1.800.000,00	8,00

O Modelo da Máquina Virtual (Ver Figura 7.6) representa as máquinas virtuais da plataforma *Eucalyptus*. A Tabela 7.2 mostra os MTTF's e MTTR's (KIM; MACHIDA; TRIVEDI, 2009) dos *softwares* que podem ser configurados na máquina virtual. Esses *softwares* são o *Moodle 2.6.2* (SC) (MOODLE, 2013), o *Ubuntu 13.10* (SO) (UBUNTU, 2013), o *MySQL 5.5.31* (BD) (MYSQL, 2013), o *Apache 2.0* (SWE) (APACHE, 2013), o KVM (MMV) (KVM, 2014) e o LVS (BC) (PROJECT, 2013d).

Figura 7.6: Modelo da Máquina Virtual

Tabela 7.2: Parâmetros de Dependabilidade dos *Softwares* que podem ser Configurados na Máquina Virtual

MTTF (horas)	MTTR (horas)
2.880,00	8,00

A plataforma *Eucalyptus* é composta de 3 tipos de máquinas virtuais que são a máquina virtual do AVA, a máquina virtual do banco de dados e a máquina virtual do balanceador de carga. A máquina virtual do AVA é composta pelo *Moodle* (SC) (MOODLE, 2013), *Ubuntu* (SO) (UBUNTU, 2013), *Apache* (SWE) (APACHE, 2013) e KVM (MMV) (KVM, 2014). Os valores do MTTF e do MTTR dessa máquina virtual são 720,00 horas e 8,00 horas, respectivamente. A máquina virtual do banco de dados é composta pelo *Ubuntu* (SO) (UBUNTU, 2013), o *MySQL* (BD) (MYSQL, 2013) e KVM (MMV) (KVM, 2014). A máquina virtual do balanceador de carga é composta pelo *Ubuntu* (SO) (UBUNTU, 2013), KVM (MMV) (KVM, 2014) e LVS (BC) (PROJECT, 2013d). Os valores do MTTF e do MTTR da máquina virtual do banco de dados e do balanceador de carga são 960,00 horas e 8,00 horas, respectivamente.

O Modelo do Gerenciador de Recursos (Ver Figura 7.7) representa os módulos de gerenciamento da plataforma *Eucalyptus*. Esses módulos de gerenciamento são o CLC, o CC e o NC. A Tabela 7.3 mostra os MTTF's e MTTR's (KIM; MACHIDA; TRIVEDI, 2009) dos componentes dos módulos de gerenciamento da plataforma *Eucalyptus*. Esses componentes são o sistema computacional (SCL), a plataforma *Eucalyptus 3.4* (PN) (EUCALYPTUS, 2013) e o *Ubuntu* (SO) (UBUNTU, 2013). Os valores do MTTF e do MTTR dos módulos de gerenciamento são 1.434,00 horas e 8,00 horas, respectivamente.

Figura 7.7: Modelo do Gerenciador de Recursos

Tabela 7.3: Parâmetros de Dependabilidade dos Componentes do Módulo de Gerenciamento de Recursos

Tipo	MTTF (horas)	MTTR (horas)
Plataforma de Nuvem	2.880,00	8,00
Sistema Computacional	329.067,64	8,00
Sistema Operacional	2.880,00	8,00

O Modelo do Sistema Computacional, o Modelo da Máquina Virtual e o Modelo do Gerenciador de Recursos foram usados para calcular os parâmetros de dependabilidade para o Modelo da Plataforma *Eucalyptus*. Este modelo representa o controlador de nuvem (CLC), o controlador de cluster (CC), o controlador de nó (NC), a máquina virtual do AVA (VMSC), a máquina virtual do banco de dados (VMBD), a máquina virtual do balanceador de carga (VMBC), o roteador (RT) e o *switch* (SW). A Figura 7.8 mostra o modelo RBD adotado para estimar a disponibilidade e o *downtime* da plataforma *Eucalyptus* (EUCALYPTUS, 2013).

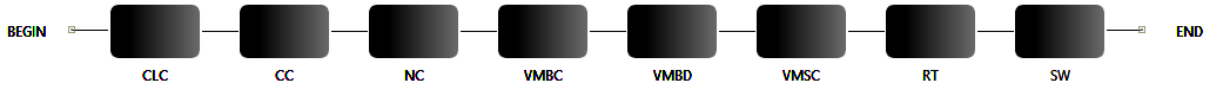


Figura 7.8: Modelo da Plataforma *Eucalyptus*

Os MTTF's e os MTTR's dos componentes da plataforma *Eucalyptus* (KIM; MACHIDA; TRIVEDI, 2009; CISCO, 2014a,b) são apresentados na Tabela 7.4. Os valores dos MTTF's do roteador e do *switch* fornecidos em (CISCO, 2014a,b) sofreram um fator de redução de 0,8 considerando um nível de confiança de 95%, conforme (SMITH, 2011).

Tabela 7.4: Parâmetros de Dependabilidade dos Componentes da Plataforma *Eucalyptus*

Tipo	MTTF (horas)	MTTR (horas)
CC/CLC/NC	1.434,00	8,00
RT	80.000,00	8,00
SW	11.200,00	8,00
VMBC/VMBD	960,00	8,00
VMSC	720,00	8,00

Os MTTF's dos mecanismos de redundância atribuídos aos componentes da plataforma *Eucalyptus* são apresentados na Tabela 7.5.

Tabela 7.5: MTTF's dos Mecanismos de Redundância

Componente	Ativo-Ativo (horas)	*Cold Standby* (horas)	*Hot Standby* (horas)	*Warm Standby* OP (horas)	*Warm Standby* NOP (horas)
CC/CLC/NC	1.434,00	1.003,80	1.434,00	1.434,00	1.003,80
RT	80.000,00	50.000,00	80.000,00	80.000,00	50.000,00
SW	11.200,00	7.840,00	11.200,00	11.200,00	7.840,00
VMBC/VMBD	960,00	672,20	960,00	960,00	672,20
VMSC	720,00	504,00	720,00	720,00	720,00

Os mecanismos de redundância ativo-ativo, *hot standby* e *warm standby* ativo estão funcionando enquanto o componente principal está operacional. Sendo assim, esse trabalho considera que podem ser usados os mesmos equipamentos para o componente principal e o componente redundante. Dessa forma, os MTTF's desses mecanismos de redundância são os mesmos dos seus componentes principais. Em contrapartida, os mecanismos de redundância *cold standby* e *warm standby* inativo não estão funcionando enquanto o componente principal está operacional. Nesse caso, esse trabalho considera que são usados equipamentos diferentes para o componente principal e o componente redundante. Logo, este trabalho adotou um fator de redução de 0,3 para os MTTF's desses mecanismos de redundância em relação aos MTTF's dos componentes principais.

Os tempos de ativação (MTA) dos componentes redundantes do Modelo *Cold Standby* (ver Figura 5.35) e do Modelo *Warm Standby* (ver Figura 5.36) são 0,08 horas e 0,16 horas, respectivamente. O tempo de erro de percepção ($MTTEP$) de um defeito é 0,08 horas, o tempo de configuração do sistema ($TNFL$) para enviar as requisições apenas para a máquina virtual

operacional é 0,08 horas e o tempo em que o sistema tenta realizar essa configuração sem sucesso (TFL) é 0,16 horas para o Modelo de Redundância Ativo-Ativo (ver Figura 5.28).

O Modelo de Custo da Equipe de Manutenção (ver Equação 5.8), o Modelo de Custo da Substituição de Equipamentos (ver Equação 5.10) e o Modelo de Custo de Aquisição de Equipamentos e *Softwares* Redundantes (ver Equação 5.6) foram adotados para avaliação do custo das infraestruturas de nuvens concebidas. Os dois primeiros modelos de custo foram mapeados no modelo SPN concebido para representação da infraestrutura de nuvem.

O Modelo de Custo da Equipe de Manutenção (ver Equação 5.7) provê o cálculo dos gastos com a equipe de manutenção. Essa equipe presta serviço ao AVA configurado na nuvem computacional e é composta por 1 técnico. O custo unitário da hora de trabalho desse técnico é US 20.00.

O Modelo de Custo da Substituição de Equipamentos (ver Equação 5.9) proporciona o cálculo dos gastos com a substituição de cada componente da nuvem computacional. A Tabela 7.6 apresenta os custos unitários para a substituição desses componentes. Esses valores são os custos unitários dos componentes da nuvem computacional com um fator de redução de 0,5.

Tabela 7.6: Parâmetros de Custo Unitário da Substituição dos Componentes da Nuvem Computacional

Componente	Custo (US$)
CC/CLC/NC	250.00
RT	1,645.00
SW	2,000.00

O Modelo de Custo de Aquisição de Equipamentos e *Softwares* Redundantes (ver Equação 5.6) foi usado para obtenção do custo das redundâncias dos equipamentos e *softwares* da infraestrutura de nuvem. Esses custos são relativos a componentes redundantes dos tipos ativo-ativo, *cold standby*, *hot standby* e *warm standby*. Esse trabalho considera que os componentes redundantes do tipo ativo-ativo, *hot standby* e *warm standby* são os mesmos dos componentes principais. Sendo assim, os custos unitários dos componentes principais e redundantes são iguais. Mas esse trabalho considera que são usados equipamentos diferentes para o componente principal e o componente redundante do tipo *cold standby*. Assim, os custos unitários desse mecanismo de redundância sofre um fator de redução de 0,3 em relação aos custos unitários do componente principal. A Tabela 7.7 apresenta os custos unitários desses componentes redundantes.

