Pós-Graduação em Ciência da Computação

Jamilson Ramalho Dantas

Planejamento de infraestrutura de nuvens computacionais para serviço de *VoD streaming* considerando desempenho, disponibilidade e custo

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

RECIFE

2018

Jamilson Ramalho Dantas

Planejamento de infraestrutura de nuvens computacionais para serviço de *VoD streaming* considerando desempenho, disponibilidade e custo

Este trabalho foi apresentado à 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: Paulo Romero Martins Maciel

RECIFE
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

D192p Dantas, Jamilson Ramalho
Planejamento de infraestrutura de nuvens computacionais para serviço de VoD streaming considerando desempenho, disponibilidade e custo / Jamilson Ramalho Dantas. – 2018.
124 f.: il., fig., tab.

Orientador: Paulo Romero Martins Maciel.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Ciência da computação. 2. Computação em nuvem. I. Maciel, Paulo Romero Martins (orientador). II. Título.

004 CDD (23. ed.) UFPE- MEI 2018-068

Jamilson Ramalho Dantas

Planejamento de Infraestrutura de Nuvens Computacionais para Serviço de VoD Streaming Considerando Desempenho, Disponibilidade e Custo

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 02/03/2018

_____

Orientador: Prof. Dr. Paulo Romero Martins Maciel

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Roberto Freire Cunha
Centro de Informática /UFPE

_____

Prof. Dr. Ricardo Massa Ferreira Lima
Centro de Informática /UFPE

_____

Prof. Dr. Eduardo Antônio Guimarães Tavares
Centro de Informática /UFPE

_____

Prof. Dr. Artur Ziviani
Laboratório Nacional de Computação Científica

_____

Prof. Dr. José Neuman de Souza
Departamento de Computação /UFC

Dedico esta tese a minha família e amigos, em especial ao meu Pai, Oscar Ramalho Dantas (in memoriam), que foi e é o meu maior exemplo. Espero ter sido merecedor de todo o seu amor, carinho e dedicação, e honrá-lo com a obtenção do título de Doutor.

Agradecimentos

Gostaria de iniciar meus agradecimentos para àquele que nos concede o dom da vida, ao bom Deus por ter me dado os instrumentos necessários para chegar até aqui, e por ter me permitido a força e persistência para não desistir diante das dificuldades encontradas no caminho.

Agradeço ao Prof. Paulo Maciel, que acreditou em meu potencial, me orientou durante toda essa jornada e ajudou em meu crescimento pessoal e profissional, ao qual tenho uma eterna dívida de gratidão.

Agradeço também a toda a minha família, que sempre esteve comigo. Em especial à minha mãe Cícera, a quem amo incondicionalmente, presente em todos os momentos da minha vida, por me ensinar a lutar e nunca desistir dos meus objetivos.

Gostaria de agradecer também a minha eterna namorada e esposa, Renata, pela paciência, cumplicidade e ajuda em todos esses anos de luta. Ao meu filho, Davi, a quem peço perdão pelas ausências nos seus 5 primeiros anos de vida, obrigada pelo apoio, pelos sorrisos sinceros e pelo amor que me fez reconhecer que o preço do doutorado era muito alto diante de tantas abdicações e me fez conseguir superar.

Meu muito obrigado a todos os companheiros do grupo de pesquisa MoDCS. Em especial a Rubens, Jean e Danilo, Carlos Melo, amigos para todos os desafios, pelas valiosas dicas e por estarem sempre dispostos a ajudar. A João, parceiro desde o Mestrado. A Maria Clara e Rosângela que estiveram comigo acompanhando o desenvolvimento do trabalho. E a todos os colegas que me ajudaram durante toda essa jornada, em especial, Ronierison Maciel, Erico e Aline.

Enfim, aos que amo, que de forma direta ou indireta contribuíram para essa conquista.

A todos, meu sincero agradecimento.

"Por aqui, no entanto, não olhamos para trás por muito tempo. Continuamos a avançar, abrindo novas portas e fazendo coisas novas, porque somos curiosos... e a curiosidade continua nos levando a novos caminhos."

(Walt Disney Company)

Resumo

O paradigma da computação em nuvem vem sendo amplamente utilizado ao longo dos últimos anos devido às características de provisionamento de recursos de forma escalável, que permitem uma gestão equilibrada dos recursos de acordo com a demanda. Em função das vantagens desse modelo computacional, empresas que oferecem serviços na internet acabam optando por adotar a computação em nuvem como alicerce da infraestrutura computacional, de forma a garantir o bom funcionamento dos serviços oferecidos. Desta forma, este trabalho apresenta uma solução integrada composta por modelos de desempenho, disponibilidade e custo, além de um mecanismo para avaliação de espaço de projeto de infraestruturas de nuvem computacionais. O mecanismo de avaliação de espaço de projeto é baseado na metaheurística GRASP para geração de cenários de infraestruturas de nuvens privadas. Os modelos de disponibilidade e disponibilidade orientada à capacidade (COA), baseiam-se em modelos RBDs e CTMCs para avaliação e geração de fórmulas matemáticas fechadas. Esses modelos permitem a avaliação da disponibilidade e COA para uma infraestrutura de nuvem considerando n possíveis combinações de máquinas físicas para infraestruturas de nuvens privadas com j máquinas virtuais. Os modelos de desempenho são baseados em redes de Petri estocásticas. O modelo de desempenho permite a avaliação do tempo de resposta das configurações de infraestruturas de nuvens computacionais. Os modelos de custo são baseados em equações fechadas e baseiam-se, apenas, nos custos de aquisições de máquinas físicas, para infraestrutura privada e custos de aquisição de máquinas virtuais, considerando infraestrutura de nuvem pública. Cinco estudos de caso são apresentados para aplicação da solução proposta. O primeiro estudo de caso apresenta uma avaliação de disponibilidade para a plataforma de nuvem privada EUCALYPTUS, considerando diferentes configurações de cenário de avaliação. O segundo estudo de caso considera a avaliação de disponibilidade por meio de uma equação genérica fechada considerando combinações de máquinas físicas em uma nuvem privada. Esse estudo de caso possibilita a validação do estudo proposto por meio de arquiteturas semelhantes através da modelagem CTMC. O terceiro estudo de caso considera um modelo de desempenho para avaliação do tempo de resposta da infraestrutura de nuvem privada para a plataforma EUCALYPTUS. O quarto estudo de caso considera a validação do modelo de desempenho através de experimentos específicos em ambientes reais. O quinto estudo de caso apresenta a junção dos modelos apresentados nos estudos de caso anteriores para geração de infraestruturas candidatas em relação à arquitetura de nuvem privada. Considera-se, então, a metaheurística GRASP como objeto avaliativo na seleção de soluções.

Palavras-chaves: Computação em nuvem. Disponibilidade. COA. Desempenho. Custo. Metaheurística GRASP.

Abstract

Cloud computing paradigm has been widely used over the last years due to the scalable resource provisioning features, allowing for a balanced management of resources according to the demand. According to the advantages of this computer model, companies offering services on the Internet prefer to choose the cloud computing as the basis of computing infrastructure, to ensure the proper functioning of the offered services. In this way, this work presents an integrated solution composed of performance, availability and cost models, as well as a mechanism for evaluating the space of computational cloud infrastructure design. The project space evaluation mechanism is based on the GRASP metaheuristic for generating private cloud infrastructure scenarios. The steady-state availability and capacity-oriented availability (COA) models are based on RBDs and CTMCs for evaluation and generation of closed mathematical functions. These models allow assessment of steady-state availability and COA for a cloud infrastructure considering n possible combinations of physical machines for private cloud infrastructures with j virtual machines. The performance models are based on stochastic Petri nets. The performance model allows the evaluation of the response time of the computational cloud infrastructure configurations. The cost models are based on closed-form equations and are based only on the costs of physical machine acquisitions, considering a private infrastructure, and virtual machine acquisition costs, considering public cloud infrastructure. Five case studies are performed to the application of the proposed solution. The first case study presents an availability assessment for the private cloud platform EUCALYPTUS, considering different configurations of the evaluation scenario. The second case study considers the availability assessment using a closed generic equation considering combinations of physical machines in a private cloud. This study allows a validation method through similar architectures using CTMC modeling. The third case study considers a performance model for assessing the response time of the private cloud infrastructure for the EUCALYPTUS platform. The fourth case study considers the validation of the performance model through specific experiments in real environments. The fifth case study shows the combination of the models presented in the previous studies for the generation of candidate infrastructures in relation to the private cloud architecture. GRASP metaheuristic is considered as an evaluative object in the selection of solutions.

Key-words: Cloud computing. Availability. COA. Performance. Cost. GRASP metaheuristic.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABR *Adaptive bit rate.*

API *Application Programming Interface.*

AWS *Amazon Web Services.*

CC *Cluster Controller.*

CLC *Cloud Controller.*

COA disponibilidade orientada a capacidade.

CPN Rede de Petri Colorida.

CTMC Cadeia de Markov de tempo contínuo.

DTMC cadeia de Markov de tempo discreto.

EBS *Elastic Block Store.*

EC2 *Amazon Elastic Compute Cloud.*

FFMPEG *Fast Forward Motion Picture Experts Group.*

IaaS *Infrastructure as a service.*

iSCSI *Internet Small Computer Systems Interface.*

MPD *Media presentation data.*

MRM *markov reward model.*

MTTF tempo médio de falha.

MTTR tempo médio de reparo.

NC *Node Controller.*

NFS *Network file system.*

NIST *National Institute of Standards and Technology.*

PaaS *Platform as a Service.*

PN Rede de petri.

RBD Diagrama de bloco de confiabilidade.

RCL *restricted candidate list*.

RdP Rede de Petri.

S3 *Simple Storage Service*.

SaaS *Software as a service*.

SC Controlador de Armazenamento - *Storage Controller*.

SLA Acordo de níveis de serviço.

SOS *Scalable Object Storage*.

SPN Rede de petri estocásticas.

VoD vídeo sob demanda.

Sumário

1 INTRODUÇÃO

Este capítulo apresenta uma breve introdução sobre computação em nuvem, destacando aspectos de dependabilidade, desempenho e custo a serem apresentadas no decorrer do trabalho. Em seguida, são apresentados os objetivos e as motivações desta tese e, por fim, a estrutura geral.

1.1 Contexto

A computação em nuvem fornece recursos computacionais (poder de processamento, armazenamento, software, etc.) disponíveis através da Internet, com prestadores de serviço que podem estar localizados em qualquer lugar no mundo.

O paradigma da computação em nuvem vem sendo amplamente utilizado ao longo dos últimos anos devido às características de provisionamento de recursos de forma escalável, o que permite uma gestão equilibrada dos recursos de acordo com a demanda, além de oferecer serviços de disponibilidade e confiabilidade elevadas. Por ser uma plataforma ideal para tarefas que exigem alto poder computacional, as nuvens podem ser adequadas também para suportar aplicações que envolvem grande fluxo de dados, como períodos de pico, fornecendo quantidades elásticas da largura de banda e outros recursos em tempo real (WU et al., 2011).

A natureza elástica e a política de "pague apenas o que usar"(*pay-as-you-go*) fazem com que esse paradigma computacional seja bastante promissor para várias aplicações (FOX et al., 2009) (ARMBRUST et al., 2010b). Grande parte dos serviços online faz uso da computação em nuvem na tentativa de garantir disponibilidade e entrega do serviço de forma apropriada para os seus usuários (ZHU et al., 2011), em particular, serviços de vídeo sob demanda (ex. YouTube (YOUTUBE, 2017a), Netflix (NETFLIX, 2016)), que devem estar preparados para receber acessos na ordem dos milhões de pessoas/dia (YOUTUBE, 2017b).

Os administradores de sistemas devem estar preparados para planejar ambientes adequados e que atendam à demanda existente, oferecendo disponibilidade, segurança e confiabilidade dos dados. Os prestadores de serviços de computação em nuvem, por sua vez, devem garantir a entrega total do serviço por meio de Acordo de níveis de serviço (SLA), (LUDWIG et al., 2003).

Nesse cenário, a avaliação de dependabilidade (AVIZIENIS et al., 2001) aparece como uma atividade essencial para promover a melhoria da qualidade do serviço prestado, assim como para o planejamento de infraestruturas de sistemas de computação. Convém ressaltar que falhas em sistemas são inevitáveis, mas podem ser contornadas por meio de técnicas adequadas.

Inúmeras técnicas têm sido propostas e adotadas para construção de *cluster de failover* (LIU; SONG, 2003). Muitas delas são baseadas em redundância, isto é, em replicação do componente de modo que várias máquinas trabalhem para um fim comum e garantam a segurança dos dados e a disponibilidade, mesmo em caso de falha de algum componente.

Modelagem analítica, tais como Cadeia de Markov (REIBMAN; SMITH; TRIVEDI, 1989), Redes de Petri (HIREL; TUFFIN; TRIVEDI, 2000) e/ou Diagrama de Bloco de Confiabilidade (MACIEL et al., 2011a) ajudam a planejar e gerenciar infraestruturas de *hardware*, rede e *software* (DANTAS et al., 2015) (DANTAS et al., 2012) (CALLOU et al., 2012), comparando-as com configurações alternativas ou permitindo a previsão de efeitos na disponibilidade, confiabilidade e desempenho de sistemas, com aplicação de mudanças necessárias.

Considerando as abordagens mencionadas, essa tese tem como objetivo desenvolver métodos para planejar uma infraestrutura de computação em nuvem privada. Modelos de desempenho, disponibilidade e disponibilidade orientada à capacidade (COA) e custo foram elaborados para auxiliar no desenvolvimento de cenários. Os modelos gerados também foram utilizados como entrada para uma metaheurística de otimização que buscou configurações que cumprissem o SLA ao mesmo tempo que minimizassem o custo.

1.2 Motivação e Justificativa

A busca por sistemas que forneçam um nível de disponibilidade com tempo de respostas e custos aceitáveis é cada vez maior em empresas que buscam oferecer serviços de forma ininterrupta, e, mesmo com a evolução das tecnologias envolvidas, é difícil garantir que um serviço seja insuscetível a falhas. Interrupções em qualquer tipo de serviço podem vir a comprometer o funcionamento do sistema e, assim, ocasionar insatisfação dos clientes e prejuízos financeiros expressivos.

Soluções a partir de *Software* têm sido criadas com o intuito de prover técnicas de virtualização e fornecer serviços em nuvem, numa tentativa de obter confiabilidade dos dados, alta disponibilidade, segurança, economia de energia, melhor desempenho, entre outras vantagens. *HPE Helion Eucalyptus* (DOCUMENTATION, 2017), *apache cloudStack* (APACHE, 2017a) e o *openstack* (OPENSTACK, 2017) são algumas soluções para infraestrutura de nuvem privada e híbridas. Elas fornecem *Infrastructure as a service* (IaaS), permitindo o controle de grandes coleções de recursos. Ainda assim, a confiabilidade desse tipo de sistema pode ser afetada por falhas ou atualizações de *software*, manutenções planejadas e troca de equipamentos.

Serviços na nuvem que apresentem um alto ou constante índice de falha, confiabilidade e segurança poderão oferecer uma experiência negativa ao usuário, acarretando em prejuízos ao provedor do serviço. O usuário possui diversas maneiras de criticar um serviço ruim na Internet e ainda tem a possibilidade de utilizar serviços concorrentes em questão de segundos. Assim, um sistema na nuvem está vulnerável a forte rejeição dos usuários, caso

não ofereça um serviço de qualidade e de alta disponibilidade.

O planejamento de infraestruturas de computação em nuvem envolve o uso de várias técnicas. Alguns trabalhos propõem o uso de técnicas para o planejamento e avaliação de sistemas em *cloud* considerando serviços (SOUSA et al., 2014); (SOUSA et al., 2014) e (MATOS et al., 2016). Os trabalhos apresentam focos específicos em disponibilidade e desempenho. Alguns outros (MELO et al., 2014); (BEZERRA et al., 2014); (YANG et al., 2015) e (QIU et al., 2012) estão inseridos no contexto de sistemas de VoD. Esses trabalhos fazem uso de técnicas específicas de modelagem de sistema e avaliação de desempenho por meio de experimentos em ambientes reais. Os estudos apresentados mostram avaliações isoladas em ambientes diversos. O uso de técnicas, métodos e modelos é muito importante para a avaliação de dependabilidade e desempenho de sistemas, já que tal avaliação pode promover melhorias na qualidade do serviço provido e no planejamento das infraestruturas. Além disso, a utilização de modelos para representar o funcionamento de sistemas permite resultados confiáveis a um custo baixo, visto que não é necessário construir um sistema real para analisá-lo.

1.3 Objetivos

O envio de dados multimídia para uma determinada plataforma exige, muitas vezes, a transcodificação para um formato que é suportado pelo ambiente. Para algumas arquiteturas, as mídias digitais devem estar acessíveis a qualquer hora do dia ou da noite. O grande desafio é identificar uma configuração que atenda aos requisitos de desempenho, disponibilidade e custo.

O principal objetivo deste trabalho é a proposição de uma solução integrada composta por modelos de desempenho, disponibilidade e custo, assim como e um mecanismo para avaliação de espaço de projeto para infraestrutura de nuvem com suporte a um serviço de vídeo. Essa solução integrada permitirá o planejamento de infraestruturas de nuvens privadas de acordo com os requisitos estabelecidos com os usuários. A abordagem deve planejar um ambiente de nuvem para um sistema de video sob demanda considerando aspectos de desempenho (processo de conversão da mídia de vídeo), disponibilidade e COA e custo de aquisição de máquinas físicas para construção da infraestrutura privada.

Especificando melhor, o trabalho ora posto possui os seguintes objetivos específicos:

- Construir modelos de desempenho, disponibilidade e disponibilidade orientada à capacidade (COA) para arquiteturas de computação em nuvem, com foco em serviços de vídeo sob demanda (VoD) *streaming*;
- Validar os modelos de desempenho, disponibilidade e disponibilidade orientada à capacidade (COA) das arquiteturas mencionadas;

- Confecção de modelos de custo por meio de expressões algébricas que permitam a avaliação das infraestruturas de nuvem privadas e/ou públicas;
- Desenvolver algoritmo de otimização, contemplando os modelos de desempenho e disponibilidade do formato de nuvem;
- Aplicar os modelos e a otimização em estudos de caso, visando planejar uma infraestrutura de nuvem ideal para os ambientes propostos.

1.4 Contribuições

As principais contribuições desta tese é a exposição de:

- Modelos para estimar a disponibilidade e disponibilidade orientada à capacidade disponibilidade orientada a capacidade (COA) para sistemas de computação em nuvem, com foco em direcionados para infraestruturas privadas que atendam às necessidades de sistemas de pequeno porte.
- Modelo de avaliação de desempenho para transcodificação de vídeo, que faz uso de três tipos de VMs, destacando-se que o conjunto de trabalho, bem como o tipo de VM a ser utilizada, podem gerar combinações e quantidades específicas de VMs para o ambiente;
- Modelos de custo para avaliação dos custos para infraestrutura privada, considerando aquisição de máquinas físicas e modelo dos custos para aquisição de máquinas virtuais, tendo em vista uma nuvem pública;
- Algoritmo de otimização de infraestruturas de computação em nuvem, contemplando COA, desempenho e custo para serviço de VoD *streaming*. O algoritmo deve parametrizar o modelo de desempenho a fim de escolher uma melhor combinação de parâmetros que atendam a um SLA específico.