Tabela 7.7: Parâmetros de Custo dos Mecanismos de Redundância

Componente	Ativo-Ativo (US$)	*Cold Standby* (US$)	*Hot Standby* (US$)	*Warm Standby* (US$)
CC/CLC/NC	500.00	350.00	500.00	500.00
RT	3,291.46	2,304.02	3,291.46	3,291.46
SW	4,000.00	2,799.30	4,000.00	4,000.00

7.2.2 Avaliação das Infraestruturas de Nuvens

O SMG4PCIP (ver Capítulo 6) gerou automaticamente as soluções de infraestruturas de nuvens conforme o modelo de otimização apresentado na Seção 5.2.2. Essa ferramenta também gerou os modelos de disponibilidade e custo apresentados na Seção 7.2.1 para representação e avaliação dessas infraestruturas.

As soluções de infraestruturas de nuvens mostradas na Tabela 7.8 foram selecionadas pelo SMG4PCIP, pois apresentam resultados de disponibilidade, *downtime* e custo que se enquadram nos requisitos dos usuários. Esses requisitos são a disponibilidade maior que 93,50%, o *downtime* menor que 560,00 Horas/Ano e a soma dos custos menor que US$ 68,000.00.

A primeira coluna dessa tabela enumera as soluções escolhidas e a segunda coluna mostra os mecanismos de redundância atribuídos aos componentes (CLC, CC, NC, RT, SW, VMBC, VMBD, e VMSC) da plataforma *Eucalyptus*. Os tipos desses mecanismos de redundância são ativo-ativo - AA, *cold standby* - CS, *hot standby* - HS, *warm standby* - WS e nenhum mecanismo de redundância - None. As demais colunas mostram os resultados da disponibilidade (%), *downtime* (Hora/Ano) e custo (US$/Ano) das soluções escolhidas.

Tabela 7.8: Soluções de Disponibilidade e Custo Escolhidas

Soluções	Tipos de Redundâncias Atribuídas aos componentes CLC, CC, NC, RT, SW, VMBC, VMBD, e VMSC	Disponibilidade (%)	*Downtime* (Hora/Ano)	Custo (US$/Ano)
1	WS, WS, None, HS, AA, WS, AA, HS	93,6530	555,996	57,607.21
2	CS, WS, None, HS, AA, None, AA, HS	93,7466	547,795	58,234.65

3	WS, WS, None, HS, AA, AA, AA, HS	93,7471	547,756	57,279.50
4	WS, None, None, HS, AA, None, AA, HS	93,7135	550,699	67,968.31
5	WS, WS, None, HS, AA, None, None, HS	93,7028	551,631	57,566.59
6	WS, WS, None, HS, AA, None, AA, None	93,7949	543,565	56,996.02
7	WS, AA, None, HS, AA, None, AA, HS	93,7772	545,117	62,799.70
8	WS, WS, None, HS, CS, None, AA, HS	93,7967	543,406	56,857.05
9	WS, WS, None, HS, None, None, AA, HS	93,8187	541,485	53,583.90
10	WS, WS, None, CS, AA, None, AA, HS	93,7889	544,096	55,402.71

As soluções escolhidas pela ferramenta proposta são infraestruturas de nuvens com diferentes mecanismos de redundância atribuídos aos seus componentes. Essas soluções foram escolhidas, pois atendem aos requisitos de disponibilidade, *downtime* e custo. Embora os níveis de disponibilidade das soluções selecionadas estejam muito próximos, a solução 9 apresenta o maior valor de disponibilidade e os menores valores de *downtime* e custo, em relação às demais soluções. Essa solução de infraestrutura de nuvem é a mais indicada, devido aos melhores valores de disponibilidade, *downtime* e custo.

O objetivo do estudo de caso foi alcançado, pois a metodologia e ferramental propostos permitiram o planejamento de infraestruturas de nuvens com diferentes mecanismos de redundância atribuídos aos seus componentes e mostrou o impacto dessa atribuição através das métricas disponibilidade, *downtime* e custo.

7.3 Estudo de Caso 2

O estudo apresentado nesta seção tem a finalidade de planejar as infraestruturas de nuvens com diferentes configurações de *software* no sistema adotado (ver Figura 4.1). Esse estudo avalia o impacto dessas diferentes configurações de *software* no sistema adotado por meio das métricas tempo médio de resposta, utilização de recursos e custo. Os resultados dessa avaliação proporcionam sugestões de infraestruturas de nuvens que atendem aos requisitos de desempenho e custo.

As próximas seções apresentarão todas as atividades necessárias para a geração e avaliação das soluções de infraestruturas de nuvens. A Seção 7.3.1 apresentará a geração e modelagem das soluções de infraestruturas de nuvens e a Seção 7.3.2 mostrará as soluções de infraestruturas de nuvens sugeridas conforme os requisitos de dependabilidade e custo.

7.3.1 Geração e Representação de Infraestruturas de Nuvens

As soluções de infraestruturas de nuvens são geradas conforme a metodologia proposta (ver Seção 4.1). Cada solução de infraestrutura de nuvem é concebida através de duas diferentes configurações de *software* e uma configuração de *hardware* no sistema adotado. Apenas 1 conjunto de *hardware* e 2 conjuntos de *software* foram considerados na geração das soluções das infraestruturas de nuvens, entretanto novo estudos podem ser concebidos com uma maior quantidade de conjuntos de *hardware* e *software*.

As soluções de infraestruturas de nuvens concebidas são selecionadas conforme o resultado da avaliação do tempo médio de resposta, utilização de recursos e custo. Neste estudo, os critérios são no máximo 10 infraestruturas de nuvens com o tempo médio de resposta menor que 1,50 segundos, a utilização do processador e da memória menores que 95% e a soma dos custos deve ser menor que US$ 60.000,00. Esse critério de custo também foi inspirado no requisito do sistema adotado.

A Modelagem de desempenho das soluções de infraestruturas de nuvem geradas foram realizadas através da fase de medição e da fase de modelagem conforme a Seção 4.2. Na fase de medição, os experimentos foram realizados com base em três atividades que são a preparação do ambiente, a geração de carga de trabalho e a medição. A preparação do ambiente trata das duas diferentes configurações de *software* nas infraestruturas de nuvens. A geração de carga de trabalho considera o desenvolvimento do *script* de testes com a ferramenta *JMeter* (JMETER, 2013). A medição trata da execução do experimento e da coleta das métricas de desempenho.

A Preparação do Ambiente considera a configuração do conjunto de *software* 1 que é composto da plataforma *Eucalyptus 3.4* (EUCALYPTUS, 2013), do *Moodle 2.6.2* (MOODLE, 2013), do *MySQL 5.5.31* (MYSQL, 2013), do *Ubuntu 13.10* (UBUNTU, 2013) e do servidor *Apache 2.0* (APACHE, 2013) na máquina virtual da plataforma *Eucalyptus*. Essa máquina virtual é composta de um processador de dois núcleos, uma memória de 2GB e uma memória

secundária de 80GB. Essa atividade também considera a configuração do conjunto de *software* 2 que difere do conjunto de *software* 1 em relação ao servidor *web*, pois o servidor *Lighttpd 1.4.35* (LIGHTTPD, 2013) foi configurado na máquina virtual.

Na Geração de Carga de Trabalho, um *script* de testes foi desenvolvido para simular de 6 até 10 usuários do *Moodle*. Os usuários foram matriculados na disciplina de Análise de desempenho do curso de Bacharelado em Ciência da Computação, seguindo o padrão aluno6 até aluno10 e senha 12345678. Esse *script* de testes tem como finalidade simular os usuários interagindo com a atividade *chat* do *Moodle*. Os usuários solicitavam acesso ao *chat* criado no *Moodle* a cada 1 segundo. A Figura 7.9 apresenta o fluxograma do *script* de testes.

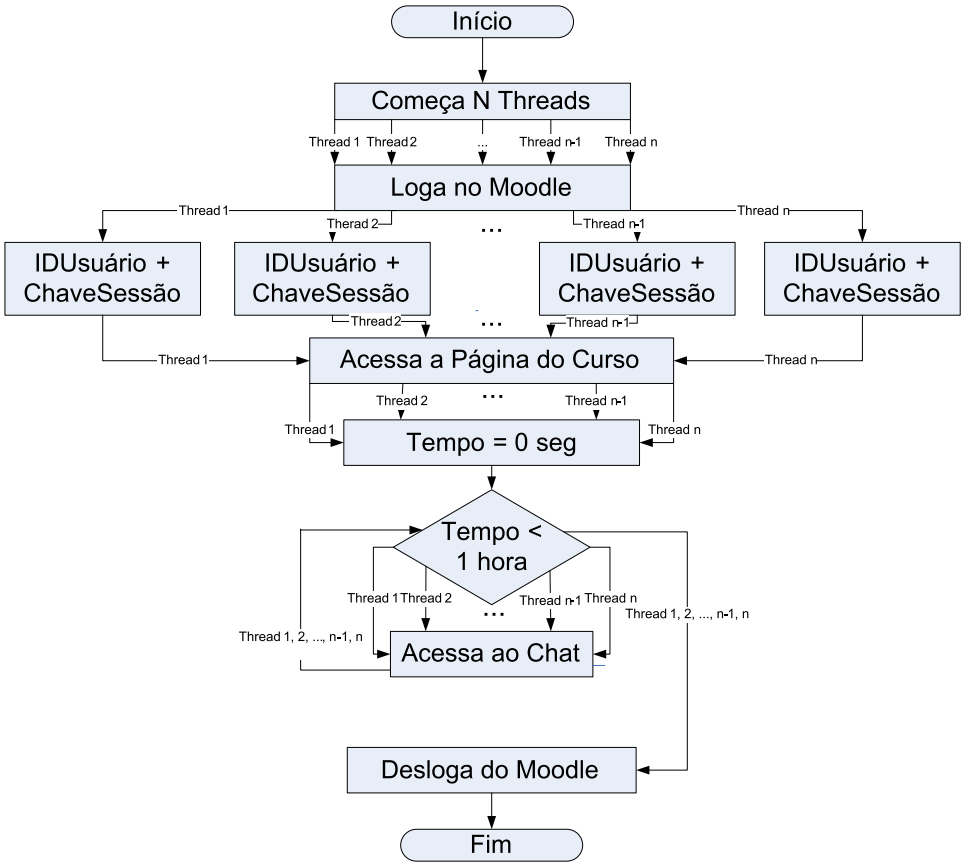


Figura 7.9: Fluxograma do *Script* de Testes

O *script* de testes é composto de 6 passos que são o Começa N *Threads*, *IDUsuário* + ChaveSessão, Loga no *Moodle*, Acessa a Página do Curso, Acessa o *Chat* e Desloga do *Moodle*. O passo Começa N *Threads* trata do início de N *threads* (N usuários). Assim, foram realizados 5 experimentos para cada conjunto de *software* atribuído à infraestrutura de nuvem. Esses experimentos consideraram o início de 6, 7, 8, 9 ou 10 usuários.

No passo *IDUsuário* + ChaveSessão, há a geração de um número (i) que varia entre 1 até n. Esse número é incrementado a cada *thread* iniciada. Para cada *thread* são criadas uma *string* aluno(i) e a senha 12345678. A tupla *username* + *password* são necessárias para o *login* dos usuários. Esse passo também trata do armazenamento do nome do usuário, ID do usuário e chave

de sessão para cada *thread*. No passo Loga no *Moodle*, cada thread faz uma requisição HTTP à página de *login* do *Moodle* contendo o nome do usuário e a senha e autentica-se no *Moodle*.

No passo Acessa a Página do Curso, cada *thread* tem acesso à página do curso de Bacharelado em Ciência da Computação. Logo após, cada thread entra em um *loop* (contador temporal) e acessa ao *chat* durante 1 hora. O passo Acessa o *Chat* trata do acesso do usuário ao *chat* e a escrita de uma mensagem que equivale a uma *string*. Esse passo é realizado até que o tempo de permanência no *loop* seja igual a 1 hora. A contagem desse tempo ocorre de forma independente para cada *thread*. Ao fim da 1 hora, cada usuário faz o *logoff* do *Moodle*, no passo Desloga do *Moodle*.

O número máximo de usuários gerados neste *script* foi 10 devido a infraestrutura de processamento e armazenamento da nuvem computacional. O *script* pode ser adaptado para gerar mais números de usuários.

A Medição consiste na execução do *script* de testes para os experimentos com vários usuários. Em cada experimento, são coletados os tempos médios de resposta para as requisições dos usuários por meio do *JMeter*. A utilização do processador e a utilização da memória da máquina virtual com a configuração dos Conjunto de *software* 1 e 2 são coletadas através de *scripts* de medição que foram criados com as ferramentas *mpstat* e *vmstat* do pacote *sysstat* (GODARD, 2013). Ao iniciar o *script* de medição, as ferramentas *mpstat* e *vmstat* são executadas durante um *loop* (contador temporal) que tem duração de 1 hora. Ao fim desse período, um arquivo de armazenamento em formato texto é gerado com as métricas de desempenho coletadas através da execução das ferramentas *mpstat* e *vmstat*. A Figura 7.10 apresenta o fluxograma do *script* de medição.

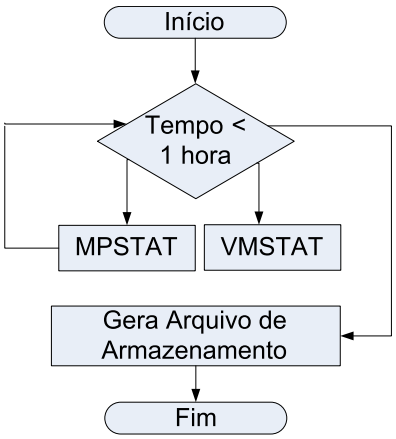


Figura 7.10: Fluxograma do *Script* de Medição

Após a execução de cada experimento, a máquina virtual foi reiniciada. Esse procedimento libera os recursos computacionais alocados no teste atual, evitando que esse teste influencie no próximo teste.

A Tabela 7.9 apresenta os tempos médios de resposta (TR), as utilizações do processador (UP) e as utilizações da memória (UM) medidas das requisições de 6 até 10 usuários. Essas

requisições foram submetidas à infraestrutura de nuvem com a configuração dos Conjuntos de *Software* 1 - CS1 e 2 - CS2.

Tabela 7.9: Resultado da Medição de Desempenho

Número de Usuários	TR (s) - CS1	TR (s) - CS2	UP (%) - CS1	UP (%) - CS2	UM (%) - CS1	UM (%) - CS2
6	0,6189	1,2411	92,3718	79,3537	80,7475	80,6797
7	0,7097	1,4748	95,0311	80,0457	83,5761	83,2397
8	0,8110	1,7112	97,1305	80,3726	84,2699	86,2422
9	0,9127	1,9762	98,4991	80,1633	86,4405	89,3239
10	1,0370	2,2250	99,0142	89,6872	80,3260	91,4094

Na fase de modelagem, a representação das infraestruturas de nuvens foi realizada com base no Refinamento do Modelo de Desempenho. O Modelo de Desempenho (ver Figura 5.4) foi adotado para representação e avaliação do tempo médio de resposta e utilização de recursos das infraestruturas de nuvens concebidas.

O Refinamento do Modelo de Desempenho foi baseado no tempo de demanda do processador e na quantidade de memória utilizada para as requisições de 6 até 10 usuários à atividade *chat* do *Moodle*. O tempo de demanda do processador é calculado através da lei de demanda de serviço (BAKOUCH, 2011; MENASCÉ; ALMEIDA, 2005; TRIVEDI, 2008) conforme a Equação 7.1. Onde TD representa o tempo de demanda do processador, UP indica a utilização do processador e TH representa o *throughput*.

$$TD = \frac{UP}{TH} \tag{7.1}$$

A Tabela 7.10 apresenta os tempos de demanda do processador (TD) e as memórias (QM) medidos das requisições de 6 até 10 usuários. Essas requisições foram submetidas a infraestrutura de nuvem com a configuração dos Conjuntos de *Softwares* 1 - CS1 e 2 - CS2.

Tabela 7.10: Resultados dos Tempos de Demanda do Processador e das Quantidades de Memória

Número de Usuários	TD (s) - CS1	TD (s) - CS2	QM (MB) - CS1	QM (MB) - CS2
6	0,2003	0,4201	1653,7080	1652,3208

7	0,2357	0,4864	1711,6395	1704,7480
8	0,2778	0,5555	1725,8466	1766,2410
9	0,3178	0,6343	1770,3006	1829,3545
10	0,3680	0,7023	1836,7940	1872,0635

Esses resultados foram analisados para escolha da distribuição expolinomial mais adequada para representar os tempos entre envios de requisições (TER) e os tempos de demanda do processador (TDP). As médias (μ_D) e os desvios-padrão (σ_D) desses tempos foram analisados para o cálculo do inverso do coeficiente de variação $1/CV = \mu_D/\sigma_D$ conforme a Seção 4.2. A Tabela 7.11 mostra os resultados das médias (μ_D), dos desvios-padrão (σ_D) e das distribuições expolinomiais escolhidas.

Tabela 7.11: Média, Desvio-Padrão e Distribuição Expolinomial

Conjuntos de *Software*	Métricas	μ_D (s)	σ_D (s)	Distribuição de Probabilidade
CS1	TER	0,1426	0,0083	Hipoexponencial
CS1	TDP	0,2799	0,0661	Hipoexponencial
CS2	TER	0,2548	0,0023	Hipoexponencial
CS2	TDP	0,5597	0,1127	Hipoexponencial

Após a definição da distribuição expolinomial mais adequada aos tempos entre envios de requisições (TER) e aos tempos de demanda do processador (TDP), deve-se calcular os parâmetros dessa distribuição. Como a distribuição hipoexponencial foi a escolhida, os parâmetros μ_1, μ_2 e γ foram calculados para as diferentes configurações de *software* da nuvem computacional. A Tabela 7.12 apresenta os valores μ_1, μ_2 e γ para o modelo refinado de desempenho da nuvem computacional.

Tabela 7.12: Parâmetros da Distribuição de Probabilidade

Conjuntos de *Software*	Métricas	μ_1 (s)	μ_2 (s)	γ
CS1	TER	0,00014	0,14	3

CS1	TDP	0,0250	0,25	17
CS2	TER	0,000035	0,136	1
CS2	TDP	0,0020	0,02	24

O modelo refinado de desempenho foi usado para obtenção das métricas tempo médio de resposta, utilização de processador e utilização de memória considerando as requisições de 6 até 10 usuários ao *chat* na nuvem computacional com diferentes configurações de *software*. As Figuras 7.11 e 7.12 apresentam as utilizações do processador e as utilizações da memória medidas e obtidas no modelo refinado de desempenho, considerando a configuração dos Conjuntos de *software* 1 e 2. O teste t emparelhado foi aplicado as utilizações do processador e memória medidas e obtidas do modelo de desempenho. Considerando um nível de significância de 5%, o teste t emparelhado gerou um intervalo de confiança de (-4,414, 0,508) para utilização do processador e um intervalo de confiança de (-4,19, 11,96) para utilização da memória. Como os intervalos de confiança contêm 0, não há evidências estatísticas para rejeitar a hipótese de equivalência entre as utilizações do processador e memória medidas e obtidas do modelo de desempenho (BAKOUCH, 2011).