1.5 Organização do Trabalho

O Capítulo 2 apresenta a fundamentação teórica do trabalho, bem como conceitos de computação em nuvem e dependabilidade, técnicas para a avaliação da dependabilidade por meio de modelagem hierárquica, que incluem redes de Petri estocásticas, cadeia de Markov e diagrama de blocos de confiabilidade. O Capítulo 3 apresenta os trabalhos atuais relevantes à pesquisa, encontrados na literatura, que demonstram a diferenciação desta tese em relação aos demais estudos realizados. No Capítulo 4 são apresentados a metodologia e os métodos da solução integrada para o planejamento de infraestruturas de nuvens para serviços de VoD streaming. O Capítulo 5 aborda a descrição dos modelos

utilizados para elaboração dos estudos de caso do capítulo seguinte. O Capítulo 6 trata de cinco estudos de caso para aplicação da solução integrada proposta. Estes estudos de acaso são baseados no planejamento de infraestrutura de nuvem para serviços de VoD *streaming*. As conclusões e trabalhos futuros são expostos no Capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos básicos para o entendimento do trabalho proposto. Inicialmente, será apresentado os conceitos de computação em nuvem, bem como *VoD streaming* e transcodificação de vídeo. Em seguida, é apresentado uma introdução das definições dos conceitos básicos sobre desempenho e dependabilidade. Posteriormente, são apresentados as técnicas de modelagem em Cadeia de Markov de tempo contínuo (CTMC). Logo após, apresenta os principais conceitos sobre Rede de petri (PN), assim como características, propriedades e técnicas de análise. Em seguida, são apresentadas as Rede de petri estocásticas (SPN), que são uma extensão à teoria inicial das redes de Petri. Por fim é apresentado o Diagrama de bloco de confiabilidade (RBD).

2.1 Computação em Nuvem

A computação em nuvem é um modelo de computação onde recursos computacionais tais como poder de processamento, rede, armazenamento e *software* são oferecidos na internet e podem ser acessados remotamente (ARMBRUST et al., 2010a). A computação em nuvem fornece uma gama de informações em relação às quais os usuários não precisam se preocupar quanto à localização; precisam saber apenas que elas existem e que poderão acessá-las de qualquer lugar do mundo.

Uma definição formal de computação em nuvem pode ser retirada de National Institute of Standards and Technology *(NIST)*, Instituto Nacional de Padrões e Tecnologia dos Estados Unidos da América, onde se diz que computação em nuvem é um modelo que permite acesso ubíquo, conveniente, sob demanda a um *pool* compartilhado de recursos computacionais (i.e, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente configurados e liberados com o esforço de gerenciamento mínimo ou interação com o provedor de serviços (MELL; GRANCE, 2011).

Os avanços tecnológicos e a necessidade de utilizar grande poder de computação têm ajudado na utilização de tecnologias em nuvem. A computação em nuvem é importante para a distribuição e acesso de recursos de computação, oferecendo algumas vantagens de recursos computacionais (ARAúJO, 2012).

- Escalabilidade: Em um sistema de nuvem, recursos de computação são altamente escaláveis, ou seja, possuem capacidade de escala para adição de recursos ou desativar os mesmos, em resposta à evolução das necessidades da aplicação do usuário (EUCALYPTUS, 2010).

- Segurança: Acessos realizados a aplicativos devem ser realizados apenas por pessoas autorizadas; usuários devem confiar seus dados à empresas que fornecem os serviços (CAROLAN; GAEDE, 2009).

- Disponibilidade Estacionária: Espera-se que sites de relacionamentos, revistas e jornais eletrônicos ou qualquer sistema computacional sejam disponíveis na ordem de 24x7, ou seja, vinte e quatro horas por dia e sete dias por semana durante um ano (CAROLAN; GAEDE, 2009). Assim, defini-se disponibilidade do sistema como a probabilidade de que o dispositivo esteja disponível sempre que necessário (KUO; ZUO, 2002). Os usuários requerem funcionamento a qualquer hora e em qualquer lugar.

- Confiabilidade: A confiabilidade é definida como a probabilidade de que um dispositivo irá desempenhar as suas funções pretendidas satisfatoriamente durante um período especificado de tempo, sob condições de operação específicas. Com base nesta definição, a confiabilidade é medida como uma probabilidade (KUO; ZUO, 2002). Diferente da disponibilidade, a confiabilidade é definida em termos de um intervalo de tempo em vez de um instante de tempo (CAROLAN; GAEDE, 2009).

Modelo de Negócio - Os serviços oferecidos na computação em nuvem envolvem plataforma de *hardware* e *software* sob demanda. Em geral, de acordo com o tipo de capacidade fornecida, os serviços de computação em nuvem são amplamente divididos em três categorias: Infraestrutura como um Serviço (IaaS), Plataforma como um Serviço (*Platform as a Service* (PaaS)) e Software como um Serviço (*Software as a service* (SaaS)) (GONG et al., 2010) (EUCALYPTUS, 2012).

IaaS (infraestrutura como serviço) fornece uma infraestrutura de armazenamento, processamento e rede como serviço onde o cliente pagará apenas por aquilo que usar. Por exemplo, em um serviço de armazenamento, o usuário não pagará pelo disco completo, apenas pela capacidade que estiver consumindo. Empresas que disponibilizam esse tipo de serviço oferecem recursos computacionais na internet por meio de máquinas virtuais. Exemplos de serviços, ou produtos, de IaaS são *Google Compute Engine* (GOOGLE, 2017a), *Amazon EC2* (AMAZON, 2017a) e *GoGrid* (GOGRID, 2017).

PaaS (plataforma como serviço) fornece uma plataforma de desenvolvimento como serviço; suporta um conjunto de interface de aplicativos de programas para aplicações na nuvem. É a ponte intermediária entre hardware e aplicação. Com PaaS os usuários podem realizar ações como desenvolver, compilar, *debugar*, além de implantação e teste na nuvem. Exemplos de serviços são *Google App Engine* (ENGINE, 2017) e *Windows Azure* (CORPORATION, 2017).

SaaS (*Software* como serviço) fornece um conjunto de software como serviço na nuvem, visando à substituição daqueles antes instalados nos computadores pessoais. Usuários da nuvem podem liberar seus aplicativos em um ambiente de hospedagem que pode ser

acessado na rede por diversos clientes (DILLON; WU; CHANG, 2010). Com objetivo de vender software como serviço, o cliente pagará um valor relativamente menor do que o valor de compra do software em questão. Exemplos de tipos de serviços são *Google Docs* (GOOGLE, 2017b), *Sales Force* (SALESFORCE, 2016) e serviços de e-mail como o *Gmail* (GOOGLE, 2017c).

Tipos de Nuvens - Há muitas questões a serem consideradas quando se trata de serviços oferecidos pelas empresas provedoras de serviços em nuvem. Enquanto algumas prestadoras de serviços estão interessadas em reduzir o custo de operação, outras podem preferir alta confiabilidade e segurança de dados (ZHANG; CHENG; BOUTABA, 2010). Os tipos de nuvens (ver Figura 1) podem ser descritos quanto à natureza de acesso e controle em relação ao uso e provisionamento de recursos físicos e virtuais. Assim, há três tipos de nuvens, cada uma com suas vantagens e desvantagens:

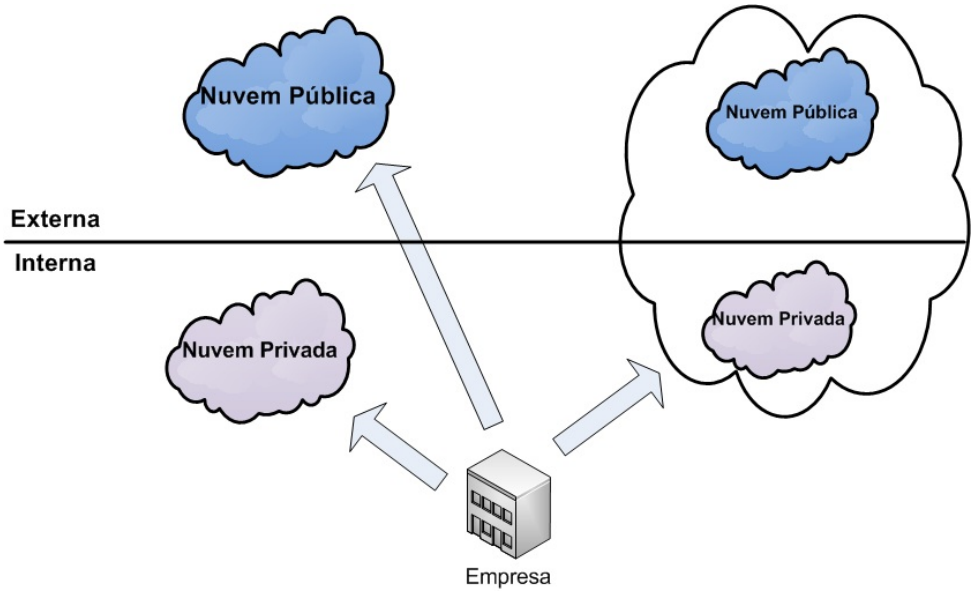


Figura 1 – Tipos de Nuvens Adaptado de (FURHT; ESCALANTE, 2010)

Nuvens Privadas - Esse tipo de implantação da nuvem é restrita apenas à empresa que adota tal recurso, ficando assim a responsabilidade da mesma em manter seus serviços funcionando. A empresa possui a infra-estrutura e tem controle sobre como os aplicativos são implantados nela (CAROLAN; GAEDE, 2009). Nesse ponto de vista, a empresa possui um alto grau de segurança, confiabilidade e controle da nuvem.

Nuvens públicas - Empresas prestadoras de serviços oferecem recursos para o público em geral. Na nuvem pública, recursos de computação são fornecidos dinamicamente através da Internet por meio de aplicações Web ou serviços da Web, a partir de um fornecedor externo (FURHT; ESCALANTE, 2010). Esse tipo de nuvem é instalada em locais fora das instalações dos clientes, de maneira que custos e riscos do cliente são minimizados. Assim, um dos benefícios desse tipo de nuvem é que elas podem ser muito maiores do que as nuvens privadas, oferecendo maior capacidade de ajuste à demanda e transferindo a

responsabilidade de gerenciamento de infra-estrutura para o provedor da nuvem (CAROLAN; GAEDE, 2009).

Nuvens Híbridas - Esse tipo de modelo é a junção dos dois modelos mencionados anteriormente, onde tanto a empresa fornecedora do serviço quanto o usuário são responsáveis por manter o serviço em funcionamento, configurando-o de tempos em tempos. Uma nuvem híbrida também pode ser usada para lidar com picos de carga de trabalho planejados. Nuvens híbridas oferecem mais flexibilidade do que nuvens públicas e privadas. Especificamente, eles provêem maior controle e segurança sobre os dados e aplicativos em comparação com as nuvens públicas, enquanto ainda facilitam a expansão sob demanda e contração de serviços (ZHANG; CHENG; BOUTABA, 2010).

2.2 *VoD streaming* e transcodificação de vídeo

O serviço de *streaming* é uma tecnologia utilizada na transmissão de multimídia digitais através da Internet (DELGADO; FRÍAS; IGARTUA, 2006). Esta transmissão permite que dados sejam entregues e vistos através de um fluxo contínuo transmitido pela rede.

De um ponto de vista comercial, a tecnologia de vídeo sob demanda (*VoD*) está inteiramente associada com a computação em nuvem, o que permite o uso de recursos computacionais de acordo com a demanda assegurando a confiabilidade, segurança e melhor desempenho. Desta forma, usuários podem pagar por uma taxa fixa pelo serviço ininterrupto e ganha a liberdade e flexibilidade em termos do que eles assistem e quando assistem (DÍAZ-SÁNCHEZ et al., 2011).

Um sistema de transmissão de vídeo é mostrado na Figura 2, que consiste de um codificador, um servidor de distribuição com armazenamento de vídeo, um servidor de retransmissão e utilizadores finais que recebem os dados de vídeo (SUN; VETRO; XIN, 2007). O servidor de distribuição armazena os dados de vídeo codificados e começa a distribuir os dados de acordo com a demanda do cliente. Os usuários podem ver o vídeo quando e onde quiserem, acessando o servidor através das redes. A codificação e a distribuição é realizada em tempo real, no caso de distribuição ao vivo e não podem ser realizadas em tempo real, para o tipo a pedido de aplicações.

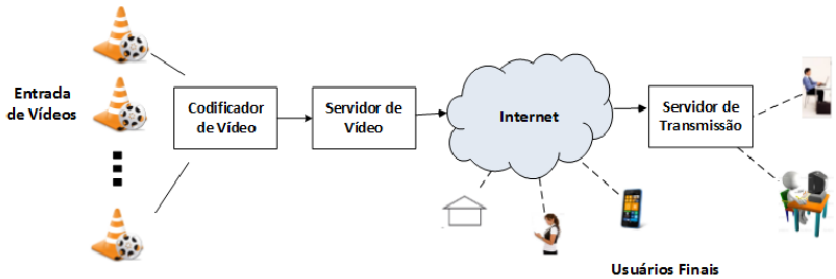


Figura 2 – Sistema de streaming de vídeo (adaptado de (SUN; VETRO; XIN, 2007)

Contudo, há uma grande variedade de dispositivos buscando compartilhar conteúdo de vídeo na Internet, cada um com capacidades e características de rede heterogêneas. Por exemplo, dispositivos gravando vídeos em alta qualidade que serão compartilhados para visualização através de *smartphones*, que em muitos casos são dispositivos de baixa capacidade, com telas pequenas, resolução limitada e restrita quantidade de codecs aceita. Outra restrição vem da variedade de acesso à rede (Ethernet, WIFI, 4g, 3g, entre outras), cada uma com diferentes larguras de banda, taxa de erros, retransmissões, perda de pacotes e flutuações (AHMAD et al., 2005).

Uma solução para esse problema, é codificar um mesmo vídeo em vários formatos para entregar a mídia adequada a cada um desses diferentes dispositivos, condições de rede e experiência de usuário. Estratégia utilizada pelo *Adaptive bit rate* (ABR), uma técnica amplamente utilizada para oferecer vídeo sob demanda (Video On Demand - *VoD*). O ABR requer que o vídeo seja codificado em diferentes versões de qualidade permitindo que o cliente escolha a versão apropriada do vídeo para o seu dispositivo e estado atual da rede (que pode mudar durante a exibição do vídeo). O ABR também requer que o vídeo seja dividido em segmentos de 2 a 10 segundos e armazenado junto com seus metadados *Media presentation data* (MPD) (KRISHNAPPA; ZINK; SITARAMAN, 2015) (STOCKHAMMER, 2011).

Neste trabalho utilizaremos o *Fast Forward Motion Picture Experts Group* (FFMPEG) para realizar as transcodificações de vídeo. Ele é um software livre capaz de transcodificar em uma ampla variedade de formatos atuais e obsoletos. Também é capaz de executar em diversos sistemas operacionais (FFMPEG, 2017).

2.3 Desempenho

Para a avaliação de desempenho de sistemas deve-se considerar um conjunto de técnicas, podendo ser classificadas em duas vertentes: técnicas baseadas em medição e técnicas baseadas em modelagem (LILJA, 2005).

Técnicas baseada em medição requerem a construção de um ambiente real e envolve a monitoração do sistema enquanto está sob a ação de uma carga de trabalho. Antes da aplicação da carga de trabalho no sistema deve-se ter um estudo primário sobre a carga que deverá ser aplicada. A escolha da carga de trabalho é tão importante quanto à definição de qual estratégia de medição deve ser seguida, pois é a partir dela que deverá escolher ferramentas e estratégias de medição.

Ferramentas que auxiliam a avaliação de desempenho de sistemas 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 (LILJA, 2005). Desta forma, ferramentas de medição

dirigida a evento são, de certa forma, mais confiáveis pois oferecem uma menor pertubação aos dados medidos, ocasionadas apenas quando a ocorrência de eventos, por outro lado, a frequência dos eventos é que vai determinar a pertubação ocasionada pelos eventos gerados. Outro tipo de ferramenta de medição são as por amostragem, esse tipo de ferramenta ocasiona pertubações independente do número de vezes que o evento ocorre (LILJA, 2005) (MENASCE et al., 2004).

Técnicas baseadas em modelagem podem ser resolvidas tanto analiticamente, quanto por simulação. Os modelos analíticos utilizam 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).

Já os modelos de simulação podem ser utilizados tanto na avaliação de desempenho de sistemas, quanto na validação dos 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 (LILJA, 2005) (MENASCE et al., 2004).

2.4 Dependabilidade

Há várias definições de dependabilidade. Uma das definições diz que dependabilidade do sistema pode ser entendido como a capacidade de oferecer uma funcionalidade específica, que pode ser justificadamente confiável (AVIZIENIS et al., 2004), ou ainda, que o sistema irá executar ações especificadas ou apresentar resultados específicos de maneira confiável e em tempo oportuno (PARHAMI, 1988). Dependabilidade abrange medidas como a disponibilidade, confiabilidade e segurança.

Devido à prestação de serviços onipresente na Internet, a confiabilidade tornou-se um atributo de interesse principal em hardware / software de desenvolvimento, implantação e operação (MACIEL et al., 2011b). Como a computação em nuvem é um paradigma de computação distribuída em grande escala e suas aplicações são acessíveis em qualquer lugar e em qualquer momento, a dependabilidade de sistemas na nuvem está se tornando mais importante e mais difícil de alcançar (SUN et al., 2010).

Em (AVIZIENIS et al., 2001), é apresentada uma exposição sistemática dos conceitos de dependabilidade, que podem ser observados na Figura 3 e são mencionados a seguir:

- Os atributos: possibilitam a obtenção de medidas quantitativas, que muitas vezes são cruciais para a análise dos serviços oferecidos.

- Os meios: são os meios pelos quais a dependabilidade é atingida.
- As ameaças: compreendem as falhas, erros e defeitos. A falha do sistema representa o evento que ocorre quando a entrega do serviço não acontece da maneira desejada.