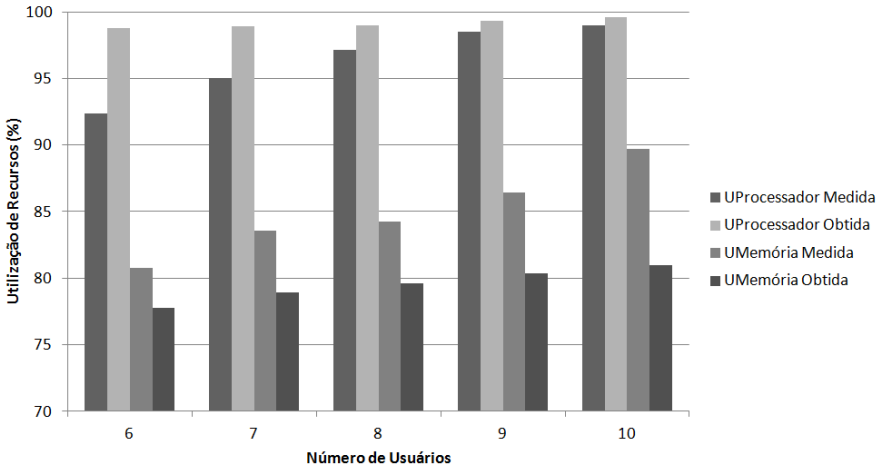


Figura 7.11: Utilização de Recursos Medida e Obtida no Modelo de Desempenho - Conjuntos de *software* 1

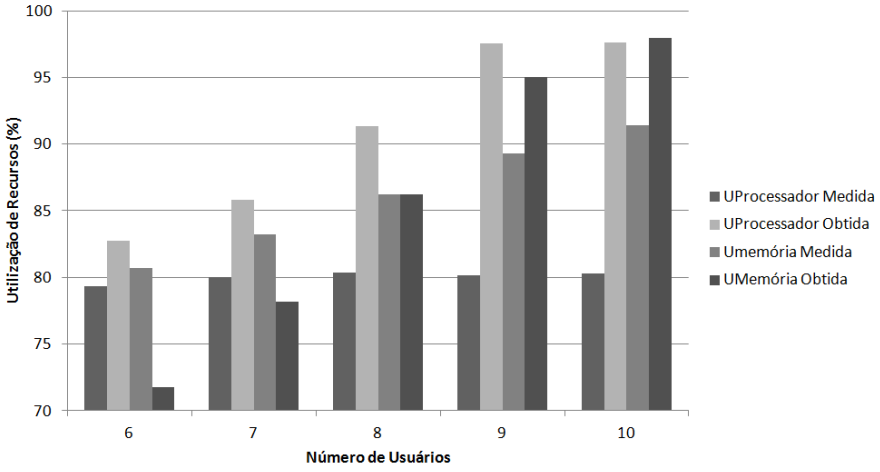


Figura 7.12: Utilização de Recursos Medida e Obtida no Modelo de Desempenho - Conjuntos de *software* 2

A Figura 7.13 apresenta os tempos médios de resposta medidos e obtidos no modelo refinado de desempenho considerando as diferentes configurações de *software*. O teste t emparelhado foi aplicado aos tempos médios de resposta medidos e obtidos do modelo de desempenho. Considerando um nível de significância de 5%, o teste t emparelhado gerou um intervalo de confiança de (-108,0, 94,3). Como o intervalo de confiança contém 0, não há evidências estatísticas para rejeitar a hipótese de equivalência entre os tempos médios de resposta medidos e obtidos do modelo de desempenho (BAKOUCH, 2011).

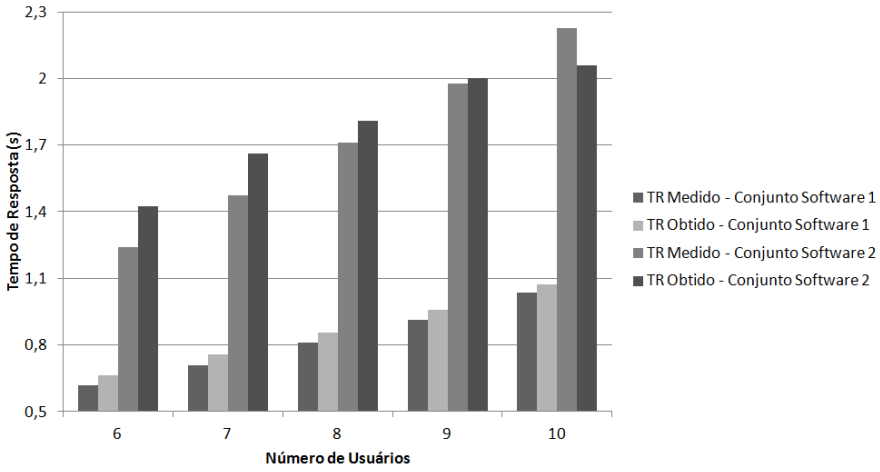


Figura 7.13: Tempos de Resposta - Conjuntos de *software* 1 e 2

Após a Validação do Modelo de Desempenho, esse modelo pode ser usado para estimar diferentes níveis de carga de trabalho no sistema adotado.

O Modelo de Custo de Aquisição de Equipamentos do Sistema de TI (ver Equação 5.3), o Modelo de Custo de Aquisição de Licenças de *Software* (ver Equação 5.4) e o Modelo de Custo da Equipe Técnica (ver Equação 5.5) foram usados para representação e avaliação das infraestruturas de nuvens concebidas. A Tabela 7.13 apresenta os custos unitários e os

custos totais da aquisição de equipamentos do sistema de TI e da aquisição de licenças de *software*. O custo unitário do *Apache 2.0* (APACHE, 2013), *Lighttpd 1.4.35* (LIGHTTPD, 2013), *Moodle 2.6.2* (MOODLE, 2013) e *Ubuntu 13.10* é $(US\$)$ 0.00. O custo total da aquisição de equipamentos do sistema de TI e da aquisição de licenças de *software* é $(US\$)$ 17,290.00.

Tabela 7.13: Parâmetros de Custo de Equipamentos e de *Software*

Componente	Custo Unitário (US$)	Quantidade	Custo Total (US$)
Banco de Dados	2,000.00	1	2,000.00
Máquina Física	500.00	13	6,500.00
Plataforma de Nuvem	1,500.00	1	1,500.00
Roteador	3,291.00	1	3,291.00
Switch	3,999.00	1	3,999.00

O custo da equipe técnica que é composta de um especialista em AVA e 1 especialista na plataforma *Eucalyptus* foi calculado por meio do Modelo de Custo da Equipe Técnica (ver Equação 5.5), considerando o custo unitário do dia de trabalho de $US\$$ 50.00. O custo anual total da equipe técnica é $US\$$ 36,500.00.

7.3.2 Avaliação de Infraestruturas de Nuvens

O SMG4PCIP (ver Capítulo 6) gerou automaticamente as infraestruturas de nuvens conforme o modelo de otimização apresentado na Seção 5.2.1. Essa ferramenta também gerou os modelos de desempenho e custo apresentados na Seção 7.3.1 para representação e avaliação dessas infraestruturas.

As soluções mostradas na Tabela 7.14 foram selecionadas pela ferramenta proposta, pois apresentam resultados de tempo médio de resposta, utilização do processador e utilização da memória que se enquadram nos requisitos dos usuários. Esses requisitos são o tempo médio de resposta menor que 1,50 segundos e a utilização do processador e da memória menores que 95%. A primeira coluna dessa tabela enumera as soluções escolhidas e a segunda coluna mostra as configurações de *hardware* e de *software*. As demais colunas mostram os resultados do tempo médio de resposta (s) e utilização de recursos (%).

Tabela 7.14: Soluções de Desempenho e Custo Escolhidas

Soluções	Configuração de *Hardware* e *Software*	Tempo Médio de Resposta (s)	Utilização de Processador (%)	Utilização de Memória (%)
1	CS1, 4 *cores*, 4 GB	1,05	92,72	77,52
2	CS1, 4 *cores*, 2 GB	1,23	94,86	93,27
3	CS1, 8 *cores*, 2 GB	0,85	82,57	91,46
4	CS1, 8 *cores*, 4 GB	0,72	80,52	76,91
5	CS1, 4 *cores*, 8 GB	0,93	90,37	70,00
6	CS1, 8 *cores*, 8 GB	0,62	78,56	66,60
7	CS2, 8 *cores*, 2 GB	1,45	78,15	81,30
8	CS2, 8 *cores*, 4 GB	1,36	76,66	71,60
9	CS2, 4 *cores*, 8 GB	1,68	84,71	61,51
10	CS2, 8 *cores*, 8 GB	1,12	65,61	59,31

O projetista pode adotar uma dessas 10 soluções de infraestruturas de nuvens com as configurações dos Conjuntos de *software* 1 e 2, pois estas atendem aos requisitos de desempenho e custo. Essas 10 soluções apresentam o mesmo custo total de *US*$ 53,790.00, mas a solução 6 tem o menor tempo médio de resposta e utilizações de processador e memória e por isso é a mais indicada para ser escolhida.