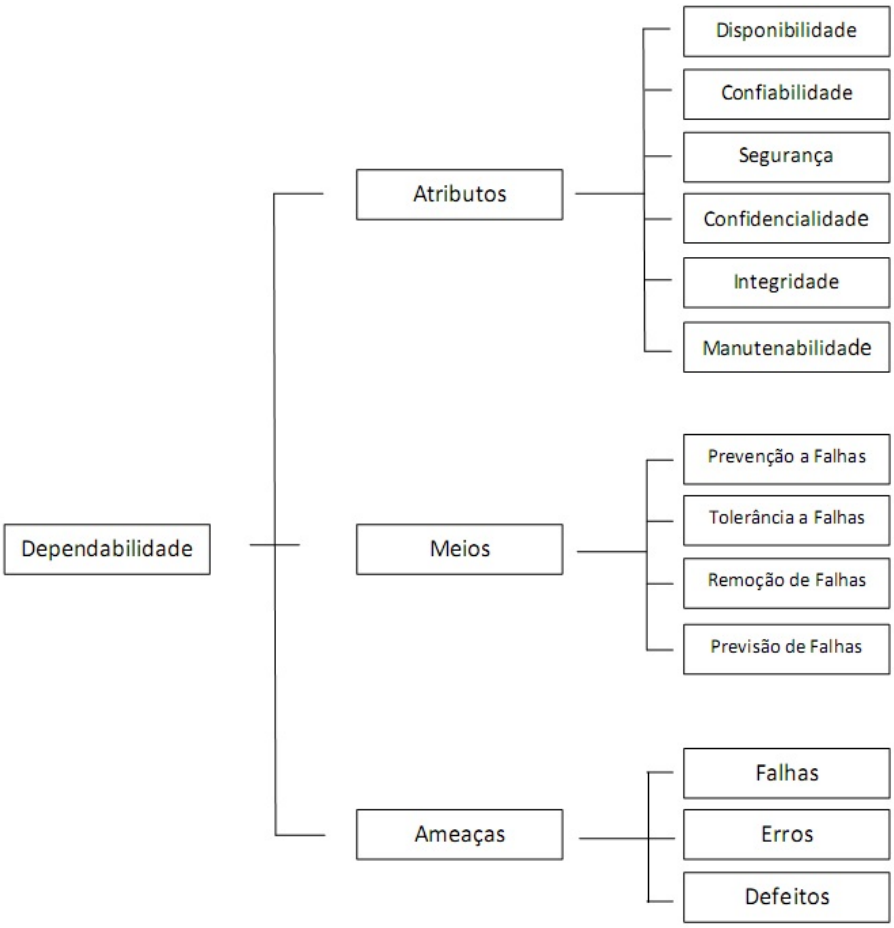


Figura 3 – Árvore de dependabilidade Adaptado de (AVIZIENIS et al., 2001)

Falhas de *software* ou quebras de *hardware* em sistemas de nuvem pode afetar a disponibilidade e, assim, a comunicação entre seus componentes, tendo um impacto sobre a dependabilidade do sistema. Desse modo, técnicas como replicação de componentes específicos em determinada arquitetura em muitos sistemas de nuvem podem garantir os aspectos de dependabilidade sendo descritos como atributos da dependabilidade.

Como mencionado, a dependabilidade de um sistema é a sua capacidade de oferecer um conjunto de serviços confiáveis que são observados por agentes externos. A falha do sistema ocorre quando o sistema não fornece sua funcionalidade especificada. A falha do sistema pode ser definida como a falha de um componente do sistema, a falha de um subsistema do sistema, ou outro sistema que interage com o sistema considerado. Desta forma, podemos considerar um indicador de uma variável aleatória $X(t)$ que representa o estado do sistema no momento t. $X(t) = 1$ representa o estado operacional do sistema e $X(t) = 0$ o estado de falha. Mais formalmente, considerando-se uma variável aleatória

T que indica o tempo necessário para atingir o estado $X(t) = 0$, uma vez que o sistema começou no estado $X(0) = 1$. T é o tempo de sistema S à falha, $F_T(t)$ a sua função de distribuição acumulada e $f_T(t)$ da respectiva função de densidade, em que:

$$\begin{aligned} F_T = 0 \; and \; \lim_{t\to\infty} F_T = 1, \\ f_T(t) = \frac{dF_T(t)}{dt}, \\ f_T(t) \geq 0 \; and \; \int_0^\infty f_T(t)dt. \end{aligned} \tag{2.1}$$

A Equação 2.2 representa a disponibilidade de um sistema expressado através da relação entre tempo médio de falha (MTTF) (Equação 2.4) e tempo médio de reparo (MTTR) (Equação 2.5). A disponibilidade calculada desta forma será um número entre zero e um. Esse valor também pode ser expresso em termos de números de noves. Por exemplo, se a disponibilidade do sistema é igual a 0,999876, isso indica que o sistema se encontra funcionando durante 99,9876% do tempo e inativo em 0,0124% do tempo observado. O número de noves da disponibilidade pode ser calculada conforme a Equação 2.3. 100 representa o nível de disponibilidade máxima que o sistema pode atingir e A representa a disponibilidade real do sistema.

$$A = \frac{uptime}{uptime + downtime} \tag{2.2}$$

$$N = 2 - log(100 - A) \tag{2.3}$$

onde,

$$MTTF = \int_0^\infty R(t)dt, \tag{2.4}$$

$$MTTR = MTTF \times \frac{UA}{A} \tag{2.5}$$

UA representa a indisponibilidade do sistema, (Equação 2.6), e A a disponibilidade do sistema *Availability* (Equação 2.2).

$$UA = 1 - A \tag{2.6}$$

Assim, o *downtime* anual (tempo de inatividade do sistema no período de um ano) pode ser calculada seguindo a Equação 2.7.

$$D = UA \times H \tag{2.7}$$

A variação da taxa de falha dos componentes de *hardware* é mostrado na Figura 4. Conhecida como curva da banheira, a figura demonstra a taxa de falhas de componentes de hardware em três fases distintas (EBELING, 2004).

Durante a primeira fase, ocorre um curto período em que a taxa de falha é bastante alta. Falhas ocorridas nesse período são decorrentes de defeitos de 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 segunda fase, as falhas ocorrem aleatoriamente. Valores de confiabilidade de equipamentos fornecidos por fabricantes aplicam-se a esse período.

Durante a fase final, a taxa de falhas cresce exponencialmente. 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.

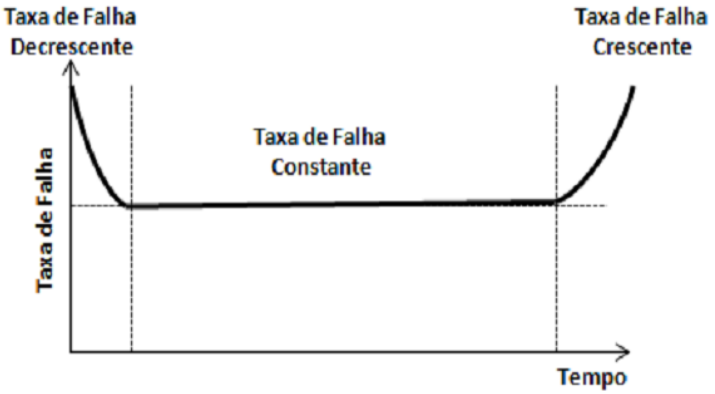


Figura 4 – Curva da Banheira (EBELING, 2004)

O estudo da disponibilidade do sistema pode ser dividida em três classes(RESNICK, 1996): disponibilidade básica, alta disponibilidade e disponibilidade contínua.

Definindo técnicas e meios de alcançar a disponibilidade desejada, é possível calcular métricas para obtenção de resultados e tomadas de decisões adequadas. Por exemplo, se os usuários estão interessados não apenas se o sistema está operacional, mas sim na capacidade que o sistema tem de oferecer ou entregar determinado serviço. Considerando, por exemplo, um sistema com dois servidores em paralelo, se os dois servidores estão operacionais, o sistema pode fornecer a sua capacidade de serviço completa. Se apenas um servidor está operacional, o sistema pode entregar apenas a metade da capacidade que o serviço pode oferecer. E quando nenhum dos servidores está operacional o sistema não pode fornecer o serviço, definindo assim a métrica de COA (HEIMANN; MITTAL; TRIVEDI, 1990):

Por exemplo, podemos considerar a capacidade computacional média de um sistema com n processadores. Seja C_i a capacidade computacional de um sistema com i processadores operacionais. Isto pode ser uma função linear simples do número de processadores, $C_i = ic_1$, ou uma função mais complexa de i, dependendo da capacidade da aplicação utilizar i processadores. A capacidade média do sistema computacional pode então ser definida como $\sum_{i=1}^{n} C_i P_i(t)$, em que $P_i(t)$ representa a probabilidade de que exatamente i processadores

estão em estado operacional no tempo t. Em contraste, a confiabilidade do sistema no tempo t será

$$R(t) = \sum_{i=m}^{n} P_i(t), \tag{2.8}$$

onde m é um número mínimo de processadores necessário para o funcionamento correto do sistema (KOREN; KRISHNA, 2010).

Em outras palavras, o COA pode ser expresso seguindo a Equação 2.9.

$$COA = \frac{\sum \forall s \pi_s \times nr(s)}{TNR}, \tag{2.9}$$

onde, $nr(s)$ é o número de recursos operacionais no estado s e TNR é o número total de recursos da infra-estrutura.

2.5 Modelagem analítica

Existem vários tipos de modelos que podem ser utilizados para a avaliação analítica de dependabilidade. Diagramas de bloco de confiabilidade, árvores de falhas, redes de Petri estocásticas e cadeias de Markov têm sido usados para modelar sistemas tolerantes a falhas e avaliar medidas dependabilidades diferentes. Estes tipos de modelo diferem de um para outro, não só na facilidade de utilização para uma aplicação em particular, mas em termos de poder de modelação (MALHOTRA, 1994).

Eles podem ser classificados em modelos combinatórios e baseados em estado (MACIEL et al., 2011b). Modelos baseados em estado podem também se referir como não combinatórios, e modelos combinatoriais podem ser identificados como modelos não baseados em estados.

2.5.1 Cadeias de Markov

Uma Cadeia de Markov é um modelo baseado em estado no qual se diz que o estado atual não depende dos estados anteriores para que se conheçam os estados seguintes, tendo sido criado para a modelagem de sistemas. Assim, com esse formalismo é possível descrever o funcionamento de um sistema por meio de um conjunto de estados e transições.

As cadeias de Markov são modelos matemáticos úteis para a descrição de análises estatísticas que possuem valores de tempo em seus parâmetros. Assim, conhecidos como processo estocástico.

Um processo estocástico X(t), t $\in$ T é um conjunto de variáveis aleatórias definidas sobre o mesmo espaço de probabilidades, indexadas pelo parâmetro de tempo (t $\in$ T) e assumindo valores no espaço de estados ($s_i \in S$) (CASSANDRAS; LAFORTUNE, 1999). Assim, se o conjunto T for discreto, ou seja, enumerável X(t), t = 1, 2, 3, ..., o processo é

dito processo de parâmetro discreto ou tempo discreto. Se T for um conjunto contínuo, tem-se um processo de parâmetro contínuo ou tempo contínuo.

O processo estocástico é classificado como um processo de Markov se, para todo $t_0 < t_1 < ... < t_n < t_{n+1}$ e para todo $X(t_0), X(t_1), X(t_2), ..., X(t_n), X(t_{n+1})$, a distribuição condicional de $X(t_{n+1})$ depender somente do último valor anterior $X(t_n)$ e não dos valores anteriores $X(t_0), X(t_1), ..., X(t_{n-1})$, isto é, para qualquer número real $X_0, X_1, X_2, ..., X_n, X_{n+1}$, $P(X_{n+1} = s_{n+1}|X_n = s_n, X_{n-1} = s_{n-1}, ..., X_0 = s_0) = P(X_{n+1} = s_{n+1}|X_n = s_n)$ (BOLCH et al., 2006).

Uma cadeia de Markov é descrita por uma sequência de varáveis aleatórias discretas, $X(t_n)$, em que t_n pode assumir um valor discreto ou contínuo, isto é, uma cadeia de Markov é um processo de Markov com um espaço de estados discretos.

As Cadeias de Markov representam o comportamento do sistema (falhas e atividades de reparo) pelos seus estados, e a ocorrência de evento é expressada pela transição do estado denominado (MACIEL et al., 2011b). As etiquetas podem ser probabilidades, taxas ou funções de distribuição. A cadeia de Markov constitui um tipo particular de processo estocástico com estados discretos e com o parâmetro de tempo podendo assumir valores contínuos ou discretos (SOUSA, 2009) (STEWART, 2009). Portanto, cadeias de Markov de tempo contínuo, as CTMC *(continuous-time Markov chains)*, possuem transições que podem ocorrer em qualquer instante do tempo, e as de cadeia de Markov de tempo discreto (DTMC) *(discrete-time Markov chains)*, possuem transições que ocorrem em tempos discretos de tempos (STEWART, 2009) (STEWART, 1994).

Sendo assim, a representação de um modelo através do formalismo de cadeia de Markov pode ser interpretada como uma máquina de estados, onde os nós (vértice de um grafo) desta representam os estados, e os arcos representam as transições entre os estados do modelo em cadeia de Markov (STEWART, 2009).

Se o modelo é discreto, a escala de tempo para a transição entre os estados do modelo pode ser de forma contínua (CTMC) ou discreta (DTMC). A transição entre os estados do modelo depende exclusivamente do estado atual deste, sem importar quais formam os estados prévios ou futuros de tal modelo. A taxa (CTMC) ou probabilidade (DTMC) de transição de estados do modelo dá-se obedecendo a uma lei exponencial ou geométrica, respectivamente (STEWART, 1994) (STEWART, 2009).

Para representar graficamente um modelo em Cadeia de Markov é feita uma associação entre os estados e em cada transição entre os estados é inserida uma taxa ao modelo de tempo contínuo (CTMC) ou probabilidade para modelos discretos (DTMC). Desta forma, um modelo em cadeia de Markov (CTMC) pode ser representado matematicamente por uma matriz de transição de estados. A probabilidade de cada estado em regime estacionário (solução de um modelo em cadeia de Markov) é a solução do sistema da Equação linear

2.11.

$$Q = \begin{pmatrix} q_{ii} & q_{ij} \\ q_{ji} & q_{jj} \end{pmatrix}, \tag{2.10}$$

$$\pi Q = 0 \tag{2.11}$$

Onde Q é a matriz de estados e π(vetor de probabilidade) é o autovetor correspondente ao autovalor unitário da matriz de transição, resultando em um vetor 0. É importante ressaltar que a soma dos elementos do vetor de probabilidade π deve ser igual a 1, ou seja, $||\pi|| = 1$ (ARAÚJO, 2009). A representação gráfica de uma cadeia de Markov é representada por um diagrama de transições. Assim, podem ser visualizados os estados, sendo representados por círculos, e as transições representadas por arcos, além das taxas e/ou probabilidades das transições.

Para as cadeias de Markov de tempo contínuo, a matriz de taxas é cada elemento não diagonal da linha i e coluna j, onde as mesmas representam a taxa de transição do estado i para o estado j do modelo. Os elementos diagonais representam o ajuste necessário para que a soma dos elementos de cada linha seja zero. As probabilidades de transição dos estados podem ser calculadas através da Equação 2.12.

$$p_{i,j}(s,t) = PX(t) = j|X(s) = i \tag{2.12}$$

A solução transiente, ou dependente do tempo, é importante quando o sistema a avaliar é dependente do tempo (SOUSA, 2009). Para modelos ergódicos, considerando tempos de execução longos, pode-se mostrar que a probabilidade dos estados converge para valores constantes (HERZOG, 2001). O comportamento transiente da cadeia de Markov nos fornece informações de desempenho e dependabilidade sobre os instantes iniciais do sistema. Assumindo-se que a probabilidade $\pi(t)$ é independente do tempo, isto é, $\pi_i = \lim_{t\to\infty} \pi_i(t)$(homogeneidade), consequentemente, $\pi'(t) = 0$, resultando nas Equações 2.11 e 2.13.

$$\sum_{i=1}^{N} \pi_i = 1 \tag{2.13}$$

A Equação 2.13 é a condição de normalização, adicionada para assegurar que a solução obtida é um único vetor de probabilidade. A Equação 2.11 tem um conjunto de soluções infinitas. Normalizando as soluções, chega-se a um único vetor de probabilidades.

Desta forma, as cadeias de Markov têm importância fundamental no processo de modelagem de sistemas redundantes e avaliação de dependabilidade nos mais variados sistemas. Por isso, modelos de Markov por recompensa estão sendo usados e a sua descrição formal pode ser vista a seguir.

Definição 1 *Um modelo de Markov por recompensa (*markov reward model *(MRM)) M é uma 3-tupla ((S,**R**,Rótulo),ρ,ι)) em que (S,**R**,Rótulo) é uma CTMC subjacente rotulado C, $\rho : S \to \Re_{\geq 0}$ é a estrutura de recompensa de estados, e $\iota : S \times S \to \Re_{\geq 0}$ é a estrutura de recompensa por impulso de tal modo que se $\boldsymbol{R}_{S,S} \geq 0$ então $\iota(s, s) = 0$.*

Uma MRM é um CTMC rotulada e aumentada com recompensa por estado e estruturas de recompensa por impulso. A estrutura de recompensa por estado é uma função ρ que atribui a cada estado $s \in S$ uma recompensa de $\rho(s)$ tal que se t unidades de tempo são gastos no estado s, uma recompensa de $\rho(s) \cdot t$ é adquirido. As recompensas que são definidas na estrutura de recompensa por estados podem ser interpretadas de várias formas. Elas podem ser consideradas como o ganho ou benefício adquirido por ficar em determinado estado e também podem ser consideradas como o custo incorrido por ficar em algum estado.

A estrutura de recompensa de impulso, por outro lado, é uma função ι que atribui a cada transição de s para s', onde $s, s' \in S$, uma recompensa $\iota(s, s')$ de tal modo que, se a transição de s para s' ocorre, uma recompensa de $\iota(s, s')$ é adquirida. Semelhante à estrutura de recompensa por estado, a estrutura de recompensa de impulso pode ser interpretada de várias formas. Uma recompensa de impulso pode ser considerada como o custo de tomar uma transição ou o ganho que é adquirida, levando em consideração a transição.

2.5.2 Redes de Petri

Rede de Petri (RdP) ou PN é representada por uma notação gráfica e matemática para representar sistemas de forma intuitiva, demonstrando características de alto nível de abstração, podendo ser aplicadas em diversas áreas, tais como desenvolvimento de *software*, engenharia, dentre outros. Segundo Peterson (PETERSON, 1977), o principal uso das redes de Petri é a modelagem de sistemas de eventos, podendo alguns deles ocorrer simultaneamente, mas há restrições sobre o concurso, a precedência ou a frequência dessas ocorrências.

A representação formal de um modelo RdP é a quíntupla $PN = (P, T, F, W, \mu_0)$, 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 : $F \to IR^+ \cup \{0\}$ é a função de atribuição de peso aos arcos;
- $\mu_0 : P \to IN$ é a função de marcação inicial, onde $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

Redes de Petri permitem a modelagem e a análise de sistemas de eventos discretos que são demasiadamente complexos para serem descritos por autômatos ou modelos de filas (REISIG, 1985). Assim, as Redes de Petri permitem a representação matemática e a análise através de modelos gráficos, fornecendo informações úteis sobre a estrutura e o comportamento dos sistemas.

O aspecto estrutural de um modelo RdP equivale a um grafo direcionado e bipartido composto por alguns elementos básicos, conhecidos como estados (lugares), ações (transições exponenciais ou imediatas), arcos (dirigido ou inibidor) e marcas (tokens).

Os lugares correspondem aos estados, as transições às ações ou eventos realizados pelo sistema e os arcos ligam os lugares às transições e vice-versa. Pode-se observar na Figura 5 como são representados graficamente os elementos que compõem uma rede de Petri.

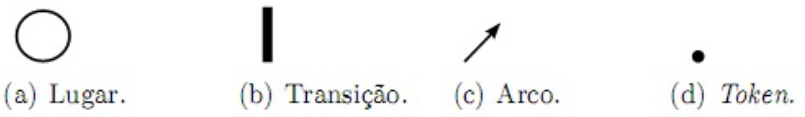


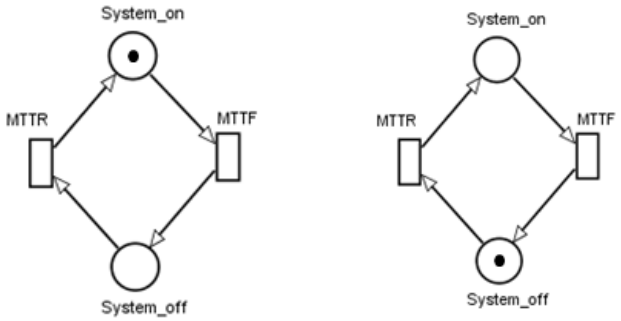
Figura 5 – Elementos da Rede de Petri

Para que seja realizada uma determinada ação em uma Rede de Petri, a mesma deve estar associada a alguma pré-condição, ou seja, existe uma relação entre os lugares e as transições que possibilitam ou não a realização de uma determinada ação. Após a realização de uma determinada ação, alguns lugares terão suas informações alteradas e podem criar uma pós-condição. Os arcos representam o fluxo das marcas pela rede e as marcas representam o estado em que o sistema se encontra em determinado momento.