O objetivo desse estudo foi alcançado pois a metodologia proposta e o SMG4PCIP permite a geração, modelagem e avaliação de soluções de infraestruturas de nuvens com diferentes configurações de *software*. Além disso, esse estudo avaliou também o efeito das diferentes configurações de *hardware* e *software* no tempo médio de resposta, utilização de recursos e custo das infraestruturas de nuvens.

Embora esse estudo de caso tenha adotado 1 conjunto de *hardware* e 2 conjuntos de *software*, mais cenários poderiam ser gerados se houvessem mais conjuntos de *hardware* ou conjuntos de *software*.

7.4 Estudo de Caso 3

Esta seção apresenta um estudo que tem por objetivo o planejamento de infraestruturas de nuvens, considerando o impacto da ocorrência de defeitos e atividades de reparo (ver Figura 4.1). Esse estudo avalia o impacto da ocorrência desses defeitos e reparos na infraestrutura de nuvem através das métricas tempo médio de resposta, utilização de recursos e custo. Os resultados dessa

avaliação proporcionam a sugestão de infraestruturas de nuvens que atendem aos requisitos de desempenho e custo. As próximas seções apresentarão as atividades necessárias para geração e avaliação das soluções de infraestruturas de nuvens.

7.4.1 Geração e Representação de Infraestruturas de Nuvens

As soluções de infraestruturas de nuvens são geradas conforme a metodologia proposta (ver Seção 4.1). A geração das soluções das infraestruturas de nuvens ocorreu através da combinação dos modelos do Estudo de Caso 1 e os parâmetros de dependabilidade e custo aos modelos do Estudo de Caso 2 e os parâmetros de desempenho e custo. Cada solução de infraestrutura de nuvem é concebida através da combinação do resultado da estratégia de modelagem de disponibilidade e modelos de custos apresentados na Seção 7.2.1, aos resultados do modelo de desempenho e custo apresentados na Seção 7.3.1.

As soluções de infraestruturas de nuvens concebidas são selecionadas conforme o resultado da avaliação do tempo médio de resposta, utilização do processador, utilização da memória e custo dessas infraestruturas. Neste estudo, os critérios são no máximo 10 infraestruturas de nuvens com o tempo médio de resposta menor que 1,50 segundos, a utilização do processador e da memória menores que 95% e a soma dos custos deve ser menor que US$ 120,000.00. Esse critério de custo também foi inspirado no requisito de custo total do sistema adotado.

7.4.2 Avaliação de Infraestruturas de Nuvens

O SMG4PCIP (ver Capítulo 6) combina o resultado da avaliação de uma solução de infraestrutura de nuvem com vários mecanismos de redundância atribuídos aos seus componentes e o resultado das soluções de infraestruturas de nuvens com diferentes configurações de *software* e *hardware*. Essa composição provê as métricas de performabilidade, as quais são calculadas, independentemente, a partir dos modelos de desempenho e custo e dos modelos de disponibilidade e custo e posteriormente combinadas para mostrar o efeito da disponibilidade no desempenho da nuvem computacional.

Os resultados dessa combinação são apresentados na Tabela 7.15. A primeira coluna dessa tabela enumera as soluções escolhidas e a segunda coluna mostra as configurações de *hardware* e de *software*. A segunda coluna da tabela mostra os mecanismos de redundância dos componentes (CLC, CC, NC, RT, SW, VMBC, VMBD, e VMSC) da plataforma *Eucalyptus*. Os tipos desses mecanismos de redundância são ativo-ativo - AA, *cold standby* - CS, *hot standby* - HS, *warm standby* - WS e nenhum mecanismo de redundância - None. A demais colunas apresentam os resultados do tempo médio de resposta, das utilizações de recursos e do custo dos cenários das infraestruturas de nuvem.

Tabela 7.15: Soluções de Performabilidade e Custo Escolhidas

Soluções	Configuração de *Hardware* e *Software*	Tipos de Redundâncias Atribuídas aos componentes CLC, CC, NC, RT, SW, VMBC, VMBD, e VMSC	Tempo Médio de Resposta (s)	Utilização de Processador (%)	Utilização de Memória (%)	Custo (US$/Ano)
1	CS1, 4 *cores*, 4 GB	WS, None, None, HS, AA, None, AA, HS	0,98	86,83	72,60	111,397.00
2	CS1, 4 *cores*, 2 GB	CS, WS, None, HS, AA, None, AA, HS	1,15	88,93	87,44	112,024.65
3	CS1, 8 *cores*, 2 GB	WS, WS, None, HS, AA, None, None, HS	0,80	77,37	85,70	111,356.59
4	CS1, 8 *cores*, 4 GB	WS, WS, None, HS, AA, None, None, HS	0,67	75,45	72,07	111,356.59
5	CS1, 4 *cores*, 8 GB	WS, WS, None, HS, AA, None, AA, None	0,87	84,76	65,66	110,786.02

6	CS1, 8 *cores*, 8 GB	WS, WS, None, HS, AA, None, None, HS	0,58	73,61	62,40	111,356.59
7	CS2, 8 *cores*, 2 GB	CS, WS, None, HS, AA, None, AA, HS	1,36	73,65	62.44	112,024.65
8	CS2, 4 *cores*, 8 GB	WS, WS, None, HS, AA, WS, AA, HS	1,57	79,33	57,61	111,397.21
9	CS2, 4 *cores*, 8 GB	WS, WS, None, HS, None, None, AA, HS	1,58	79,47	57,71	107.373.90
10	CS2, 8 *cores*, 8 GB	WS, None, None, HS, AA, None, AA, HS	1,04	61,45	55,55	111,397.00

A ferramenta proposta sugeriu soluções de infraestruturas de nuvens com diferentes configurações de *software* e *hardware* para o sistema adotado. Além disso, cada componente dessa infraestrutura tem a atribuição de um mecanismos de redundância. O projetista pode adotar uma dessas soluções visto que elas atendem aos requisitos de desempenho e custo. Os resultados das métricas das soluções de infraestruturas de nuvens com várias configurações de *software* e *hardware* sofrem alterações devido à atribuição dos mecanismos de redundância aos seus componentes. O custo de cada solução obtida é o somatório dos custos das soluções de infraestruturas de nuvens combinadas. A solução 8 tem o menor custo, tempo médio de resposta e utilizações de processador e memória, e por isso é a mais indicada para ser escolhida.

Esse estudo de caso alcançou seu objetivo, pois apresentou soluções de infraestruturas de nuvens, considerando a ocorrência de defeitos e atividades de reparo. As soluções apresentadas são o resultado da combinação das soluções de infraestruturas de nuvens com vários mecanismos

de redundância e das soluções de infraestruturas de nuvens com diferentes configurações de *software*.

7.5 Considerações Finais

Este capítulo apresentou três estudos de caso com o objetivo de planejar infraestruturas de nuvens conforme a metodologia e o ferramental propostos. O primeiro estudo proporcionou o planejamento de infraestruturas de nuvens com vários mecanismos de redundância atribuídos aos seus componentes através das métricas disponibilidade, *downtime* e custo. No Estudo de caso 2, infraestruturas de nuvens com diferentes configurações de *software* foram concebidas para atender aos requisitos de desempenho e custo. O Estudo de caso 3 proporciona a seleção de infraestruturas de nuvens considerando o impacto de defeitos e atividades de reparo no desempenho da infraestrutura de nuvem.

8

Conclusões e Trabalhos Futuros

A computação em nuvem está se tornando cada vez mais popular na medida em que permite o provimento de computação como serviço através da *Internet*. As maiores empresas de TI estão desenvolvendo seus *data centers* nos cinco continentes para prover diferentes serviços na nuvem. O valor total esperado para a receita global dos serviços na nuvem é de 241 bilhões até o final de 2020 (RIED et al., 2014). O rápido desenvolvimento de serviços na nuvem está motivando mais indústrias a utilizarem esses serviços (AREAN, 2013). Como exemplo, cerca de 61% das empresas do Reino Unido estão utilizando pelo menos um serviço na nuvem (FORUM, 2014).

A computação em nuvem apresenta alguns desafios que precisam ser vencidos, tais como o planejamento de infraestruturas de nuvens que possibilitem a manutenção dos tempos de resposta quando ocorrerem variações na carga de trabalho e o planejamento de infraestruturas que conservem a disponibilidade quando ocorrerem eventos de falhas e atividades de reparo. O planejamento de infraestruturas de nuvens que atendam aos aspectos de desempenho, de dependabilidade e de custo é uma atividade essencial, pois garante a continuidade do negócio e a satisfação do cliente.

A maioria dos trabalhos relacionados ao planejamento de infraestruturas de nuvens privadas trata de apenas um dos aspectos citados. Esses aspectos são avaliados com o auxílio das técnicas de medição, de modelagem ou de simulação. Essa tese propôs uma solução integrada para o planejamento de infraestruturas de nuvens privadas, considerando os aspectos de desempenho, de dependabilidade e de custo. Esse trabalho representa as infraestruturas de nuvens através de redes de *Petri* estocásticas, diagramas de bloco de confiabilidade e equações de custo.

Este trabalho apresentou um estudo de caso para avaliação da solução integrada proposta. Esse estudo de caso foi baseado na configuração de um ambiente virtual de aprendizagem em uma nuvem privada. Nesse estudo de caso, a solução integrada proporcionou a geração, a avaliação e a seleção de infraestruturas de nuvens privadas, considerando aspectos de desempenho, de dependabilidade e de custo através do Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas.

As próximas seções descrevem as principais contribuições dessa tese e propõe possíveis extensões como trabalhos futuros.

8.1 Contribuições

A principal contribuição desta tese foi a solução integrada composta de uma metodologia, métodos, modelos de representação, modelos de otimização e uma ferramenta para o planejamento de infraestruturas de nuvens privadas, considerando aspectos de desempenho, de dependabilidade e de custo. As demais contribuições da tese foram detalhadas abaixo:

• **Metodologia**: a metodologia proposta proporciona a geração de cenários de infraestruturas de nuvens através de modelos de otimização. Essa metodologia também permite a representação e avaliação dos cenários de infraestruturas de nuvens por meio de modelos de representação.

• **Modelos de Otimização**: este trabalho concebeu modelos que são baseados na metaheurística GRASP. Esses modelos permitem a geração de cenários de infraestruturas de nuvens através da atribuição de conjuntos de *hardware* e *software* e a geração de cenários de infraestruturas de nuvens através da atribuição de vários mecanismos de redundância aos seus componentes.

• **Modelos de Representação**: esta tese propôs modelos de representação que expressam os aspectos de desempenho, dependabilidade e custo dos cenários de infraestruturas de nuvens privadas gerados pelos modelos de otimização.

Os modelos de desempenho representam os clientes enviando vários níveis de carga de trabalho e a infraestrutura de processamento e de armazenamento da nuvem privada. Os modelos de disponibilidade representam infraestruturas de nuvens privadas configuradas com as plataformas de nuvens. Esses modelos representam também o sistema computacional, a máquina virtual e os módulos de gerenciamento das plataformas de nuvem. Além disso, os modelos de disponibilidade representam os mecanismos de redundância ativo-ativo, *cold standby*, *hot standby* e *warm standby*. Os modelos de custo representam os custos com a aquisição dos equipamentos do sistema de TI, a aquisição de equipamentos e *softwares* redundantes, a aquisição de licenças de *software*, a equipe técnica, a equipe de manutenção e a substituição de equipamentos.

• **Ferramenta**: este trabalho apresentou o Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas que é uma ferramenta utilizada para implementar a metodologia proposta para o planejamento de infraestruturas de nuvens privadas.

• **Planejamento de Infraestruturas de Nuvens Privadas conforme os Requisitos de Desempenho**: este trabalho proporciona o planejamento de infraestruturas de nuvens por meio

da avaliação do impacto da atribuição de diferentes configurações de *software* e de *hardware*. Esse trabalho também possibilita o planejamento de nuvens privadas que atendam a determinados níveis de carga de trabalho, mantendo os valores do tempo de resposta e da utilização de recursos aceitáveis.

• **Planejamento de Infraestruturas de Nuvens Privadas conforme os Requisitos de Dependabilidade**: esta tese proporciona o planejamento de nuvens privadas por meio da avaliação do efeito da atribuição de mecanismos de redundância aos seus componentes. Essa avaliação foi realizada através da disponibilidade e *downtime* da infraestruturas de nuvens avaliadas.

• **Planejamento de Infraestruturas de Nuvens Privadas conforme os Requisitos de Custo**: esta tese possibilita o planejamento de infraestruturas de nuvens privadas que atendam aos requisitos de custo estabelecidos.

8.2 Limitação

A principal limitação deste trabalho está relacionada a solução integrada ter sido proposta para o planejamento de infraestruturas de nuvens privadas. Esse planejamento poderia considerar as infraestruturas de nuvens híbridas. Neste contexto, os modelos de otimização e os modelos de representação deveriam ser modificados para representar os componentes dessa infraestrutura.

8.3 Trabalhos Futuros

Embora esta tese apresente uma solução integrada para o planejamento de infraestruturas de nuvens privadas, considerando aspectos de desempenho, de dependabilidade e de custo, há muitas possibilidades de estender o trabalho atual. Algumas dessas possibilidades foram detalhadas abaixo:

• Este trabalho propôs o planejamento das infraestruturas de TI das nuvens privadas. Todavia, esse trabalho pode ser melhorado através do planejamento das infraestruturas energéticas, de resfriamento e de TI (SILVA et al., 2012) das nuvens privadas.

O planejamento das infraestruturas energéticas, de resfriamento e de TI proporcionará a geração de nuvens computacionais com maior disponibilidade, pois serão identificados os mecanismos de redundância que precisam ser atribuídos aos componentes dessas infraestruturas.

O planejamento dessas infraestruturas também proporcionará a geração de nuvens computacionais que tenham menores tempos de resposta e utilizações de recursos no atendimento as requisições dos usuários.

Para considerar o planejamento das infraestruturas energéticas e de resfriamento, o Modelo para Geração de Cenários de Dependabilidade e Custo deve ser modificado para representar os componentes dessas infraestruturas. O modelo de desempenho deve ser modificado e novos modelos de dependabilidade devem ser propostos para representar os componentes das infraestruturas energéticas e de resfriamento. Um modelo de custo deve ser proposto para representação do consumo energético dessas infraestruturas.

• O trabalho proposto provê infraestruturas de nuvens através da atribuição de mecanismos de redundância internos como o ativo-ativo e os ativo-passivos. Esses mecanismos de redundância são modelados através de redes de *Petri* estocásticas e diagramas de bloco de confiabilidade. Entretanto, esse trabalho pode ser expandido por meio da modelagem de mecanismos de redundância externos, ou seja georedundância (KHOSHKHOLGHI et al., 2014; SILVA et al., 2013, 2014). A estratégia de georedundância é baseada no acionamento de um *site* secundário quando houver a ocorrência de um evento de falha na nuvem computacional (*site* primário). As requisições enviadas pelos clientes são respondidas pelo *site* primário, mas quando há a ocorrência de um evento de falha, o *site* secundário é acionado. As requisições do cliente passam a ser atendidas pelo *site* secundário. Quando o *site* primário é reparado, há a migração do serviço do *site* secundário para o *site* primário (BAUER; ADAMS; EUSTACE, 2011). Um modelo baseado em redes de *Petri* estocásticas deverá ser proposto para representação da dependência entre o *site* primário e o *site* secundário.

• Esta tese apresenta um modelo para representação da manutenção corretiva de infraestruturas de nuvens privadas (HIGGINS; MOBLEY; SMITH, 2002). Esse modelo retrata aspectos de dependabilidade e custo. Esse trabalho pode ser expandido através da proposição de modelos baseados em redes de *Petri* estocásticas para representação de políticas de manutenção preventiva (HIGGINS; MOBLEY; SMITH, 2002) de infraestruturas de nuvem. Esse modelo deve representar os equipamentos substituídos, as diferentes especialidades da equipe de manutenção e os tempos entre manutenções preventivas.

• Este trabalho provê modelos de otimização baseados na metaheurística GRASP (FEO; RESENDE, 1989, 1995) para geração de cenários de infraestruturas de nuvens privadas. Entretanto, esse trabalho pode ser expandido através da confecção de modelos com base em outro método de busca local como a Busca Tabu (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007) e um método de busca populacional como os algoritmos genéticos (GENDREAU; POTVIN, 2010; LUZIA; RODRIGUES, 2009; SARKER; NEWTON, 2007).

Os modelos de otimização propostos nesta tese podem ter a fase de busca local modificada com a utilização de outro método de busca local. Esse ajuste possibilitará a comparação dos tempos de geração das infraestruturas de nuvens entre os modelos propostos e os modelos com

outro método de busca local. Novos modelos também poderão ser propostos com a utilização de um método de busca populacional e seus resultados de geração de infraestruturas de nuvens poderão ser comparados com os resultados dos modelos propostos.

• O estudo de caso apresentado foi baseado no planejamento de infraestruturas de nuvens privadas configuradas com a plataforma *Eucalyptus* para hospedagem do ambiente virtual de aprendizagem *Moodle*. No entanto, uma possível estensão deste trabalho será a proposição de um estudo de caso baseado no planejamento de infraestruturas de nuvens privadas configuradas com as plataformas *Eucalyptus* (EUCALYPTUS, 2013), *Nimbus* (PROJECT, 2013a), *OpenNebula* (PROJECT, 2013b) e *OpenStack* (PROJECT, 2013c).

A confecção desse estudo de caso necessita que o Modelo para Geração de Cenários de Dependabilidade e Custo seja modificado de forma a representar a atribuição de mecanismos de redundância aos componentes de diferentes plataformas de nuvens. De forma semelhante, o Modelo para Geração de Cenários de Desempenho e Custo deverá ser ajustado para representar a atribuição de diferentes conjuntos de *software* as máquinas virtuais de várias plataformas de nuvens.

• O Gerador de Cenários e Modelos Para o Planejamento de Infraestruturas de Nuvens Privadas (SMG4PCIP) proporciona soluções de infraestruturas de nuvens privadas. Essas soluções são providas com base em requisitos de desempenho, dependabilidade e custo. Uma possível melhoria para essa ferramenta será o ranqueamento das soluções de infraestruturas de nuvens conforme pesos estabelecidos para as métricas de desempenho, dependabilidade e custo.

O SMG4PCIP também poderá ser estendido com a indicação de soluções de infraestruturas de nuvens não viáveis e a indicação das soluções de infraestruturas de nuvens que podem ser geradas através dos parâmetros fornecidos pelos usuários.

Referências

AHSON, S. A.; ILYAS, M. **Cloud Computing and Software Services**: theory and techniques. [S.l.]: CRC Press, Inc., 2010. 458p.

AMAZON. **Amazon Elastic Compute Cloud - (Amazon EC2)**. URL: http://aws.amazon.com/pt/ec2/.