A Figura 6 demonstra um exemplo de um sistema por meio de um simples modelo em Redes de Petri. É demonstrado um servidor onde o mesmo pode estar ativo ou inativo em determinado tempo. Os lugares representam um estado *System_on* (estado em que os serviços do servidor estarão disponíveis) e um estado *System_off* (estado em que há alguma quebra de serviço ou componente deixando o sistema indisponível). Nesse exemplo, o arco dirigido do lugar *System_on* para a transição MTTF (*Mean Time to Failure*) indica que, para que seja representado a quebra de um componente, é necessário que haja um token no lugar *System_on*. De forma similar, o arco dirigido do lugar *System_off* para a transição MTTR (*Mean Time to Repair*) indica que, para que o sistema seja reparado recuperando sua funcionalidade, necessita-se que haja um token no lugar servidor. A localização do token na rede indicará se o sistema estará funcionando *System_on* (Figura 6a) ou em quebra *System_off* (Figura 6b).

Assim, a partir de uma marcação M, a habilitação de determinada transição da RdP estará ativada se prioridades da rede permitirem a habilitação.

Propriedades das Redes de Petri - Com a análise do modelo de rede de Petri, podemos obter diversas propriedades. Com isso, características do sistema podem ser observadas. As propriedades podem ser subdivididas em comportamentais e estruturais.



(a) Sistema Funcionando (b) Sistema em Quebra

Figura 6 – Exemplo de uma Rede de Petri

As propriedades comportamentais dependem apenas da marcação inicial da rede de Petri. Assim, serão subscritas as principais propriedades baseando-se em(MURATA, 1989).

- **Alcançabilidade** - A propriedade de alcançabilidade indica a possibilidade de atingir uma determinada marcação pelo disparo de um número finito de transições a partir de uma marcação inicial. Uma marcação M_0 é dita alcançável a partir de M_i, se existir uma sequência de disparo que transforme M_0 em M_i. A sequência de disparo é denotada pelo conjunto $\sigma = t_1, t_2, ..., t_n$. Nesse caso, M_i é alcançável a partir de M_0 por σ. Onde σ é formalmente descrito por $M_0[\sigma > M_i$.

- **Limitação** - Uma rede é dita k-limitada se todos os seus lugares forem limitados, ou seja, o número de tokens em cada lugar não deve ultrapassar um número finito k, para qualquer marcação alcançável a partir de M_0.

- **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.

- **Vivacidade ou *liveness*** - Uma rede é dita *live* se não importam quais marcações sejam alcançáveis a partir de uma marcação inicial m_0, se for possível disparar qualquer transição através do disparo de alguma sequência de transições $L(M_0)$.

- **Cobertura** - A propriedade de cobertura está associada ao conceito de alcançabilidade e *liveness*. Uma marcação M_i é coberta se existir uma marcação $M_j \neq M_i$, tal que $M_j \geq M_i$.

- **Reversibilidade** - Uma rede é reversível se, para cada marcação M em $R(M_0)$, M_0, é alcançada a partir de M. Assim, a marcação inicial pode ser novamente alcançada.

Já as propriedades estruturais não dependem da marcação. Assim, serão subscritas as principais propriedades baseando-se em (KARTSON et al., 1994).

- **Estruturalmente Limitada** - Uma rede de Petri é classificada como estruturalmente limitada se for limitada para qualquer marcação inicial, ou seja, se o número de tokens é limitado para qualquer marcação inicial.

- **Conservação** - A conservação é uma importante propriedade das redes de Petri, pois permite a verificação da não destruição de recursos através da conservação de tokens. Assim, com o disparo de qualquer transição o número de marcações não é alterado.

- **Repetitividade** - Uma rede é classificada como repetitiva se, para uma marcação e uma sequência de transições disparáveis, todas as transições dessa rede são disparadas ilimitadamente.

- **Consistência** - Uma rede é classificada como consistente se, a partir de uma sequência de transições disparáveis partindo-se de uma marcação inicial M_0, ele retorna a M_0, porém todas as transições da rede são disparadas pelo menos uma vez.

Redes de Petri Estocásticas - As redes de Petri estocásticas (SPNs) possuem transições que podem ser imediatas e temporizadas. As transições temporizadas possuem um atraso exponencialmente distribuído, enquanto que as transições imediatas possuem tempo de retardo associado igual a zero. As redes de Petri estocásticas permitem a modelagem e a análise probabilística de sistemas. A propriedade de ausência de memória da distribuição exponencial no atraso dos disparos implica no fato das SPNs serem isomórficas às cadeias de Markov de tempo contínuo (*continuous time Markov chain*, CTMCs), provendo então medidas de desempenho e dependabilidade (SOUSA, 2009).

Em 1982, (MOLLOY, 1982) apresentou as redes de Petri estocásticas (*Stochastic Petri Nets - SPN*) como uma técnica capaz de especificar sistemas e apresentar uma análise probabilística dos mesmos. Elas surgiram a partir do formalismo de Redes de Petri Temporizadas (TPN), que têm como sua principal característica a associação de um atraso fixo para cada transição do modelo. (MOLLOY, 1982) definiu que todas as transições em uma SPN eram temporizadas (timed) e que possuíam um retardo exponencialmente distribuído.

As SPNs oferecem possibilidade de unir a habilidade do formalismo de Redes de Petri para descrever sincronização e concorrência com um modelo estocástico, permitindo a descrição de um comportamento dinâmico na modelagem de desempenho (*performance*) e dependabilidade (*dependability*) de sistemas (MARSAN; BALBO; CONTE, 1986).

Uma SPN é definida (GERMAN et al., 1995) 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, $P \cap T = \emptyset$;

- $I \in (\mathbb{N}^n \to \mathbb{N})^{n\ddot{O}m}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \to \mathbb{N})^n\ddot{O}m$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações);
- $H \in (\mathbb{N}^n \to \mathbb{N})^n\ddot{O}m$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações);
- $\Pi \in \mathbb{N}^n$ 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 à marcação do lugar a cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, Markdep, Policy, Concurrency, W)^m$ compreende o conjunto de atributos associados às transições, onde:

- $Dist \in \mathbb{N}^n \to F$ é uma possível função de distribuição de probabilidade associada ao tempo de uma transição (esta distribuição pode ser dependente de marcação) (o domínio de F é $[0, \infty)$;

- $Markdep \in \{constante, enabdep\}$, onde a distribuição de probabilidade associada ao tempo de uma transição pode ser independente (*constante*) ou dependente de marcação (*enabdep*- a distribuição depende da condição de habilitação atual);

- $Policy \in \{prd, prs\}$ define a política de memória adotada pela transição (*prd-preemptive repeat different*, valor padrão, de significado idêntico à *race enabling policy; prs-preemptive resume*, corresponde ao *age memory policy*);

- $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica *single server* e is representa a semântica *infinity server*.

- $W : T \to IR^+ \cup \{0\}$ é a função peso, que representa o peso (w_t) de transições imediatas e a taxa λ_t de transições temporizadas, onde:

$$\pi(t) = \begin{cases} \geq 1, & \text{se } t \text{ é uma transição imediata;} \\ 0, & \text{caso contrário.} \end{cases}$$

Se t é uma transição temporizada, então λ_t será o valor do parâmetro da função densidade probabilidade exponencial.

Se t é uma transição imediata, então W_t será um peso, que é usado para o cálculo das probabilidades de disparo das transições imediatas em conflitos.

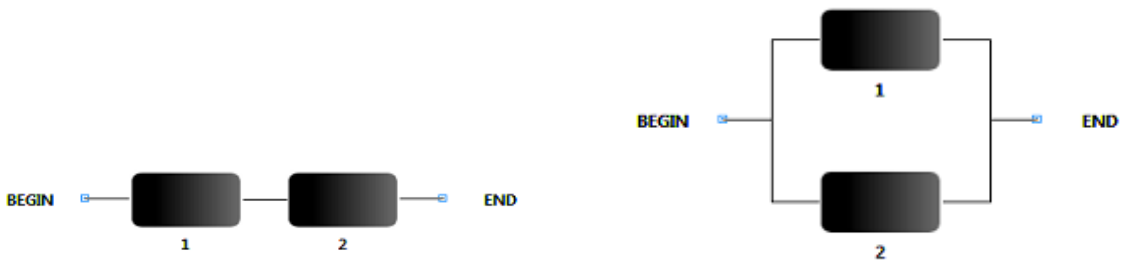
Os arcos inibidores são usados para prevenir transições de serem habilitadas quando certa condição é verdadeira.

Nas SPNs, quando múltiplas transições estão habilitadas em uma mesma marcação m, a transição que tem a maior probabilidade de disparo é a transição que possuir menor tempo de atraso associado a ela (BALBO, 2001). Assim como as redes de Petri, quando uma transição de uma rede de Petri estocástica é disparada, uma nova marcação pode ser gerada em um outro lugar.

2.5.3 Diagramas de Bloco de Confiabilidade

Diagrama de bloco de confiabilidade (RBD) foi a primeira técnica apresentada para determinar a confiabilidade de um sistema. Em um modelo de diagrama de blocos, os componentes são representados como blocos e são combinados com outros blocos, ou seja, combinados com outros componentes, em série, paralelo, ou configurações *k-out-of-n* (TRIVEDI et al., 1996). Assim, RBD's possibilitam a modelagem de sistema, fornecendo uma visualização gráfica dos componentes do sistema. O modelo RBD fornece uma iteração lógica entre componentes do sistema, definindo quais combinações ativas definem a funcionalidade do sistema. Com essa técnica, é possível também calcular as métricas de dependabilidade, tais como a disponibilidade e manutenabilidade.

A Figura 7 mostra dois exemplos de modelos RBD's, onde os blocos são organizados em série (Figura 7a) e paralelo (Figura 7b). No modelo em série, se um único componente falhar, o sistema para de prover o serviço entrando no modo de falha.



(a) Série (b) Paralelo

Figura 7 – Diagrama de Blocos de Confiabilidade

A confiabilidade de dois blocos conectados em série (ver Figura 7a) pode ser obtida através da Equação 2.14.

$$R_s(t) = R_1(t) \times R_2(t) \tag{2.14}$$

Onde:
$R_1(t)$ é a confiabilidade do bloco 1.
$R_2(t)$ é a confiabilidade do bloco 2.

De uma forma geral, considerando um sistema com n elementos, a confiabilidade do modelo em série pode ser obtida através da Equação 2.15.

$$R_s(t) = \prod_{i=1}^{n} R_i(t) \tag{2.15}$$

Onde:

$R_i(t)$ é a confiabilidade do bloco b_i.

A confiabilidade de dois blocos conectados em paralelo (ver Figura 7b) pode ser obtida através da Equação 2.16.

$$R_p(t) = 1 - \prod_{i=1}^{2}(1 - R_i(t)) \tag{2.16}$$

Para esse tipo de sistemas temos que pelo menos um componente deve ser operacional para que todo sistema esteja funcionando. De uma forma geral, considerando n componentes, a confiabilidade do sistema pode ser obtida através da Equação 2.17.

$$R_p(t) = 1 - \prod_{i=1}^{n}(1 - R_i(t)) \tag{2.17}$$

onde:

$R_i(t)$ é a confiabilidade do bloco b_i.

Outra representação dos RBDs são os blocos $k-out-of-n$ que representam estruturas em que o subsistema pode funcionar se k componentes estão em funcionamento (XIE; DAI; POH, 2004). Vamos tomar como exemplo uma infraestrutura com dez componentes, desses componentes necessita-se que ao menos quatro estejam em funcionamento para prover o serviço esperado, temos, então, uma estrutura de $4 - out - of - 10$ (ou 4 de 10). As estruturas em série-paralelo são casos especiais de estruturas $k - ou - of - n$, uma estrutura em série é uma n-out-of-n e uma estrutura em paralelo é uma estrutura $1 - out - of - n$ (SAHNER; TRIVEDI, 1987). Para a definição matemática da confiabilidade deste arranjo lógico, é necessária a definição da variável aleatória discreta X, que define o número de blocos que não apresenta falhas, em um determinado intervalo de tempo (ver Equação 2.18). Os eventos probabilísticos de dependabilidade são independentes para cada bloco da configuração $kden$ e todos os n blocos possuem a mesma taxa de falha (SAHNER; TRIVEDI, 1987) (MACIEL et al., 2011b).

$$R_{k-out-of-n}(t) = \sum_{i=k}^{n} P(X = i) \tag{2.18}$$

Modelos RBD's são utilizados em sistemas que contêm módulos independentes, onde cada um pode ser facilmente representado por um bloco de confiabilidade. Assim, havendo a necessidade de modelar sistemas complexos, onde se exige a adição de redundância em módulos do sistema, o usuário tem que recorrer a técnicas de modelagem hierárquica,

fazendo uso, em conjunto, de modelos como SPN, CTMC e RBD, na tentativa de obter resultados mais expressivos.

2.6 Considerações Finais

Este capítulo apresentou a fundamentação teórica básica para o melhor entendimento do leitor sobre os principais tópicos que compõem esta tese. Os conceitos básicos sobre avaliação de dependabilidade e desempenho, bem como modelagem analítica, permite entender a aplicação de tais conceitos e tem como objetivo, introduzir, a cerca do que vem a ser trabalhado nesta tese.

3 TRABALHOS RELACIONADOS

Este capítulo descreve os trabalhos encontrados durante a revisão da literatura. Podemos subdividir os trabalhos relacionados nas categorias descritas a seguir: Dependabilidade, desempenho e custo das infraestruturas de computação em nuvem. Os trabalhos apresentados estão inseridos no contexto de planejamento e avaliação de sistemas de computação em nuvem, avaliação de desempenho de serviços de vídeo *streaming* e custos financeiros em implantação de nuvens públicas e/ou privadas. Essa seção tem por objetivo fornecer uma visão geral dos trabalhos publicados sobre o tema em questão, desta forma, ao final dessa seção, é possível encontrar uma conclusão efetiva mostrando a relevância do trabalho aqui proposto com os demais apresentados.

3.1 Dependabilidade

Em (CHUOB; POKHAREL; PARK, 2011) Chuob *et al.*, os autores realizaram um experimento para uma plataforma de Governo Eletrônico em uma infraestrutura de computação em nuvem baseada na plataforma *Eucalyptus*, observando o comportamento do ambiente,a fim de alcançar parâmetros de falha e reparo, com o intuito de obter resultados de disponibilidade através de modelagem hierárquica. A partir das observações, os autores consideraram que os componentes Cluster Controller e Node Controller são os mais críticos da infraestrutura da nuvem, portanto, a partir desses dois componentes, é possível obter melhores resultados de disponibilidade.

Em (GHOSH et al., 2014) Ghosh *et al.*, o autor realizou análises de disponibilidade em uma infraestrutura na nuvem, onde máquinas físicas estão agrupadas em três tipos de conjuntos de provisionamento de recursos: hot (em execução), warm (ligado, mas não operante) e cold (desligado). Essa divisão foi realizada com base no consumo de energia e nas características de aguardo no provisionamento de recursos. Dessa maneira, é feita uma abordagem por modelagem estocástica para análise de dependabilidade desses sistemas na nuvem de provisionamento de IaaS nos três modos acima citados, por meio de cadeia de Markov.

Em (MUÑOZ et al., 2013) Munoz *et al.* os autores propõem uma arquitetura para serviços que possuem a capacidade de resistir à falhas em nuvens híbridas. Os autores monitoram a disponibilidade de provedores de nuvens distintas e fornecem degradação suave para a adaptação às interrupções ou falta de resposta desses prestadores de serviços.

Já em (CUOMO et al., 2013) Cuomo *et al.*, os autores preveem mecanismos de acompanhamento e previsão de qualidade do serviço e métricas de SLA em nuvens privadas. Eles preveem disponibilidade de recursos através da medição do tempo médio de falha (MTTF) e tempo de reparo (MTTR). O monitoramento remoto por meio de *heartbeat* e tempo de

boot da VM são os principais mecanismos para o monitoramento dos valores de MTTF e MTTR. Vale a pena notar que as VMs são o único recurso considerado nesse trabalho e modelos analíticos ou de simulação não são utilizados.

Em (SOUSA et al., 2014) Sousa et al., os autores abordaram uma estratégia de modelagem baseada em um modelagem hierárquica e heterogênea para o planejamento de infraestruturas de nuvem. A estratégia de modelagem permite a seleção de infraestruturas de nuvem de acordo com as exigências de dependabilidade e custo. Um estudo de caso baseado em ambientes virtuais de aprendizagem hospedado na plataforma de nuvem Eucalyptus é adotado para demonstrar a viabilidade da estratégia e dos modelos propostos.

Os pesquisadores no trabalho de (BRILHANTE et al., 2014) Brilhante *et al.*, avaliaram a disponibilidade em uma infraestrutura de nuvem privada. Os autores fizeram uso de modelagem estocástica por meio de Redes de Petri e validaram a proposta por meio de técnicas de injeção de falhas em um ambiente real considerando ciclo de vida das VMs. Posteriormente, os autores comparam os resultados do experimento com o modelo proposto.

O trabalho de (COSTA et al., 2015) Costa *et al.* é similar com o proposto por Brilhante *et al.*; esse trabalho se concentra em modelagem hierárquica em um sistema de MBaaS OpenMobster com foco em dois cenários partindo de uma arquitetura básica: sem processo de recuperação em caso de falha, e o outro, com processo de recuperação automática. Os modelos construídos foram validados através de experimento em ambiente real por meio de técnicas de injeção de falha. Os autores consideraram *hardware*, sistema operacional e o serviço (MBaaS OpenMobster).

Similar aos trabalhos descritos em (COSTA et al., 2015) e (BRILHANTE et al., 2014), o trabalho proposto por (BEZERRA et al., 2014) Bezerra *et al.* e (MELO et al., 2014) De Melo *et al.*, avaliam a disponibilidade de um serviço de vídeo *streaming* implementado em uma infraestrutura de nuvem privada. Os autores também fazem uso de técnicas de injeção de falha em um ambiente de testes real. Os autores, fazem uso ainda, de técnicas de análise de sensibilidade paramétrica para identificar gargalos no ambiente e assim propor melhorias.

Em (GUIMARAES et al., 2011) Guimaraes *et al.*, os autores investigam a disponibilidade no cenário de redes de computadores, fazendo uso de mecanismos de redundância para alcançar a disponibilidade desejada. Para isso, os autores fazem uso de Redes de Petri estocástica como abordagem de modelagem permitindo uma avaliação analítica de cenários complexos. Além disso, os autores fazem uso da técnica de RI (*reliability importance*) com o objetivo de encontrar os componentes mais importantes do sistema, justificando, assim, o uso de redundância em pontos estratégicos no cenário de rede.