AMAZON. **Amazon Simple Storage Service - (Amazon S3)**. URL: http://aws.amazon.com/pt/s3/.

APACHE. **Apache**. URL: http://www.apache.org/.

AREAN, O. Disaster Recovery in the Cloud. Network Security. **Network Security**, [S.l.], v.2013, p.5–7, 2013.

BABCOCK, C. **Management Strategies for the Cloud Revolution**: how cloud computing is transforming business and why you can't afford to be left behind. [S.l.]: McGraw Hill Professional, 2010. 272p.

BAKOUCH, H. S. Probability, Markov Chains, Queues, and Simulation. **Journal of Applied Statistics, Taylor & Francis**, [S.l.], v.38, n.8, p.1746–1746, 2011.

BALBO, G. Introduction to Stochastic Petri Nets. **Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands, July 3-7, 2000: Revised Lectures, Springer**, [S.l.], p.84–155, 2001.

BAUER, E.; ADAMS, R. **Reliability and Availability of Cloud Computing**. [S.l.]: John Wiley & Sons, 2012. 352p.

BAUER, E.; ADAMS, R.; EUSTACE, D. **Beyond Redundancy**: how geographic redundancy can improve service availability and reliability of computer based systems. [S.l.]: Wiley. com, 2011. 330p.

BOLCH, G. et al. **Queueing Networks and Markov Chains**: modeling and performance evaluation with computer science applications. [S.l.]: Wiley-Interscience, 2006. 896p.

BRIAN, J. S.; CURTIS JR., F. **Cloud Computing - Tecnologias e Estratégias**. [S.l.]: M.Books, 2013. 256p.

CALLOU, G. et al. Impact Analysis of Maintenance Policies on Data Center Power Infrastructure. **2010 IEEE International Conference on Systems Man and Cybernetics**, [S.l.], p.526–533, 2010.

CALLOU, G. et al. Estimating Sustainability Impact of High Dependable Data Centers: a comparative study between brazilian and us energy mixes. **Computing, Springer**, [S.l.], v.95, n.12, p.1137–1170, 2013.

CENTOS. **CentOS**. URL: http://centos.org/.

CHADES, L. **Cloud Computing Opportunities and Challenges**. URL: http://www.chades.net/?p=220.

CISCO. **Cisco 7606-S Router Data Sheet**. URL: http://www.cisco.com/c/en/us/products/collateral/routers/7606-router/product_data_sheet0900aecd8057f3c8.html.

CISCO. **Nexus 9000 Switching Series**. URL: http://www.cisco.com/c/dam/en/us/products/collateral/switches/nexus-9000-series-switches/cisco-n9k-lippisreport.pdf.

D, J. et al. **Eucalyptus Beginner's Guide - UEC Edition**. [S.l.]: CSS Corp., 2010. 11p.

DANTAS, J. et al. An Availability Model for Eucalyptus Platform: an analysis of warm standy replication mechanism. **Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on**, [S.l.], p.1664–1669, 2012.

DANTAS, J. et al. Models for Dependability Analysis of Cloud Computing Architectures for Eucalyptus Platform. **International Transactions on Systems Science and Applications**, [S.l.], v.8, p.13–25, 2012.

DESROCHERS, A. A.; AL-JAAR, R. Y. **Applications of Petri Nets in Manufacturing Systems**: modeling, control, and performance analysis. [S.l.]: IEEE Press, 1995. 348p.

EADUFRPE. **Unidade Acadêmica de Educação a Distância e Tecnologia**. URL: http://www.ead.ufrpe.br/.

EBELING, C. E. **An introduction to Reliability and Maintainability Engineering**. [S.l.]: McGraw Hill, 2004. 544p.

EUCALYPTUS. **Amazon Web Services, Eucalyptus Open Source Cloud Computing Infrastructure - An Overview.** URL: http://aws.amazon.com.

FEO, T. A.; RESENDE, M. G. C. A Probabilistic Heuristic for A Computationally Difficult Set Covering Problem. **Operations Research Letters, Elsevier**, [S.l.], v.8, n.2, p.67–71, 1989.

FEO, T. A.; RESENDE, M. G. C. Greedy Randomized Adaptive Search Procedures. **Journal of Global Optimization, Springer**, [S.l.], v.6, n.2, p.109–133, 1995.

FORUM, C. I. **Cloud Adoption and Trends for 2013**. URL: http://cloudindustryforum.org/white-papers/uk-cloud-adoption-and-trends-for-2013.

FRIEDMAN, F. **Microsoft Windows Server 2003 Performance Guide**. [S.l.]: Microsoft Press Redmond, WA, USA, 2005. 696p.

FURHT, B.; ESCALANTE, A. **Handbook of Cloud Computing**. [S.l.]: Springer Publishing Company, Incorporated, 2010. 634p.

GENDREAU, M.; POTVIN, J. **Handbook of Metaheuristics**. [S.l.]: Springer, 2010. 648p. v.2.

GERMAN, R. **Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets**. [S.l.]: John Wiley & Sons, Inc. New York, NY, USA, 2000. 456p.

GHOSH, R. et al. End-to-end Performability Analysis for Infrastructure-as-a-service Cloud: an interacting stochastic models approach. **Proceedings of the 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing**, [S.l.], p.125–132, 2010.

GHOSH, R. et al. Modeling and Performance Analysis of Large Scale IaaS Clouds. **Future Generation Computer Systems, Elsevier**, [S.l.], v.29, n.5, p.1216–1234, 2013.

GHOSH, R. et al. Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud. **IEEE Transactions on Services Computing**, [S.l.], v.99, p.1, 2013.

GHOSH, R.; NAIK, V. K.; TRIVEDI, K. S. Power-performance Trade-offs in IaaS Cloud: a scalable analytic approach. In: **Anais...** Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, 2011. p.152–157.

GODARD, S. **Sysstat**. URL: http://sebastien.godard.pagesperso-orange.fr/.

GOOGLE. **Google Trends**. URL: http://www.google.com.br/trends/.

GROUP, O. M. **Unified Modeling Language**. URL: http://http://www.uml.org/.

GUIMARÃES, A. P. **Modelos para Planejamento de Redes Convergentes Considerando a Integração de Aspectos de Infraestrutura e de Negócios**. 2013. Tese (Doutorado em Ciência da Computação) — Centro de Informática, Universidade Federal de Pernambuco.

GUIMARÃES, A. P.; MACIEL, P. R. M.; MATIAS JR., R. An Analytical Modeling Framework to Evaluate Converged Networks Through Business-oriented Metrics. **Reliability Engineering & System Safety, Elsevier**, [S.l.], v.118, p.81–92, 2013.

HANDY, J. **The SSD Guy**. URL: http://thessdguy.com/seagates-big-intro-four-new-ssd-families-in-one-day/.

HAVERKORT, B. R. Markovian Models for Performance and Dependability Evaluation. In: . [S.l.]: Lectures on Formal Methods and PerformanceAnalysis, 2001. p.38–83.

HAVERKORT, B. R. et al. **Performability Modelling**: techniques and tools. [S.l.]: John Wiley & Sons Inc, 2001.

HIGGINS, L.; MOBLEY, K.; SMITH, R. **Maintenance Engineering Handbook**. [S.l.]: McGraw-Hill, 2002.

HUGOS, M. H.; HULITZKY, D. **Business in the Cloud**: what every business needs to know about cloud computing. [S.l.]: John Wiley & Sons, 2010. 205p.

INA. **Integrated Net Analyzer**. URL: http://www2.informatik.hu-berlin.de/ starke/ina.html.

JAIN, R. **The Art of Computer Systems Performance Analysis**. [S.l.]: John Wiley & Sons Chichester, 1991. v.182.

JMETER. **JMeter**. URL: http://jmeter.apache.org/.

KHOSHKHOLGHI, M. et al. Disaster Recovery in Cloud Computing: a survey. **Computer and Information Science, Canadian Center of Science and Education**, [S.l.], v.7, n.4, p.39–54, 2014.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. S. Availability Modeling and Analysis of a Virtualized System. **Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on**, [S.l.], p.365–371, 2009.

KIM, H. et al. A Trust Evaluation Model for QoS Guarantee in Cloud Systems. **International Journal of Grid and Distributed Computing**, [S.l.], v.3, n.1, p.1–10, 2010.

KONDO, D. et al. Cost-benefit Analysis of Cloud Computing Versus Desktop Grids. **Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on**, [S.l.], p.1–12, 2009.

KUO, W.; ZUO, M. J. **Optimal Reliability Modeling**: principles and applications. [S.l.]: Wiley.com, 2003. 560p.

KVM. **Kernel Based Virtual Machine**. URL: http://www.linux-kvm.org/page/Main_Page.

LAPRIE, J. C. C.; AVIZIENIS, A.; KOPETZ, H. **Dependability**: basic concepts and terminology. [S.l.]: Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1992. 268p.

LI, X. et al. The Method and Tool of Cost Analysis for Cloud Computing. **Proceedings of the 2009 IEEE International Conference on Cloud Computing**, [S.l.], p.93–100, 2009.

LIGHTTPD. **Lighttpd**. URL: http://www.lighttpd.net/.

LILJA, D. J. **Measuring Computer Performance**: a practitioner's guide. [S.l.]: Cambridge University Press, 2005. 278p.

LINS, F. A. A. **Towards Automation of Security-Aware SOA-Based Business Process**. 2012. Tese (Doutorado em Ciência da Computação) — Centro de Informática, Universidade Federal de Pernambuco.

LUZIA, L. F.; RODRIGUES, M. C. Estudo Sobre as Metaheurísticas. **IME SP**, [S.l.], p.1–38, 2009.