Esta tese apresenta estudos de disponibilidade em modelos que são semelhantes aos encontrados em (BEZERRA et al., 2014), a fim de combinar técnicas de disponibilidade e análise de sensibilidade, são apresentados estudos de desempenho e custo ao estudo em questão. Outra contribuição original da obra em curso é a proposição de uma abordagem

geral para o planejamento de infraestrutura de nuvem considerando serviço de vídeo sob demanda.

3.2 Desempenho

Técnicas para avaliação de desempenho de sistemas podem ser realizadas através de métodos analíticos, numéricos e por medição (LILJA, 2005). Para a escolha do método de avaliação a ser utilizado, deve ser levado em consideração alguns fatores: O quão novo é o sistema a ser avaliado, ou o quanto de tempo e dinheiro você tem para investir. Quando há o *design* de um novo sistema e possível usar o método analítico ou numérico. Quando se tem tempo necessário e dinheiro suficiente para investir, a técnica de medição pode ser adotada (JAIN, 2008).

Xiong e Perros os autores apresentam modelos em redes de filas para a avaliação de desempenho e com isso ter subsídios necessários para se planejar infraestruturas de nuvens computacionais (XIONG; PERROS, 2009). 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 por um servidor web e direcionadas para infraestrutura computacional de nuvem. O modelo concebido proporciona o planejamento de uma infraestrutura de nuvem capaz de atender a um número de solicitações distintas com um tempo de resposta esperado. Esse modelo, no entanto, não proporciona o dimensionamento da quantidade de máquinas virtuais necessárias para atender a um número de clientes específicos levando em consideração custo por aluguel de VM com um determinado tempo de resposta. Da mesma maneira, esse modelo não proporciona o dimensionamento de máquinas físicas que podem ser utilizadas dependendo da carga de trabalho aplicada.

A computação em nuvem foi utilizada, também, para que fosse atingido um critério de Quality of Service - QOS através do provimento dinâmico de recursos da nuvem, ou seja, ajustando a quantidade de recursos dinamicamente para se adequarem às variações na entrada de vídeos no sistema (YANG et al., 2015). Foi utilizada, da mesma maneira, uma política de que o primeiro vídeo a chegar seria o primeiro vídeo a ser transcodificado (First come, first served - FCFS).

Ghosh *et al.* propõem uma abordagem de modelagem composta para tratar de questões de performabilidade de infraestruturas como serviço (IaaS) em nuvens. Os autores utilizam modelos de desempenho e disponibilidade composto em um único modelo, por isso, métricas como probabilidade de rejeição de *jobs* e atraso de tempo de resposta são obtidos como resultado final. A análise de sensibilidade, através da variação paramétrica permite-lhes quantificar os efeitos de variações na carga de trabalho, taxas de falha e a capacidade do sistema (número de máquinas físicas) sobre a qualidade do serviço de nuvem IaaS. Os autores quantificam também a redução do espaço de estados conseguida pelo modelo

composto quando comparado com um modelo monolítico, que se reflete no menor tempo necessário, assim como solução de armazenamento de memória principal e o espaço (GHOSH et al., 2010).

A diferença do trabalho apresentado por (GHOSH et al., 2013) é semelhante ao proposto pelos autores em estudos anteriores (GHOSH et al., 2010), mas concentra-se em modelos de desempenho puro. Eles usam uma abordagem de multinível de interação de submodelos estocásticos, na qual a solução global do modelo é obtido de forma iterativa em relação às soluções do submodelo individual. Os autores mencionam a necessidade de uma análise de sensibilidade formal neste modelo composto. O grande número de parâmetros traz a necessidade de determinar os parâmetros mais importantes, além de revelar estrangulamentos no sistema.

Garcia, Kalva e Furht (GARCIA; KALVA; FURHT, 2010) analisam uma nuvem computacional baseado no *Hadoop* é usado como auxílio na transcodificação de conteúdo de mídia. O cenário avaliado leva em considerção um cenário de *live streaming* via HTTP que suporta *streaming* de audio ou vídeo para diferentes arquiteturas computacionais (celulares, tablets, computadores, etc.), bem como a transferência de vídeo sob demanda. Os autores levam em consideração o desempenho do servidor no processo de transcodificação e transmissão. Outra métrica que os autores levam em consideração é o *delay* relacionado ao tempo de transmissão em três variações: divisão do arquivo de mídia em partes iguais a quantidade de nós para transcodificação; divisão do arquivo de mídia em grandes partes (tamanhos referentes a capítulos de DVD) e divisão do arquivo de mídia em segmentos de 10s.

Em (QIU et al., 2012) Qiu *et al.*, os autores investigam a otimização de serviços de VoD em uma nuvem híbrida, que consiste em servidores locais e data centers em nuvem geograficamente distribuídas. Os autores propõe um algoritmo dinâmico com base na teoria de otimização de Lyapunov para replicar vídeos na nuvem híbrida e para distribuir as solicitações dos usuários. Dessa maneira, minimiza-se o que minimiza o custo operacional de longo prazo do prestador de serviços VoD sob restrições de qualidade de serviço. Eles demonstraram, com base em uma análise teórica preliminar, que o algoritmo se aproxima da solução de otimização alcançada por um mecanismo com informações conhecidas nos futuros intervalos de tempo T através de um pequeno intervalo constante. Os autores demonstram, por meio de simulações, o seu desempenho em cenários realistas. Os autores ainda estão trabalhando em um protótipo real de um sistema de VoD para analisar critérios de desempenho e custo em sistema real.

Os trabalhos de Sousa *et al.* (SOUSA et al., 2014) e (SOUSA et al., 2014), apresentam abordagens similares. O foco para o trabalho apresentado em (SOUSA et al., 2014) está inserido no contexto de modelos de desempenho de uma infraestrutura de nuvem privada. Os autores avaliam cenários específicos por meio de modelos, formalismos matemáticos e expressões matemáticas. Estes modelos são avaliados para o cálculo de métricas de desempenho e custo.

Em Matos *et al.* (MATOS et al., 2016) os autores modelaram uma aplicação que combina a utilização de diversos *Web Services* em CTMC. O modelo permite avaliar a confiabilidade, custo e o tempo médio de resposta desse tipo de aplicação, considerando a escolha dos diferentes provedores de *Web Service*. Portanto, uma solução otimizada deve considerar a escolha de *Web Services* que minimizem a não confiabilidade, considerando também o tempo de resposta. Soluções otimizadas foram encontradas através da meta-heurística GRASP. Além disso, também foi proposta uma versão do GRASP integrada com análise de sensibilidade com objetivo de reduzir o tempo necessário para encontrar uma solução.

Ghosh, Naik e Trivedi (GHOSH; NAIK; TRIVEDI, 2011), propõem uma estratégia de modelagem de desempenho composta de submodelos baseados em cadeias de Markov. Estes submodelos foram concebidos para representação das atividades necessárias, com atenção ao atendimento das requisições dos usuários. Os submodelos podem ser divididos em: modelo de decisão do fornecimento do recurso, modelo de instanciação e fornecimento de máquina virtual e, também, modelo de execução da máquina virtual. Os modelos de decisão do fornecimento do recurso e o modelo de execução da máquina virtual são representados por um submodelo cada, enquanto o modelo de instanciação e fornecimento de máquina virtual é representado por três submodelos correspondentes às máquinas físicas dos grupos de representação de técnicas de replicação (*hot*, *warm* e *cold*). Os submodelos concebidos foram combinados para o dimensionamento do número de máquinas físicas nos três grupos, a fim de atender às demandas dos usuários com os tempos de resposta e os custos requeridos. Essa estratégia de modelagem de desempenho proposta por Ghosh, Naik e Trivedi (GHOSH; NAIK; TRIVEDI, 2011) permite a representação de nuvens computacionais com diferentes dimensões sem que haja um aumento significativo da complexidade dos modelos propostos.

3.3 Custo

Como mencionado nos capítulos anteriores, uma das características da computação em nuvem é a possibilidade de provisionamento de recursos de forma escalável, essa capacidade de fornecer a alocação dinâmica de recursos conforme a necessidade dos seus clientes, cobrando de acordo com a utilização destes recursos, faz com que estudos de custo sejam realizados, a fim de planejar determinada infraestrutura que atenda à demanda com um custo relativamente baixo.

Ribas *et al.* (RIBAS et al., 2015) apresentam uma Rede de Petri Colorida (CPN) para simular a utilização de instâncias *Spot* para prover instâncias elásticas. Instâncias *Spot* não possuem um preço fixo por hora. Seu custo muda de acordo com a oferta e demanda da nuvem pública. O cliente da nuvem oferece um preço da instância; se o preço oferecido pelo provedor for inferior, a instância será criada, caso o preço seja maior que a oferta do cliente, a instância é terminada. Os autores identificaram que instâncias *Spot* podem

ajudar a reduzir o custo quando comparadas com instâncias sob demanda e reservadas. No entanto, esse trabalho não leva em consideração requisições de usuários e SLAs que definam a demanda da aplicação por mais recursos.

Chaisiri *et al.*, propõem um algoritmo para otimização dos custos de prestação do serviço de VMs em nuvens públicas. Os autores formulam um modelo de programação estocástica que leva em conta a procura de VMs e os respectivos custos, entretanto, os autores não avaliam métricas específicas de desempenho, confiabilidade e disponibilidade em relação ao serviço prestado na nuvem (CHAISIRI et al., 2012).

Li *et al.* propõe um conjunto de modelos de custo baseados em expressões matemáticas para o cálculo dos gastos com servidores, softwares, suporte da manutenção, equipamento de rede, sistema energético, sistema de refrigeração, instalações e imóvel. O trabalho de Li *et al.* 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 apresenta insumos para o planejamento do custo da nuvem computacional (LI et al., 2009).

De forma semelhante ao trabalho apresentado em (LI et al., 2009), esta tese propõe expressões matemáticas para o cálculo de gastos com a aquisição de equipamentos para construção de infraestruturas de nuvem privadas, considerando a capacidade de oferecer determinado serviço e expressões matemáticas para o cálculo de gastos com o provisionamento de VM em uma nuvem pública, tendo em vista o número de acesso simultâneo e tempo de resposta. Além disso, 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.

3.4 Comparação entre os trabalhos relacionados mais relevantes

A Tabela 1 resume os principais trabalhos relacionados a esta tese que foram mencionados neste capítulo, estabelecendo uma comparação entre eles e a tese em relação a quatro temas: modelos analíticos, métricas utilizadas, sistema de VoD e modelo de otimização.

Todos os trabalhos apresentados estão inseridos de alguma maneira no contexto de computação em nuvem. Os trabalhos de (SOUSA et al., 2014); (SOUSA et al., 2014) e (MATOS et al., 2016) apresentam características próprias com foco em disponibilidade e desempenho. Esses trabalhos fazem uso de técnicas específicas de modelagem de sistema, as quais algumas foram as mesmas adotadas nesta tese.

Já os trabalhos de (MELO et al., 2014); (BEZERRA et al., 2014); (YANG et al., 2015) e (QIU et al., 2012) estão inseridas no contexto de sistemas de VoD. Esses trabalhos fazem uso de técnicas como experimentos em ambiente controlado para obter resultado. Em outros, fazem uso de algum modelo analítico e os validam com resultados dos experimentos apresentados.

Tabela 1 – Comparação entre os trabalhos relacionados mais relevantes

Autores	Modelos analíticos	Métricas	VoD	Otimização
(QIU et al., 2012)	Não	Desempenho	Sim	Sim
(CHAISIRI et al., 2012)	Não	Custo	Não	Sim
(GHOSH et al., 2013)	CTMC	Desempenho	Não	Não
(MELO et al., 2014)	RBD e CTMC	Disponibilidade	Sim	Não
(GHOSH et al., 2014)	SRN	Disponibilidade	Não	Não
(SOUSA et al., 2014)	RBD e SPN	Disponibilidade, Custo	Não	Sim
(SOUSA et al., 2014)	SPN	Desempenho, Custo	Não	Sim
(BEZERRA et al., 2014)	RBD e CTMC	Disponibilidade	Sim	Não
(RIBAS et al., 2015)	CPN	Custo	Não	Não
(YANG et al., 2015)	Não	Desempenho	Sim	Não
(MATOS et al., 2016)	CTMC	Desempenho	Não	Sim
Esta Tese	CTMC, SPN, RBD	Desemp., COA, Disp., Custo	Sim	Sim

Dessa forma, esse trabalho torna-se relevante por reunir diversas características desses trabalhos, apresentando uma evolução para o planejamento de infraestruturas na nuvem com uma maior quantidade de características oferecidas por ajuste dinâmico do tamanho da infraestrutura ativa. Da mesma maneira, e por abordar de forma consolidada os temas que foram expostos separadamente em cada trabalho. Esta pesquisa apresenta modelos SPN que representem o comportamento de desempenho de sistemas para a transcodificação em nuvem e modelos matemáticos cuja característica é calcular a disponibilidade orientada à capacidade e custo pelo aluguel de nuvens públicas ou aquisição de máquinas para construção de nuvens privadas. Assim, os modelos serão utilizados para identificar configurações otimizadas a fim de atender diferentes cargas de trabalho restringidas por requisitos mínimos de desempenho.

3.5 Considerações finais

Este capítulo apresentou os principais trabalhos correlatos ao estudo proposto. Embora existam vários trabalhos na literatura que proporcionam a avaliação de desempenho e dependabilidade de sistemas de VoD em nuvens computacionais por meio de modelos analíticos ou ambiente real de experimentação, nenhum desses trabalhos tem seu foco na avaliação de aspectos de desempenho, dependabilidade e custo através de modelos analíticos e expressões matemáticas conjuntas. Alguns trabalhos apresentam uma estratégia de modelagem hierárquica e heterogênea para avaliação de dependabilidade de nuvens computacionais, mas poucos trabalhos avaliam a questão do mecanismo de auto escalonamento de VMs e custos relacionados a instâncias reservadas ou sob demanda em uma nuvem computacional. Embora existam trabalhos que apresentem uma ferramenta

para avaliação de dependabilidade ou de custo de nuvens computacionais, nenhum trabalho proporciona uma metodologia composta por modelos de representação, otimização e custo que atendam aos critérios aqui mencionados.

4 METODOLOGIA

Este capítulo apresenta a metodologia de apoio para o planejamento de infraestruturas de computação em nuvem tomando como base uma infraestrutura capaz de prover serviços de *VoD streaming* que atendam a requisitos como, disponibilidade, desempenho e custo.

4.1 Visão geral

A metodologia de apoio utilizada nesta tese visa avaliar infraestruturas de computação em nuvem, gerando boas combinações de componentes de um sistema de *cloud* a fim de planejar uma melhor infraestrutura de serviços de *VoD streaming*. Essa avaliação leva em consideração fatores como COA, tempo para publicação de um vídeo na *web* e custo com aquisição de máquinas físicas, se o planejamento envolve uma infraestrutura de nuvem privada, ou custos com aquisição de máquinas virtuais, se o planejamento envolve uma infraestrutura de nuvem pública. A metodologia de apoio utilizada nesta tese consiste em três macro atividades: **Estudo preliminar**, **Avaliação** e **Otimização**. O **estudo preliminar** está dividida em três atividades: estudo do sistema, definição de parâmetros e métricas, assim como geração de modelos. Já a **avaliação** está dividida em: geração de modelos, obter valores de entrada para os modelos e avaliação. Concluindo, o processo de **otimização** consiste na seleção de cenários e análise dos resultados.

O fluxograma da Figura 8 representa de maneira visual a metodologia de apoio adotada no planejamento de infraestruturas de computação em nuvem para serviços de *VoD*. Os retângulos representam cada etapa da metodologia de apoio que foi seguida obedecendo à ordem de execução apontada pelas setas. Somente quando uma etapa é finalizada, o avaliador segue para a próxima. O losango representa uma etapa que pode seguir por dois caminhos diferentes, dependendo do resultado obtido. Nesse caso, a análise segue para a próxima etapa caso os modelos e os resultados sejam satisfatórios, não apresentem erros no modelo ou resultados incomum, ou voltam para etapa anterior para ajustes caso não sejam. Os círculos tracejados representam as possíveis instanciações (estratégias, ações, análises, etc.) de cada etapa da metodologia. Vale destacar que as possíveis instanciações que são apresentadas no fluxograma significam que cada uma delas foi efetivamente adotada em algum momento do estudo, mesmo podendo haver outras possíveis instanciações que poderiam ser adotadas para cumprir cada etapa. As setas horizontais tracejadas apontam para as possíveis instanciações adotadas em cada etapa.

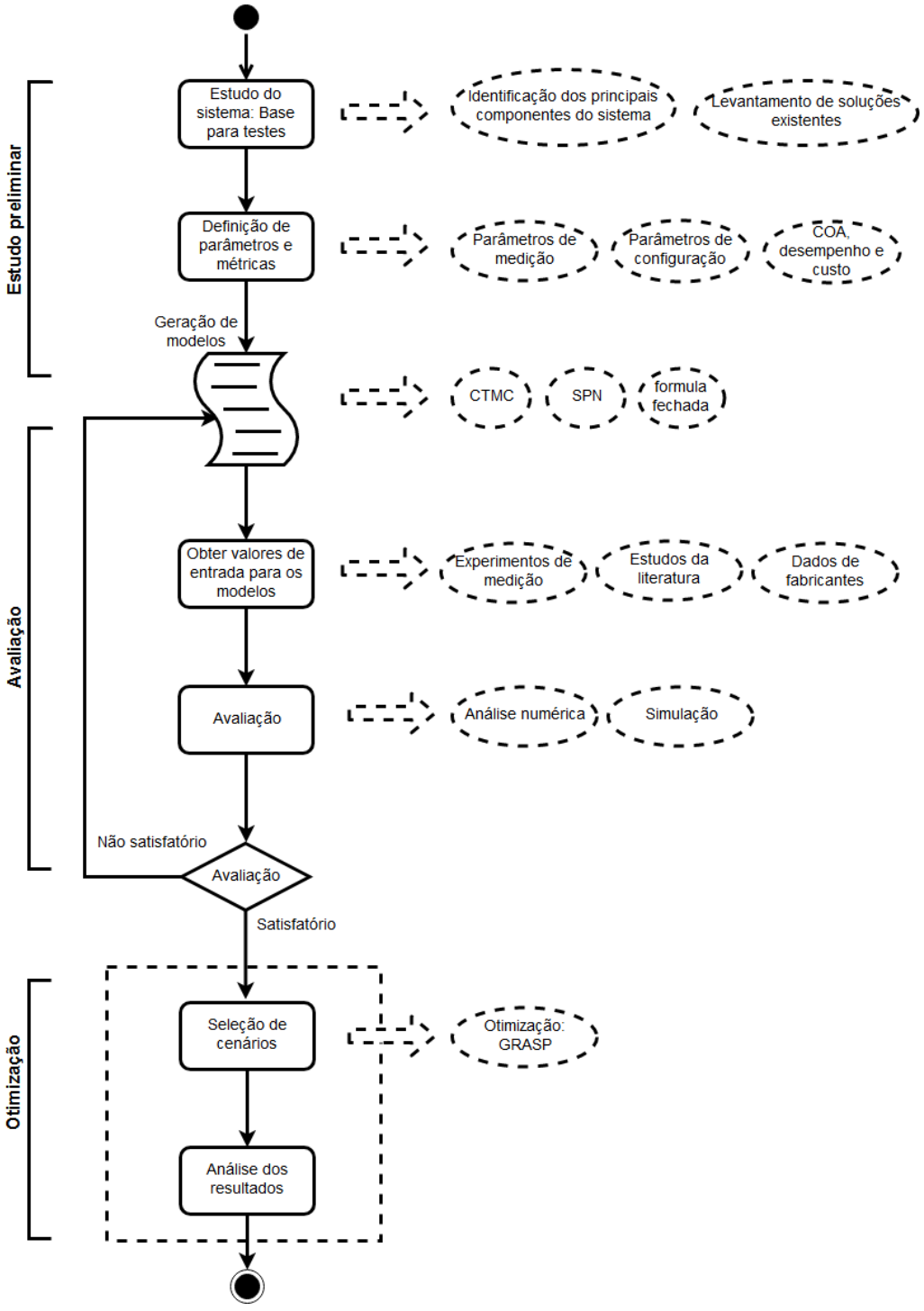


Figura 8 – Metodologia de apoio para o planejamento de nuvem para serviços de *VoD*

4.2 Estudo preliminar

Esta seção apresenta as três atividades que constituem o estudo preliminar: estudo do sistema, definição de parâmetros e métricas, e geração de modelos. O objetivo é apresentar os insumos necessários para a implementação da etapa posterior de avaliação

Estudo do sistema - base para testes: para planejar uma infraestrutura de nuvem, que atendam a determinados requisitos de um ambiente de *VoD*, é necessário ter um

entendimento do funcionamento do sistema, identificando os seus principais componentes, efetuando um levantamento das principais soluções existentes, aplicações e funcionalidades, com o objetivo de delimitar a infraestrutura a ser planejada. A compreensão do sistema requer grande atenção e cuidados especiais por parte do avaliador, para assim, evitar erros de interpretação e comprometimento das demais etapas da metodologia. Essa etapa é essencial, pois possibilita o conhecimento das técnicas que poderão ser adotadas, adaptadas ou que devem ser criadas.

É nesta etapa que será feito uma leitura do material técnico sobre computação em nuvem, sistemas de *VoD* e modelagem por meio de livros, artigos científicos, sites, fóruns, etc. Da mesma maneira, será realizado testes em ambientes reais, os quais permitem ao usuário um melhor entendimento do funcionamento. Assim, é possível determinar o modo operacional do sistema (critérios de funcionamento e entendimento do método de conversão e armazenamento de vídeos). Desta forma, como resultado parcial, teremos: referencial teórico e trabalhos relacionados, entendimento geral dos sistemas de nuvem e seu funcionamento, entendimento de alguns sistemas de *VoD*, assim como seu funcionamento e conhecimento sobre técnicas de modelagem de sistema.

Definição de parâmetros e métricas: o próximo passo é **definir os parâmetros e métricas** a serem avaliados no sistema que está sendo planejado, pois isso irá afetar diretamente nos tipos de modelos que serão criados mais adiante. Nessa etapa, o avaliador deve definir quais cenários e métricas são de interesse, focando principalmente naqueles que possuem maior influência nos níveis de qualidade do serviço oferecido.

Os parâmetros escolhidos definirão o nível de abstração da modelagem e irão depender do tipo de nuvem utilizado. Para esse tipo de dado, podemos subdividi-los em duas categorias: os parâmetros que requerem medição e os parâmetros colhidos da literatura. Os de medições, representam as características específicas da nuvem, VM e tipo de vídeo utilizado (parâmetros relacionados a desempenho), enquanto os parâmetros colhidos da literatura estão diretamente relacionados à dependabilidade. Desta forma, as principais respostas dessa análise são obtidas através das métricas. Nesse estudo buscamos identificar métricas de desempenho (tempo de resposta), dependabilidade(disponibilidade e COA) e custo.

Para a métrica de custo, são levados em consideração: se nuvem pública, computado pela soma dos custos médios de utilização de VMs sob demanda calculado pela *Amazon Web Services* (AWS) *simple monthly calculator* (AMAZON, 2017b). Cabe ressaltar que o custo associado à VM irá depender de cada fornecedor. Para nuvens privadas, a métrica de custo será a soma da aquisição de máquinas físicas utilizadas para atender a quantidade de VM requerida para o planejamento da infraestrutura de *VoD*. Uma melhor descrição dos custos são apresentados no Capítulo 5.

Geração de modelos: com os parâmetros e métricas estabelecidos, possíveis lacunas sobre o funcionamento do sistema identificadas e contornadas, assim como o foco da

avaliação estabelecido, é possível propor modelos de alto nível para obtenção de resultados preliminares do sistema. Dada uma infraestrutura de computação em nuvem, faz-se necessário criar uma visão geral do sistema para permitir a criação de modelos de alto nível. Dependendo do tipo de métrica a ser avaliada, pode ser adotado formalismos como SPN, RBD ou CTMC. Devido à complexidade e possibilidade de explosão de estados pelos modelos citados, formulações matemáticas de forma fechada podem ser usadas.

O ambiente de modelagem conta com uma ferramenta visual que auxilia o desenvolvimento e permite a computação das métricas. Dentre algumas ferramentas podemos citar o Mercury (SILVA et al., 2015). Desta forma, como entradas para elaboração desta etapa, é necessário que se tenha o tipo de métrica a serem avaliadas, métricas de disponibilidade e/ou desempenho, estabelecidas em etapas anteriores, o entendimento e funcionamento dos principais componentes ou subsistemas, assim como a descrição de dependências ou interligações entre os componentes do sistema.

Nesta etapa deve ser feito ajustes ao modelo, caso os resultados não sejam satisfatórios, os quais serão apresentados na etapa seguinte.

4.3 Avaliação:

A seção de **avaliação** está dividida em duas atividades: obter valores de entrada para os modelos e a avaliação dos cenários. Objetiva-se utilizar todo o conhecimento adquirido e material desenvolvido nas etapas anteriores para efetivamente avaliar o sistema.

Obter valores de entrada para os modelos: com base no entendimento do sistema apresentado anteriormente, a etapa para **obter valores de entrada para os modelos** é a atividade de coleta, que pode ser através de experimentos. Nesta etapa *scripts* de monitoramento são criados para coleta de valores específicos a fim de alimentar o modelo. Para compreendermos a utilização dos recursos e identificarmos um recurso limiar adequado em uma aplicação de transcodificação, foram criados *scripts* de monitoramento de recursos em uma VM transcodificadora, que registrou o tempo de transcodificação para determinados tipos de vídeos considerando os de tamanho único.

Para as métricas de dependabilidade (por exemplo, disponibilidade e COA) foram utilizados valores obtidos através de artigos científicos, livros e dados dos fabricantes dos equipamentos. Com os dados obtidos, é possível encontrar resultados suficientemente bons e, assim, tomar decisões corretas para se planejar ambientes de alta disponibilidade e com capacidade adequada.

Avaliação: a **avaliação** dos modelos é a atividade que compara os valores de saída com um valor de referência predefinido através de SLAs ou da expectativa do usuário final, ou de administradores de sistema. Quando o valor computado é satisfatório, o processo segue para a próxima etapa. No entanto, se ao menos uma métrica de interesse não alcançou um nível satisfatório, o processo deve reiniciar na atividade de **geração de modelos** para

que seja feito as alterações necessárias e, dessa maneira, o modelo seja reavaliado.

No processo de avaliação, submodelos que não dependem de resultados de outros modelos devem ser resolvidos primeiro, pois os resultados desse submodelo devem ser usados como parâmetros de entrada para modelos subsequentes ou submodelos dependentes. O método de solução (isto é, análise numérica ou simulação) pode variar para cada modelo, dependendo das restrições do formalismo de modelagem.

4.4 Otimização:

A etapa final da metodologia é composta por duas atividades: seleção de cenários e análise dos resultados. O objetivo é utilizar os resultados que foram obtidos na etapa anterior de avaliação e usá-los para construir cenários, dando possibilidades de configurações para diferentes perfis de usuário.

Seleção de cenários: com base nas métricas escolhidas (desempenho, dependabilidade e custo), diversos cenários podem ser propostos. Cada cenário leva em consideração a aplicabilidade de mecanismos de redundância, número de VMs, quantidade de máquinas físicas, etc. Com base nessas configurações é criado um algoritmo de otimização para identificar qual a configuração de cada parâmetro para que o sistema cumpra um SLA estabelecido de tempo médio de resposta com um custo otimizado. Essa técnica permite a verificação de diversos cenários otimizados, que seria impraticável, tanto no sistema real, quanto na variação manual dos parâmetros do modelo, devido à complexa interação dos parâmetros.

Essa técnica não garante encontrar a melhor resposta. Mas, sendo utilizada com uma quantidade de iterações suficientemente grande, é possível encontrar resultados otimizados bons o suficiente para aplicações práticas. Cabe ressaltar que a implementação dessa técnica é viabilizada pelo uso da *Application Programming Interface* (API) dos sistemas de modelagem SPN através de linguagens de programação, para que seja possível a mudança automatizada dos parâmetros do modelo. A aplicação desse método pode ajudar a responder questões como:

- Como deve ser configurado o sistema em nuvem privada para que o tempo de resposta seja inferior a X segundos com o menor custo possível;
- Qual a infraestrutura física mínima para atender ao SLA;
- Comparações de custo entre infraestruturas privadas e públicas.

Análise dos resultados: essa fase verifica os resultados alcançados com base em cenários complexos, com uma boa estimativa que esse mesmo comportamento possa ser apresentado por um sistema real. Dessa maneira, permite-se identificar a influência de determinado parâmetro no custo e nas métricas de desempenho. Com a sequência de

mudanças de parâmetros e avaliações do modelo, cenários são gerados. A escolha da melhor combinação de parâmetros nos fornece o melhor cenário com base nas características adotadas. A descrição da geração de cenário e busca de melhores soluções propostos nessa Tese de Doutorado são explicados em detalhes no Capítulo 5 na Seção 5.5.

4.5 Considerações finais

Este capítulo apresentou a metodologia utilizada para o planejamento de infraestruturas de computação em nuvem atendendo a serviços de *VoD* com foco principal na avaliação de requisitos de disponibilidade, COA, desempenho e custo. Através de um conjunto de etapas que aborda desde o entendimento do funcionamento básico do sistema até a seleção e analise de resultados de cenários para o serviço pretendido, este capítulo proporciona detalhes do planejamento da avaliação que podem ser replicados por outros pesquisadores.

5 MODELOS e OTIMIZAÇÃO

Este capítulo apresenta os modelos de disponibilidade, disponibilidade orientada à capacidade, de desempenho e os modelos de otimização. Para o melhor entendimento das descrições dos modelos apresentaremos modelos de alto nível do sistema, mostrando suas características e funcionalidades.

5.1 Visão geral

Esta tese de doutorado propõe modelos para o planejamento de infraestruturas de computação em nuvem considerando aspectos de transcodificação de vídeo para sistemas de *VoD*, com foco em aspectos de disponibilidade, desempenho e custo. Esta abordagem foca em modelos hierárquicos e possibilita a identificação de pontos de falhas de maior impacto para o sistema, bem como critérios para conversão de diferentes tipos de vídeos na nuvem, possibilitando a organização de quantidade de VMs de acordo com um *workload* específico.

A arquitetura da infraestrutura de computação em nuvem considerada nesta tese é muito semelhante a diversas situações para quaisquer tipo de ferramental para IaaS escolhido por administradores de sistemas em nuvem, tais como: sistema de nuvem baseado na plataforma *Open Nebula* (OPENNEBULA, 2017), o sistema *OpenStack* (OPENSTACK, 2017), *CloudStack* (APACHE, 2017a), etc. O Eucalyptus foi o sistema escolhido para realizar testes e experimentos durante a elaboração desta tese. A Figura 9 mostra uma visão geral de uma infraestrutura de computação em nuvem baseada na plataforma Eucalyptus. Esta tese toma como arquitetura base esse modelo computacional. É importante ressaltar que esse modelo base pode sofrer alterações para que possamos ter diferentes tipos de opções para avaliação.

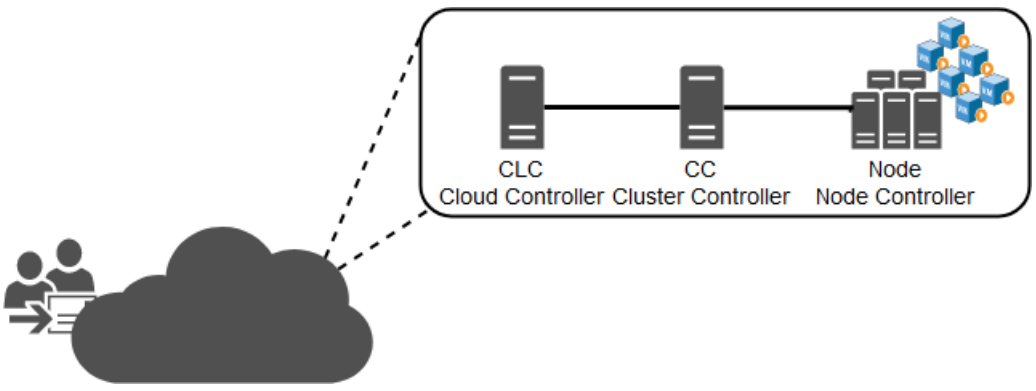


Figura 9 – Exemplo de infraestrutura baseado no sistema de *IaaS Eucalyptus*

Para o propósito geral deste estudo e facilitar a explicação da interação dos principais componentes do sistema, estamos considerando um serviço de *VoD streaming* implantado

em uma infraestrutura de computação em nuvem *Eucalyptus*, a qual é composta por três componentes principais: ***cloud controller***, ***cluster controller*** e ***node controller***. A definição para cada componente é descrita a seguir:

- O *Cloud Controller* (CLC) é o *front-end* para a infra-estrutura de nuvem. O CLC é responsável por expor e administrar os recursos subjacentes virtualizados (servidores, rede e armazenamento) via *Amazon Elastic Compute Cloud* (EC2) (EC2, 2016). Este componente utiliza interfaces de serviços da *Internet* para receber os pedidos de ferramentas de clientes, de um lado, e de interagir com o resto dos componentes do *Eucalyptus*, no outro lado. Neste nível, podemos encontrar também um arquivo baseado em serviço de armazenamento de dados, sendo sua interface compatível com o *Simple Storage Service* (S3) da *Amazon* (SERVICES, 2016b), conhecido como *Scalable Object Storage* (SOS).

- O *Cluster Controller* (CC) geralmente executa em um *cluster* de máquinas *front-end* (PACKARD, 2016) ou em qualquer máquina que tenha conectividade na rede. Os CCs reúnem informações sobre um conjunto de máquinas virtuais e horários de execução de VM em um nó específico. O CC tem três funções principais: agendar visitas, que consiste em implantação e execução para os nós do *cluster* específico, controlar a instância virtual de rede e reunir/relatar informações sobre um conjunto de nós (PACKARD, 2016). Assim, o CC deverá conferir os recursos disponíveis informados por cada nó e decidir qual máquina física deverá instanciar as VMs requisitadas pelo CLC. Neste nível, podemos encontrar também, um arquivo baseado em serviço de armazenamento, o Controlador de Armazenamento - *Storage Controller* (SC) que, por sua vez, fornece armazenamento de bloco persistente para uso das instâncias de máquinas virtuais. Ele implementa acesso a bloco de armazenamento em rede, semelhante ao proporcionado pela *Elastic Block Store* (EBS) (SERVICES, 2016a), e é capaz de interagir com vários sistemas de armazenamento (*Network file system* (NFS), *Internet Small Computer Systems Interface* (iSCSI), etc.). Para um único CLC podemos configurar várias máquinas físicas com o CC.

- O *Node Controller* (NC) é uma máquina que executa o serviço do nó controlador e controla o ciclo de vida das instâncias em execução. O NC interage com o sistema operacional e com o *hypervisor* em execução no nó. Os NCs controlam a execução, fiscalização e terminação das instâncias de VMs no *host* onde está sendo executado. Ele consulta e controla o *software* do sistema em seu nó em resposta às consultas e solicitações do controle do CC (PACKARD, 2016). Para cada CC configurado pode existir um aglomerado de NCs prontos para instânciação de VMs.

A seguir, iremos apresentar os conjuntos de modelos propostos. O primeiro conjunto tem por objetivo principal avaliar a disponibilidade de uma infraestrutura de computação

em nuvem voltada para serviços de *VoD streaming*. Para esse tipo de modelo apresentamos as características e método de avaliação utilizando uma abordagem hierárquica. Fazemos uso de um serviço de *VoD* específico e a arquitetura mencionada anteriormente. Esse modelo é representado para demonstrar as características e dificuldades encontradas na representação de sistemas usando essa abordagem. Características como processo de *upload* e transcodificação (desempenho) de vídeo não são levadas em consideração. O segundo conjunto descreve um modelo escalável para estimativa de capacidade e disponibilidade no sistema da nuvem IaaS. Esse modelo foi concebido na tentativa de ajudar administradores de sistema a construir uma infraestrutura de nuvem dependente de disponibilidade e capacidade para um grande conjunto de máquinas físicas e virtuais. O terceiro modelo descreve o sistema de *VoD*, o qual tem por objetivo principal avaliar o tempo de resposta de um determinado tipo de vídeo para ser publicado em uma infraestrutura de nuvem. Para esse modelo podem ser considerados diferentes tipos de vídeo e diferentes tipos de VMs. Em conjunto com o segundo modelo, esse terceiro modelo pode também descrever o impacto do tempo de resposta e perdas relacionadas à disponibilidade. O quarto conjunto de modelo tem por objetivo relacionar o custo de aquisição de máquinas físicas, para infraestrutura de nuvem privada, e o custo de aquisição de máquinas virtuais, para uma infraestrutura de nuvem pública. O quinto conjunto modela o processo de otimização. Esse processo torna-se importante para, de acordo com os parâmetros de configuração e os modelos apresentados anteriormente, ser gerado um conjunto de melhores cenários com base na disponibilidade, capacidade, tempo de resposta e custo.

5.2 Modelos de Disponibilidade

Disponibilidade para o serviço de *VoD* considerando um subsistema de *cluster* individual - O propósito deste estudo é avaliar um cenário de *VoD* com base em uma infraestrutura de nuvem e verificar a complexidade dos modelos analisados. Para a construção do sistema de *VoD streaming* foram estabelecidos alguns requisitos para a escolha do *software* capaz de realizar o *streaming*. Um dos requisitos requeridos pela equipe no processo de construção, implementação e testes no serviço de *VoD* foi que o *software* fosse gratuito. Desta forma, por possuir uma boa documentação, além de ser gratuito e de código aberto, foi iniciado um estudo preliminar sobre o FFMPEG (POWERED, 2017) e uma plataforma que pudesse suportar os vídeos *online*, com o intuito de criar um ambiente de serviço de *VoD streaming* de forma que atendesse as condições mínimas (envio do fluxo de vídeo, armazenamento dos arquivos de vídeos e que trabalhasse com diversos formatos de arquivos de vídeo).

A Figura 10 representa um modelo de alto nível do ambiente apresentado anteriormente, com *softwares* montados em um *host*. O servidor do serviço de *streaming* é instalado e configurado em uma VM, a qual é instanciada no ambiente da nuvem, e é formado pelas

aplicações FFMPEG e *apache web server*.