MACIEL, P. **Modeling of Distributed and Concurrent Systems**. URL: http://http://www.modcs.org/.

MACIEL, P. et al. **Performance and Dependability in Service Computing**: concepts, techniques and research directions, chapter dependability modeling. [S.l.]: IGI Global, 2012. 477p.

MACIEL, P. R. M.; LINS, R. D.; CUNHA, P. R. F. **Introduction of the Petri Net and Applied**. [S.l.]: X Escola de Computação, Campinas, SP, 1996. 201p.

MARSAN, M. A. et al. Modelling with Generalized Stochastic Petri Nets. **ACM Sigmetrics Performance Evaluation Review, ACM Press New York, NY, USA**, [S.l.], v.26, n.2, 1998.

MATEUS, G. R.; RESENDE, M. G. C.; SILVA, R. M. A. GRASP with Path-relinking for the Generalized Quadratic Assignment Problem. **Journal of Heuristics, Springer**, [S.l.], v.17, n.5, p.527–565, 2011.

MENASCÉ, D. A.; ALMEIDA, V. A. F. **Performance by Design**: computer capacity planning by example. [S.l.]: Prentice Hall PTR, 2005. 462p.

MENKEN, I. **Cloud Computing-The Complete Cornerstone Guide to Cloud Computing Best Practices Concepts, Terms, and Techniques for Successfully Planning, Implementing. Enterprise IT Cloud Computing Technology**. [S.l.]: Emereo Pty Ltd, 2008. 172p.

MERCURY. **Mercury**. URL: http://www.cin.ufpe.br/ bs/MercuryTool/mercury.html.

MICROSOFT. **Network Load Balancing Technical Overview**. URL: http://technet.microsoft.com/en-us/library/bb742455.aspx.

MOLLOY, M. K. **On the Integration of Delay and Throughput Measures in Distributed Processing Models**. [S.l.]: University of California, Los Angeles, 1981. 108p.

MONTGOMERRY, D. C. R.; GEORGE, C. Estatística Aplicada e Probabilidade para Engenheiros. 4. **São Paulo: LTC**, [S.l.], 2009.

MOODLE. **Moodle**. URL: http://www.moodle.org.br/.

MURATA, T. Petri Nets: properties, analysis and applications. **IEEE Computer Society**, [S.l.], v.77, n.4, p.541–580, 1989.

MYSQL. **MySQL**. URL: http://www.mysql.com/.

NATKIN, S. O. **Les Reseaux de Petri Stochastiques et Leur Application a L'evaluation des Systemes Informatiques**. [S.l.]: Conservatoire National des Arts et Metiers, 1980. 4p.

NIST. **NIST Cloud Computing Program**. URL: http://www.nist.gov/itl/cloud/index.cfm.

NURMI, D. et al. The Eucalyptus Open Source Cloud Computing System. **Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid**, [S.l.], p.124–131, 2009.

ORACLE. **Oracle Database 12c Enterprise Edition**. URL: http://www.oracle.com/br/products/database/enterprise-edition/overview/index.html.

PROJECT, L. V. S. **Linux Virtual Server**. URL: http://www.linuxvirtualserver.org/.

PROJECT, N. **Nimbus v2.10.1 Documentation**. URL: http://www.nimbusproject.org/.

PROJECT, O. **OpenNebula v4.0 Documentation**. URL: http://www.opennebula.org/documentation:documentation.

PROJECT, O. **OpenStack Documentation**. URL: http://docs.openstack.org/.

PULIAFITO, A.; RICCOBENE, S.; SCARPA, M. Evaluation of Performability Parameters in Client-server Environments. **The Computer Journal**, [S.l.], v.39, n.8, p.647–662, 1996.

RIED, S. et al. **Sizing The Cloud Understanding And Quantifying The Future Of Cloud Computing**. URL: https://www.forrester.com/Sizing+The+Cloud/fulltext/-/E-RES58161.

RITTINGHOUSE, J. W.; RANSOME, J. F. **Cloud Computing**: implementation, management, and security. [S.l.]: CRC press, 2009. 340p.

ROSENBERG, J. B.; MATEOS, A. **The Cloud at your Service**. [S.l.]: Manning, 2010. 200p.

RUPE, J. W. Reliability of Computer Systems and Networks Fault Tolerance, Analysis, and Design. **IIE Transactions, Taylor & Francis**, [S.l.], v.35, n.6, p.586–587, 2003.

SAHNER, R. A.; TRIVEDI, K.; PULIAFITO, A. **Performance and Reliability Analysis of Computer Systems**: an example-based approach using the sharpe software package. [S.l.]: Springer Publishing Company, Incorporated, 2012. 404p.

SAHNER, R.; TRIVEDI, K.; PULIAFITO, A. **Performance and Reliability Analysis of Computer Systems**: an example-based approach using the sharpe software package. [S.l.]: Kluwer Academic Publishers, 1996.

SARKER, R. A.; NEWTON, C. S. **Optimization Modelling**: a practical approach. [S.l.]: CRC Press, 2007. 504p.

SCHMIDT, K. **High Availability and Disaster Recovery**: concepts, design, implementation. [S.l.]: Springer, 2006. 410p.

SHROFF, G. **Enterprise Cloud Computing**: technology, architecture, applications. [S.l.]: Cambridge Univ Pr, 2010. 290p.

SILVA, B. et al. ASTRO: an integrated environment for dependability and sustainability evaluation. **Sustainable Computing: Informatics and Systems, Elsevier**, [S.l.], v.3, p.1–17, 2012.

SILVA, B. et al. Dependability Models for Designing Disaster Tolerant Cloud Computing Systems. **Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on**, [S.l.], p.1–6, 2013.

SILVA, B. et al. GeoClouds Modcs: a perfomability evaluation tool for disaster tolerant iaas clouds. **Systems Conference (SysCon), 2014 8th Annual IEEE**, [S.l.], p.116–122, 2014.

SILVA, B.; MACIEL, P.; ZIMMERMANN, A. Performability Models for Designing Disaster Tolerant Infrastructure-as-a-Service Cloud Computing Systems. **Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for**, [S.l.], p.647–652, 2013.

SMITH, D. J. **Reliability, Maintainability and Risk**: practical methods for engineers. [S.l.]: Elsevier, 2011. 436p.

SOUSA, E. et al. Maintenance Policy And Its Impact On The Performability Evaluation Of EFT Systems. **International Journal of Computer Science, Engineering and Applications**, [S.l.], v.2, p.95–114, 2012.

SOUSA, E. et al. Stochastic Model Generation for Cloud Infrastructure Planning. **Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on**, [S.l.], p.4098–4103, 2013.

SOUSA, E. et al. Performance and Cost Modeling Strategy for Cloud Infrastructure Planning. **Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on**, [S.l.], p.546–553, 2014.

SOUSA, E. et al. A Modeling Approach for Cloud Infrastructure Planning Considering Dependability and Cost Requirements. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, [S.l.], p.1–10, 2014.

SOUSA, E. et al. Dependability evaluation of cloud infrastructures. **Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on**, [S.l.], p.1282–1287, 2014.

TAURION, C. **Cloud Computing**: computação em nuvem: transformando o mundo da tecnologia da informação. [S.l.]: Rio de Janeiro: Brasport, 2009. 228p.

TAVARES, E. et al. Performance Evaluation of Medical Imaging Service. **Proceedings of the 27th Annual ACM Symposium on Applied Computing**, [S.l.], p.1349–1354, 2012.

TIMENET. **TimeNET**. URL: http://www.tu-ilmenau.de/sse/timenet/.

TIMENET4.1. **Modeling and Evaluation of Stochastic Petri Nets With TimeNET 4.1**. URL: https://www.tu-ilmenau.de/fileadmin/public/sse/Veroeffentlichungen/2012/VALUETOOLS2012.pdf.

TRIVEDI, K. S. **Probability & Statistics with Reliability, Queuing and Computer Science Applications**. [S.l.]: John Wiley & Sons, 2008. 830p.

TRIVEDI, K. S. et al. **Reliability Analysis Techniques Explored Through a Communication Network Example**. [S.l.]: International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability, 1996. 241p.

UBUNTU. **Ubuntu**. URL: http://www.ubuntu.com/.

VELTE, A. T. et al. **Cloud Computing**: a practical approach. [S.l.]: McGraw-Hill, 2010. 352p.

WANG, H.; PHAM, H. **Reliability and Optimal Maintenance**. [S.l.]: Springer Verlag, 2006. 346p.

WEI B., L. C.; KONG, X. Dependability Modeling and Analysis for the Virtual Data Center of Cloud Computing. **High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on**, [S.l.], p.784–789, 2011.

XEN. **The Hypervisor**. URL: http://www.xenproject.org/developers/teams/hypervisor.html.

XIE, M.; DAI, Y. S.; POH, K. L. **Computing System Reliability**: models and analysis. [S.l.]: Kluwer Academic/Plenum Publishers, 2004. 293p.

XIONG, K.; PERROS, H. Service Performance and Analysis in Cloud Computing. **Services-I, 2009 World Conference on**, [S.l.], p.693–700, 2009.

ZIMMERMANN, A. et al. Towards version 4.0 of TimeNET. **Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2006 13th GI/ITG Conference**, [S.l.], p.1–4, 2006.

ZIMMERMANN, A.; KNOKE, M. **A Software Tool the Performability Evaluation with Stochastic and Colored Petri Nets**. [S.l.]: Technische Universitt Berlin. Real-Time Systems and Robotics Group, 2007. 104p.