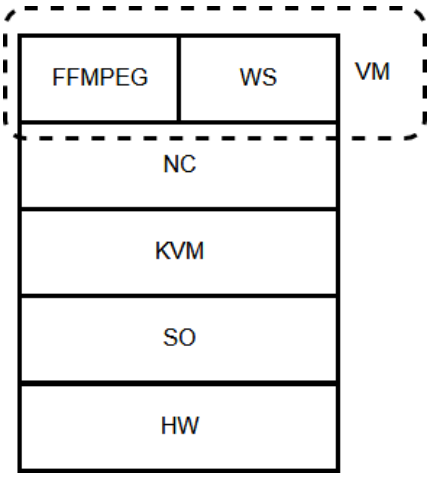


Figura 10 – Arquitetura de alto nível do sistema *VoD* em cima do NC

A função do Apache é dar suporte para que usuários possam visualizar o conteúdo oferecido através da *Internet*, enquanto o *FFMPEG* é uma ferramenta que possibilita a realização das conversões das mídias de vídeo. É importante salientar que a imagem da VM a ser utilizada para dar suporte ao sistema de vídeo *streaming* foi criada e personalizada a partir do *Virt-Manager* (VIRT-MANAGER, 2016). Essa imagem tem o *Ubuntu Server 14.04* como sistema operacional, com as aplicações *Apache* e *FFMPEG*, já instaladas e configuradas para armazenagem dos arquivos de vídeo. Assim que a VM for instanciada, ela será inicializada com todos os recursos prontos a serem utilizados. Esse passo é importante para que, no caso da necessidade de uma nova VM (processo de reparo da VM, por exemplo), a inicialização do serviço aconteça de forma rápida, diminuindo o *downtime*.

Os modelos para representação e avaliação de disponibilidade envolvem os modelos para avaliação da infraestrutura de *cloud* e os modelos para avaliação do serviço de *VoD* streaming. Para a análise de disponibilidade podem ser usadas técnicas de modelagem baseadas em Cadeia de Markov de Tempo Contínuo (CTMC) e Diagrama de Blocos de confiabilidade (RBDs). As Redes de Petri Estocásticas (SPNs) possuem um maior poder de representação por meio das simulações e análise numérica (HIREL; TUFFIN; TRIVEDI, 2000) (BALBO, 2002), que pode ser útil quando se tem sistemas mais complexos a serem avaliados, por exemplo, análise de desempenho. No entanto, as CTMCs e RBDs apresentam cálculos mais rápidos por meio de fórmulas fechadas (DANTAS et al., 2012) (DANTAS et al., 2016) e são mais úteis em ambientes mais simples.

Considerando a arquitetura mencionada anteriormente (*baseline*), gerente da *cloud* (CLC), gerente do *cluster* (CC) e o controlador do nó (NC), tem-se o *front-end* de toda a infraestrutura de nuvem, enquanto toda informação é recebida por esse elemento e retransmitia para o próximo componente da infraestrutura. O CC que, por sua vez, identifica o nó disponível no qual está rodando a VM e retransmite a mensagem para o nó

da *cloud*, o NC. Finalizando o processo, a requisição chega ao nó destino que a retransmite para a VM.

Todo o sistema está disponível se o CLC, o CC e se, pelo menos, um nó do conjunto de nós estiver disponível. É necessário que a VM também esteja disponível, ou seja, todos os quatro componentes devem estar funcionando corretamente. Uma única falha no CLC ou no CC vai fazer com que o sistema de *VoD* fique indisponível.

Para representar tal ambiente e estimar a disponibilidade do sistema, alguns modelos analíticos foram criados com auxílio da ferramenta Mercury (SILVA et al., 2015), enquanto as fórmulas fechadas foram obtidas com a ferramenta Mathematica (MATHEMATICA, 2017) para os modelos CTMCs. Alguns modelos foram subdivididos em subsistema para melhor facilitar o entendimento. O primeiro modelo de nuvem é representado por um modelo RBD, mostrado na Figura 11 (DANTAS et al., 2015). Cada subsistema possui métricas de disponibilidade calculadas através dos submodelos correspondentes considerando que o MTTF e o MTTR de cada componente sejam exponencialmente distribuídos. O mesmo pressuposto é usado para todos os submodelos.

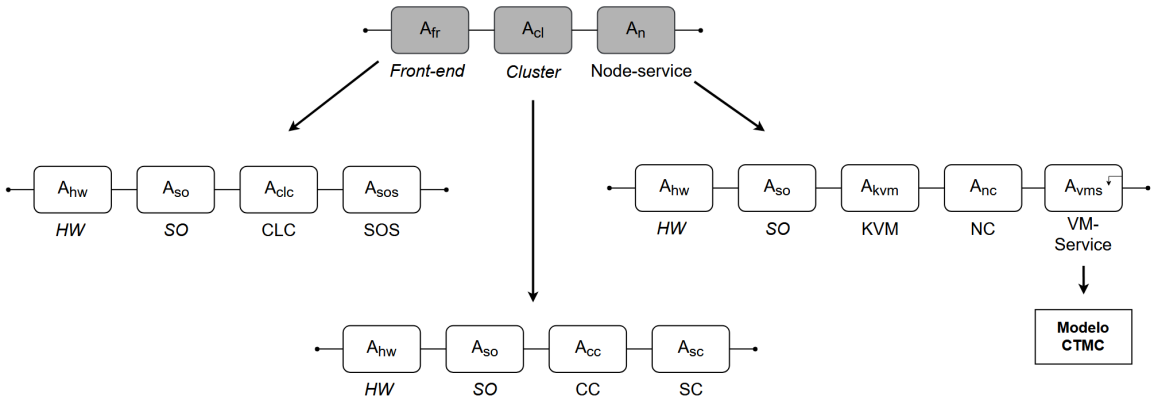


Figura 11 – Modelo RBD hierárquico para arquitetura básica - Com um *cluster* físico

A fórmula fechada para a disponibilidade do sistema completo (A_{sys}) é expressa pela Equação 5.1. Cada componente nesta equação é calculado a partir da avaliação do respectivo submodelo, que também pode ser feito através de fórmulas fechadas, caso seja possível obtê-las, ou através de solução numérica.

$$A_{sys} = A_{fr} \times A_{cl} \times A_n \tag{5.1}$$

O *front-end* é composto pelos componentes de *hardware* (hw), sistema operacional (SO), e os componentes do *Eucalyptus cloud controller* (CLC), assim como *Scalable Object Storage* (SOS). O *hardware* refere-se aos componentes eletrônicos, memória, placas e circuitos; o SO é o sistema em execução; o CLC e o SOS são aplicações configuradas e instaladas como parte do sistema *Eucalyptus*. Para a disponibilidade do componente

front-end (A_{fr}), podemos calcular seguindo a Equação 5.2.

$$A_{fr} = A_{hw} \times A_{so} \times A_{clc} \times A_{sos} \tag{5.2}$$

O *cluster* é composto pelos componentes de *hardware* (hw), sistema operacional (SO), assim como os componentes do *Eucalyptus cluster controller* (CC) e *storage controller* (SC). O *hardware*, assim como no *front-end*, refere-se aos componentes eletrônicos compostos no componente; o SO é o sistema em execução no *cluster*, enquanto o CC e o SC sao aplicações configuradas e instaladas como parte do sistema *Eucalyptus*. Para a disponibilidade do componente do *cluster* (A_{cl}), podemos calcular seguindo a Equação 5.3.

$$A_{cl} = A_{hw} \times A_{so} \times A_{cc} \times A_{sc} \tag{5.3}$$

O *node-service* é composto pelos componentes de *hardware* (hw), sistema operacional (SO), o *hypervisor* KVM e o componente do *Eucalyptus node controller* (NC), além do serviço de *VoD*, que devido a dependência entre componentes e a possibilidade de extração da fórmula fechada, vai ser representado por um modelo CTMC. Para a disponibilidade do componente do *nó* (A_n), podemos calcular seguindo a Equação 5.4.

$$A_n = A_{hw} \times A_{so} \times A_{kvm} \times A_{nc} \times A_{vms} \tag{5.4}$$

Como mencionado anteriormente, por ser um serviço que contém dependência entre componentes, o modelo da *VM-service* foi concebido por meio de um modelo CTMC. O (A_{vms}) é representada pelo modelo da Figura 12. O modelo apresenta cinco prováveis estados que o serviço de *VoD* pode alcançar: Up, F_{ap}, F_{apff}, F_{ff} e F_{all}. Dentre esses estados, apenas o estado Up indica que todos os componentes envolvidos estão funcionando adequadamente para que o sistema de *VoD* esteja entregando o serviço adequadamente, enquanto que os demais estados representam falhas no serviço. Quando algum componente alcança um estado de falha, o serviço de *streaming* deixa de ser provido. O estado F_{ap} representa a falha da aplicação Apache, o estado F_{apff} representa a falha da aplicação FFMPEG, mesmo depois da ocorrência da falha do Apache. A falha da aplicação FFMPEG, quando nenhuma outra aplicação apresentou defeito, é representado pelo estado F_{ff}. O estado F_{all} representa a falha em todos os componentes do sistema (Apache, FFMPEG e Máquina Virtual).

Na tentativa de entender o modelo CTMC que representa o serviço *VoD*, a descrição das taxas e estados em cada passo no modelo é definido a seguir: o estado Up indica que os componentes estão funcionando e que, por esse motivo, o serviço está disponível. A partir do estado UP, há a possibilidade do sistema ir para dois outros estados, F_{ap} com uma taxa de falha λ_{ap} ou para o estado F_{ff} com uma taxa de falha λ_{ff}, com as falhas das aplicações apache e FFMPEG respectivamente (não simultaneamente). Em ambos os casos, pode haver a recuperação da aplicação com as taxas de reparo μ_{ap} e μ_{ff} retornando para o

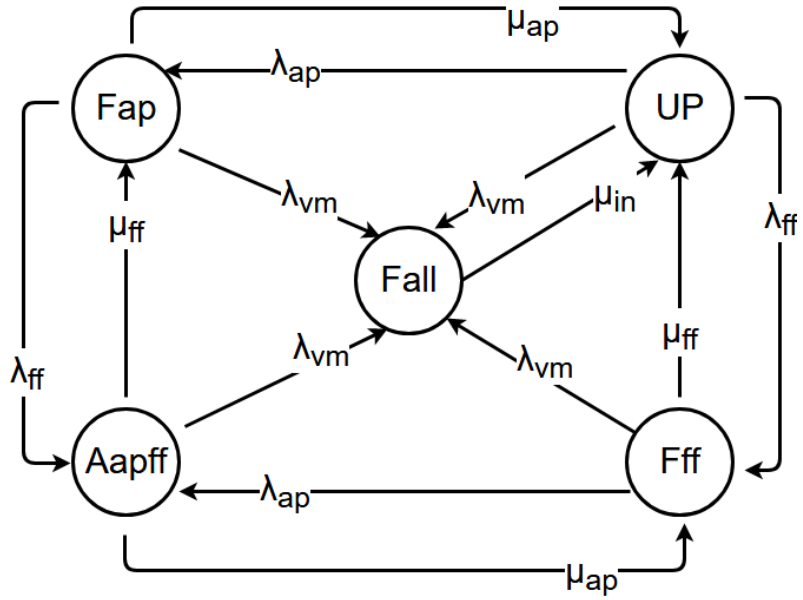


Figura 12 – Modelo CTMC - VM *VoD* service

estado UP. É importante salientar que as ocorrências de falhas, não necessariamente, vão ocorrer de forma simultâneas, assim como as ações de reparo. A partir do estado F_{ap}, no qual o serviço não está mais sendo provido devido à falha do Apache, o estado F_{apff} pode ser alcançado através da falha da aplicação FFMPEG. Da mesma maneira, ocorre para o estado F_{ff}, ressaltando apenas que nesse caso, o sistema alcança o estado F_{apff} através da ocorrência da falha do apache. Em todos os estados, há a possibilidade de ocorrência da falha da máquina virtual, alcançando o estado F_{all} com uma taxa de falha de λ_{vm}. Nesse estado, o sistema pode ser recuperado com a instanciação de uma nova máquina virtual com uma taxa μ_{in}. Ao ser inicializada, a VM também inicializa todos os serviços envolvidos automaticamente. É importante salientar que a imagem da VM que estamos utilizando é uma imagem customizada com todas as aplicações necessárias, automatizando o serviço de *start* do sistema *VoD*.

Os modelos analíticos alcançados são utilizados para análise de disponibilidade de uma arquitetura em específico. Primeiramente, a partir do CTMC apresentado acima, é possível alcançar a expressão da disponibilidade (Equação 5.5) para calcular a disponibilidade do serviço *VoD*.

$$A_{vms} = \frac{(\mu_{in}(\lambda_{ap}\lambda_{vm}(\beta) + \lambda_{ap}(\beta_1)\mu_{ff} + (\beta_1)(\beta_2)(\beta + \mu_{ff})))}{((\lambda_{ap} + \beta_1)(\lambda_{vm} + \mu_{in})(\beta)(\lambda_{ap} + \beta + \mu_{ff}))} \tag{5.5}$$

no qual,

$\beta = \lambda_{ff} + \lambda vm + \mu_{ap}$,

$\beta_1 = \lambda_{vm} + \mu_{ap}$,

$\beta_2 = \lambda_{vm} + \mu_{ff}$

Disponibilidade para o serviço de *VoD* considerando um subsistema de *cluster* mesclado ao *front-end* - Outra possibilidade de representação da arquitetura

de nuvem, considerando o *Eucalyptus* como ferramenta, e sem considerar o subsistema de *cluster* em uma única máquina física individual, pode-se considerar o *cluster* em conjunto com o subsistema *front-end*, **gerente da nuvem** (as aplicações CLC e CC mesclados em uma única máquina física, ver Figura 13). Desta forma, a Figura 14 representa um modelo de alto nível do ambiente de nuvem considerando apenas duas máquinas físicas. Os serviços e configurações são as mesmas mostradas na Seção anterior.

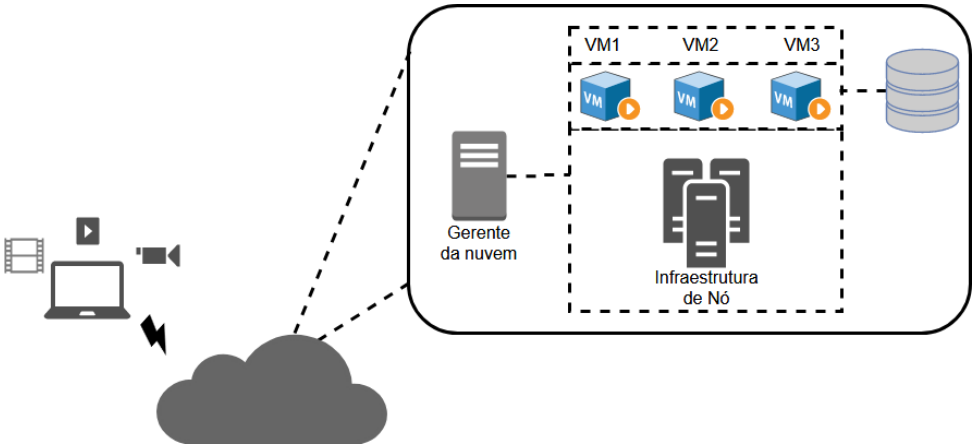


Figura 13 – Arquitetura de alto nível da arquitetura de nuvem

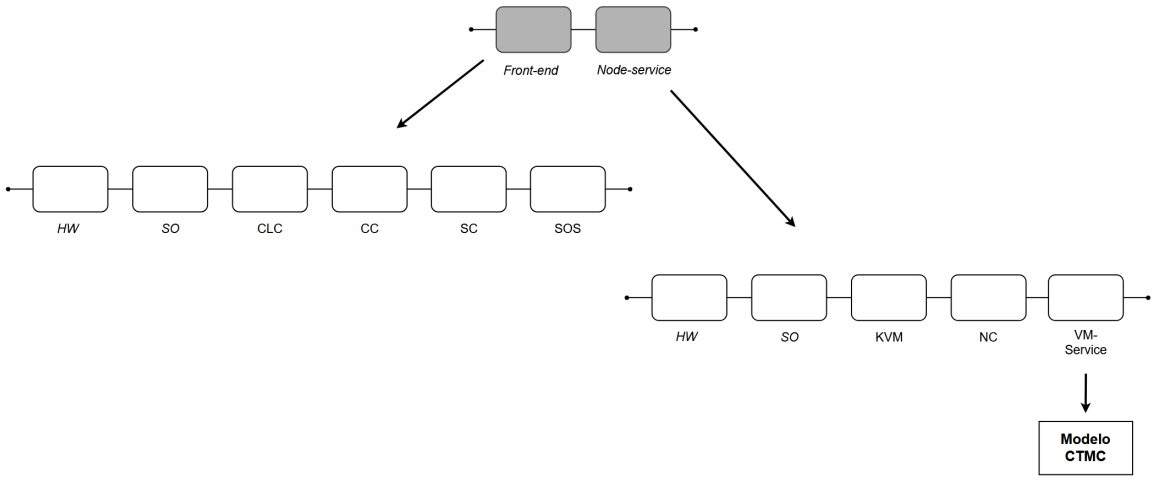


Figura 14 – Modelo RBD hierárquico para arquitetura básica - sem um *cluster* físico

A fórmula fechada para a disponibilidade nesse sistema (A_{sys}) é expressa pela Equação 5.6. Cada componente nesta equação é calculado a partir da avaliação do respectivo submodelo, que também pode ser feito através de fórmulas fechadas, se for possível obtê-las, ou através de solução numérica.

$$A_{sys} = A_{fr} \times A_n \tag{5.6}$$

Neste caso, o *front-end* é composto pelos componentes de *hardware* (hw), sistema operacional (SO), assim como os componentes do *Eucalyptus cloud controller* (CLC),

cluster controller (CC), *storage controller* (SC) e *Scalable Object Storage* (SOS). O *hardware* refere-se aos componentes eletrônicos, memória, placas e circuitos; o SO é o sistema em execução; o CLC, CC, SC e o SOS são aplicações configuradas e instaladas como parte do sistema *Eucalyptus*. Para esse tipo de subsistema, podemos calcular a disponibilidade do componente *front-end* (A_{fr}) seguindo a Equação 5.7.

$$A_{fr} = A_{hw} \times A_{so} \times A_{clc} \times A_{cc} \times A_{sc} \times A_{sos} \tag{5.7}$$

O *Node-service* é o mesmo apresentado na subseção anterior, desta forma, não será mostrado novamente.

Modelo de disponibilidade escalável - Visto a complexidade dos modelos apresentados e a possibilidade de crescimento de uma infraestrutura de nuvem para serviços de N categorias, o propósito deste estudo é apresentar uma estratégia para avaliar a disponibilidade de máquinas virtuais orientadas para a capacidade em uma infraestrutura de nuvem privada. A estratégia proposta visa fornecer uma computação eficiente e precisa da métrica COA, por meio de equações fechadas.

Para chegar a uma equação de forma fechada, faz-se necessário conhecer algumas etapas anteriores, etapas essas que são importantes para elaboração e construção do modelo matemático final. A Figura 15 apresenta etapas necessárias para a construção e avaliação do modelo matemático. As três atividades são designadas respectivamente como: (**i**) compreensão do sistema; (**ii**) modelagem e (**iii**) avaliação.

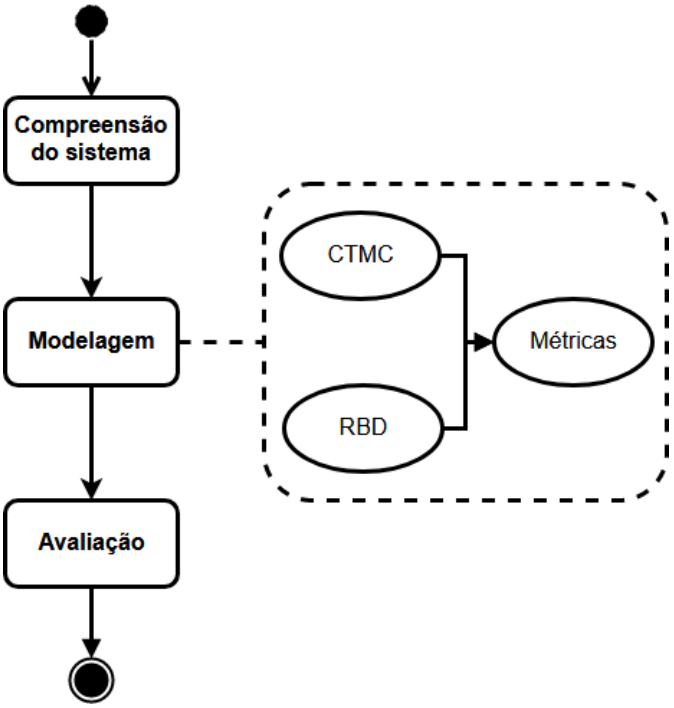


Figura 15 – Método para construção da equação

É importante destacar que o objetivo principal da visão geral do método é demonstrar passo a passo como obtemos a equação de forma fechada. Inicialmente (i), **compreensão**

do sistema representa como a infra-estrutura da computação em nuvem vai funcionar. Nesta etapa, a quantidade de VMs que serão executadas na infra-estrutura e o número de máquinas físicas que serão usadas para hospedá-las são definidas. O passo (ii), **Modelagem**, apresenta os dois formalismos de modelagem adotados nesta etapa de avaliação: CTMC e RBD. Este passo considera as vantagens de extrair equações e métricas de forma fechada a partir de modelos comportamentais, mais especificamente, do modelo mais adequado que representa o comportamento da infra-estrutura da VM e a interação de hardware e software na infraestrutura de computação em nuvem. No passo (iii), **avaliação**, aplica-se e estima-se os resultados das métricas de COA e disponibilidade.

Para a aplicabilidade prática do método proposto, aplicamos um exemplo para demonstrar a viabilidade da metodologia (veja o quadro tracejado da Figura 15): A partir do primeiro passo, assumimos que entendemos o sistema e o ambiente de nuvem fornecido, o que nos leva ao segundo passo que consiste em modelar a infraestrutura de VM e infraestrutura de computação em nuvem utilizando os modelos mencionados, CTMC e RBD respectivamente, a partir deste ponto, tentamos entender as equações e se seguem um determinado comportamento para que assim possamos generalizá-las.

Duas máquinas físicas compõem o exemplo proposto, PM1 e PM2, como mostrado na Figura 16. Cada máquina física suporta duas máquinas virtuais (VMs). A quantidade de máquinas virtuais no sistema representa a sua capacidade.

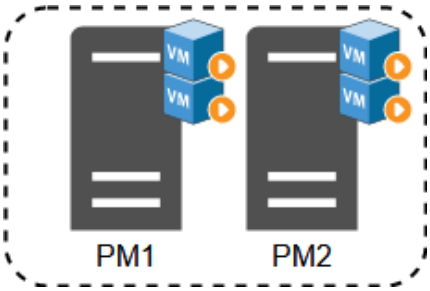


Figura 16 – Exemplo de infraestrutura com *computer nodes*

O sistema possui quatro VMs como capacidade máxima. Consideramos que a capacidade do sistema diminui sempre que ocorre uma falha em uma VM hospedada pela infraestrutura da nuvem ou por um nó físico da mesma, acarretando, então, a perda de 50% da capacidade. Assim, o número de VMs que podem ser executadas no sistema é expresso pelo acrônimo NVM, que é obtido pela Equation 5.8.

$$NVM = [P_4 \times 4] + [P_3 \times 3] + [P_2 \times 2] + [P_1] \tag{5.8}$$

Onde P_i é a probabilidade de o sistema estar em um estado com exatamente i VMs em execução.

Para cada P_i, fez-se necessário a criação de um modelo CTMC. O modelo está ilustrado na Figura 17. No modelo, os estados 4, 3, 2, 1 e 0 representam a probabilidade da

quantidade de VMs em execução no sistema, 4 VMs, 3 VMs, 2 VMs, 1 VM e 0 VMs, respectivamente.

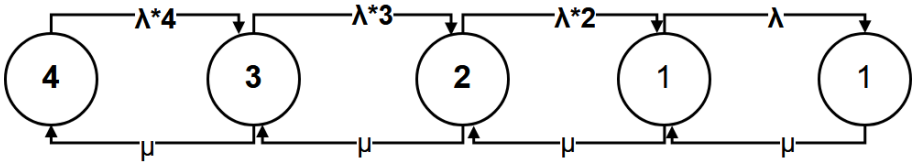


Figura 17 – Modelo de probabilidade de VMs

A partir da cadeia de Markov, podemos obter uma equação que representa a probabilidade para cada estado (veja as Equações 5.9, 5.10, 5.11 e 5.12). De acordo com as equações apresentadas, é possível notar o comportamento das equações considerando cada estado de probabilidade.

$$P_4 = \frac{\mu^4}{24\lambda^4 + 24\lambda^3\mu + 12\lambda^2\mu^2 + 4\lambda\mu^3 + \mu^4} \tag{5.9}$$

$$P_3 = \frac{4\lambda\mu^3}{24\lambda^4 + 24\lambda^3\mu + 12\lambda^2\mu^2 + 4\lambda\mu^3 + \mu^4} \tag{5.10}$$

$$P_2 = \frac{12\lambda^2\mu^2}{24\lambda^4 + 24\lambda^3\mu + 12\lambda^2\mu^2 + 4\lambda\mu^3 + \mu^4} \tag{5.11}$$

$$P_1 = \frac{24\lambda^3\mu}{24\lambda^4 + 24\lambda^3\mu + 12\lambda^2\mu^2 + 4\lambda\mu^3 + \mu^4} \tag{5.12}$$

Assim como o modelo CTMC da infraestrutura de VMs, também é proposto um modelo RBD para representar a infraestrutura dos nós físicos. Tal formalismo foi escolhido porque os componentes da infraestrutura eram independentes, o que implica que o RBD é uma representação mais simples e de alto nível do que um modelo CTMC. No entanto, conduzimos o estudo com *hosts* de computação organizados em paralelo (veja Figura 18). Neste contexto, o modelo de disponibilidade é representado pela probabilidade de pelo menos um bloco funcional. Assim, a capacidade do sistema cairá pela metade no caso de uma ocorrência de falha em um desses blocos.

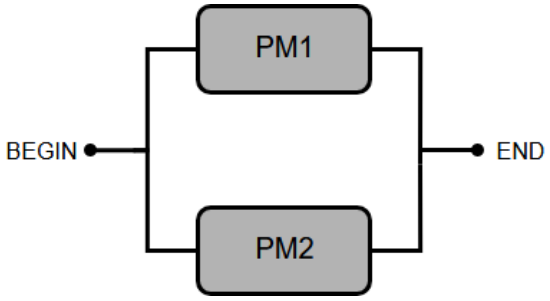


Figura 18 – Modelo RBD com dois nós físicos

A expressão de forma fechada para o exemplo descrito acima com os modelos apresentados pode ser descrito pela Equação 5.13.

$$COA = \frac{1}{4}(((\frac{\mu_{pm}}{\mu_{pm}+\lambda_{pm}})^2) \times ([P_4 \times 4] + [P_3 \times 3] + [P_2 \times 2] + [P_1]))+ \\ ((\frac{2\lambda_{pm}\mu_{pm}}{(\lambda_{pm}+\mu_{pm})^2}) \times ([P_2 \times 2] + [P_1])). \tag{5.13}$$

onde, $(\frac{\mu_{pm}}{\mu_{pm}+\lambda_{pm}})^2$ representa a expressão de disponibilidade para as duas máquinas físicas na infra-estrutura indicada pelo modelo RBD e $\frac{2\lambda_{pm}\mu_{pm}}{(\lambda_{pm}+\mu_{pm})^2}$ indica a expressão para pelo menos uma máquina física em execução na infra-estrutura também fornecida pelo modelo RBD. As taxas de falha e reparo são expressas por λ_{pm} e μ_{pm} respectivamente.

Seguindo as etapas descritas anteriormente, fizemos uma equação de disponibilidade orientada à capacidade generalizada. Portanto, o COA para uma ampla gama de possíveis configurações de infraestrutura de nuvem pode ser estimado por meio da Equação 5.14.

$$COA = \frac{1}{NVM}((\sum_{k=0}^{n}((A_s)^p VMs[k,n](n-k)))+ \\ (\sum_{j=1}^{p-1}\sum_{k=0}^{NV_j}(A_s(j,p) - A_s(j+1,p))(VMs[k,NV_j])(NV_j - k))) \tag{5.14}$$

onde NVM indica o número de máquinas virtuais no sistema e

$$VMs[K\_, n\_] := \left(\frac{\frac{n!}{(n-k)!}\mu^{(n-k)}\lambda_{vm}^{k})}{\sum_{i=0}^{n}\frac{n!}{i!}\lambda_{vm}^{n-i}\mu^i}\right) \tag{5.15}$$

- $VMs[K\_, n\_]$ é uma função que define a probabilidade de estado k, considerando um total de n VMs;
- NV_j representa o número de VMs (NV) por máquina física (j).

Ainda assim, podemos assumir que os componentes do *cluster* são independentes e idênticos. Portanto, todos os componentes têm a mesma distribuição de falhas e reparos. No caso de componentes em um arranjo *k-out-of-n*, a disponibilidade do sistema com essa configuração pode ser avaliada usando a distribuição binomial, ou:

$$A_s(j,p) = \sum_{i=j}^{p}\binom{p}{i}A^i(1-A)^{p-1} \tag{5.16}$$

onde,

- p é o número total de máquinas físicas em paralelo;

- j é o número mínimo de unidades necessárias para o sucesso do sistema;
- A é a disponibilidade de cada componente.

Para um sistema série-paralelo, a disponibilidade do ambiente, considerando um número j de máquinas físicas A_{ph}, pode ser expressa pela Equação 5.17.

$$A_{ph} = \left(\prod_{j=1}^{n} A_j \right) (A_s(j, p)) \tag{5.17}$$

5.3 Modelo de Desempenho

Um estudo de dependabilidade da infraestrutura de nuvem para serviços de *VoD* é importante para demonstrar o quão disponível é o sistema com base em um tempo de interesse, porém, em se tratando de sistema de vídeo sob demanda, faz-se necessário um estudo detalhado de desempenho da infraestrutura de nuvem que fornece o serviço. Esse estudo baseia-se na métrica de tempo de resposta, assim, o tempo de resposta é o tempo necessário para o usuário ter seus arquivos de vídeo publicados na *web*, considerando o fator de transcodificação de diferentes arquivos de vídeo para diferentes tipos de VM que pode ser usado na infraestrutura de nuvem.

Para facilitar o entendimento do funcionamento do modelo de desempenho, vamos aplicá-lo em uma arquitetura de um sistema de *VoD*. No entanto, ele poderia ser aplicado a qualquer outro cenário de computação em nuvem para serviços que exijam um alto desempenho e transcodificação dos arquivos.

A Figura 13, mostrada anteriormente, apresenta uma visão geral da arquitetura de um sistema de transcodificação de vídeo. A arquitetura pode ser subdividida em três partes: os clientes que enviam seus arquivos para serem publicados; a infraestrutura de nó que contém as VMs de diferentes tipos e capacidade; e o armazenamento e publicação do vídeo.

Nesse cenário, o cliente envia seu arquivo de vídeo para a nuvem, o qual está em algum formato (como por exemplo: .vp9[1], .h265[2], .vp8[3], etc). Em grande parte das infraestrutura de *VoD* é necessário converter o arquivo de vídeo para um formato específico no qual o administrador do sistema ache conveniente trabalhar ou que siga o padrão adotado para a tecnologia atual. Desta forma, a infraestrutura de nuvem usada nesta tese considera o formato de vídeo H.264/MPEG-4[4] a ser trabalhado. Todo vídeo que chegar à infraestrutura de nuvem será convertido para o formato especificado anteriormente.

Para esse cenário de transcodificação de vídeo, propomos um modelo SPN para representar esse processo. A Figura 19 apresenta o modelo abstrato de conversão de vídeo

[1] https://trac.ffmpeg.org/wiki/Encode/VP9
[2] http://x265.org/hevc-h265/
[3] https://trac.ffmpeg.org/wiki/Encode/VP8
[4] https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

na nuvem. Esse modelo é chamado de modelo abstrato por possuir transições temporizadas (transições cinzas) capazes de adotar valores que seguem uma determinada distribuição.

A Figura 20 apresenta o modelo refinado. A diferença entre os modelos apresentados na Figura 19 e 20 se refere à possibilidade de se trabalhar com determinadas distribuições - no nosso caso, adotamos a distribuição exponencial para as transições **T1** e **T2**. Em casos mais específicos, podem ser adotadas técnicas para o refinamento do modelo de desempenho, calculando-se médias e desvios-padrões provenientes de experimentos, valores esses adotados para as transições. 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 propósito geral deste modelo é calcular o tempo médio de resposta para a atividade de conversão de vídeo, no entando, nada impede de que outras métricas - como vasão - sejam obtidas. Os significados dos nomes utilizados no modelo estão na Tabela 2; a descrição dos atributos das transições na Tabela 3. Os parâmetros da distribuição das transições temporizadas devem ser providos pelo usuário do modelo o qual representa o processo de transcodificação considerando um tipo de VM em específico. Nesse caso, cada tipo de VM possuirá diferentes tempos para transcodificação. É importante ressaltar que, para a utilização de diferentes tipos de VM em uma mesma infraestrutura de conversão de vídeo, mudanças no modelo deverão ser consideradas. Os valores podem ser obtidos através de medidas em um sistema já hospedado na nuvem ou considerando cargas de trabalho esperadas em diferentes VMs em um sistema privado.

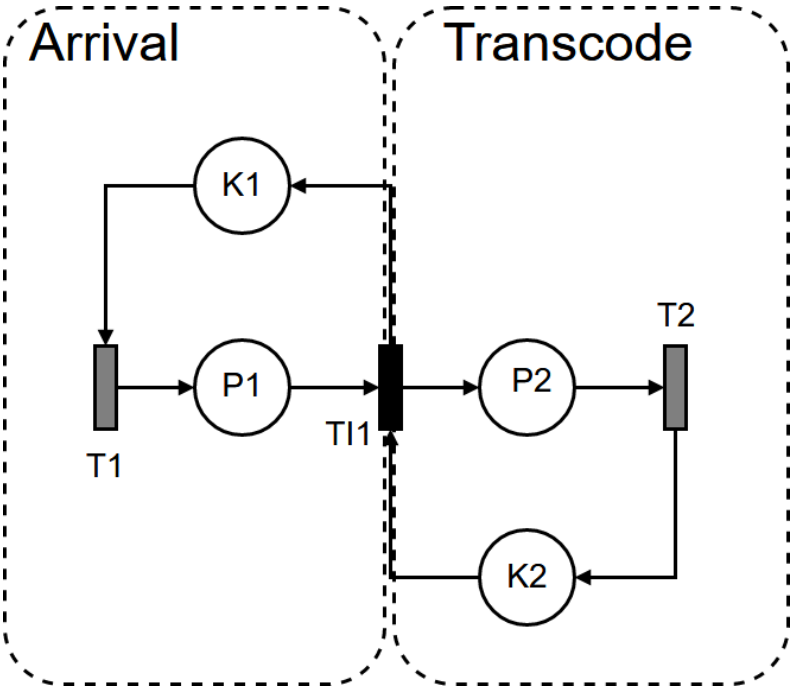


Figura 19 – Modelo SPN de desempenho - abstrato

O modelo é dividido em duas sub-redes: *Arrival*, a qual considera a chegada de trabalhos na infraestrutura; e *Transcode*, responsável pela fila e processo de conversão de vídeos. A

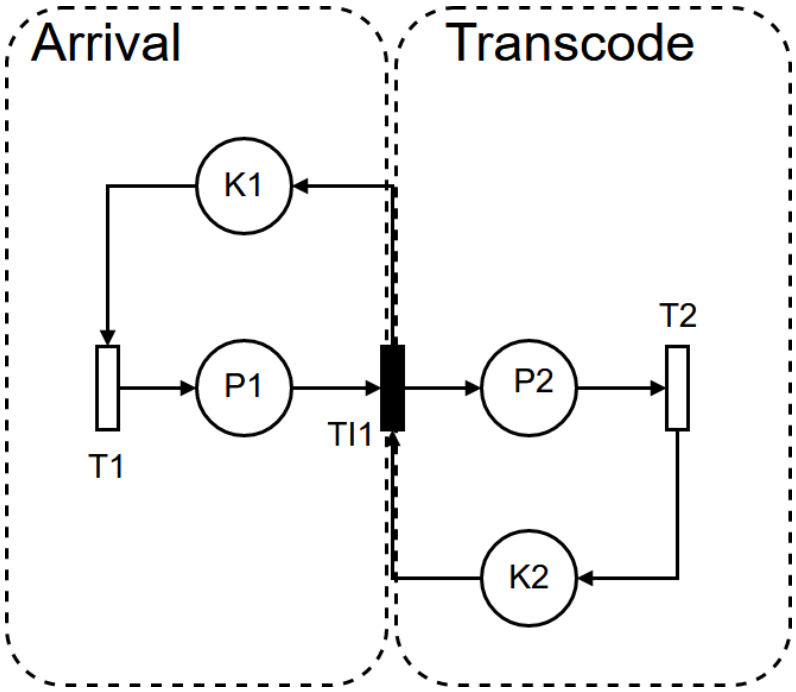


Figura 20 – Modelo SPN de desempenho - refinado

Tabela 2 – Descrição dos *places* e transições do modelo SPN

***Place* - (P/K)/Transição - (T)**	**Descrição**
T1	Tempo entre chegada de trabalhos
T2	Tempo para transcodificação
P1	Espera pela disponibilidade da fila
P2	Transcodificações sendo realizadas
K1	Tamanho da fila
K2	Recursos de processamento disponíveis

Tabela 3 – Descrição das transições para o modelo SPN

Transição	**Tipo**	**Server Semantic**	**Peso**	**Prioridade**
T1	Temporizada	Single Server	-	-
T2	Temporizada	Infinite Server	-	-
TI1	Imediata	-	1	1

sub-rede *Arrival* é composta por dois lugares **P1** e **K1**, que representam a espera entre chegada de trabalhos e a aceitação desse trabalho na fila, respectivamente, se o sistema tem capacidade K para aceitar os *jobs* para conversão, enquanto trabalha em conjunto com as transições **T1** e **TI1**. Quando a transição **T1** está habilitada, um *token* é consumido do lugar **K1** e é depositado no lugar **P1**, representando a chegada de vídeo para ser convertido. A transição **T1** representa uma transição temporizada com tempos entre chegadas de arquivos de mídia exponencialmente distribuídos; essa suposição pode ser modificada,

alterando essa distribuição. Outra consideração importante é que o tempo associado à transição **T1** considera apenas o tempo em que as transações entram no sistema, ou seja não são consideradas as perdas provenientes da rede.

Ao chegar trabalhos a serem convertidos no lugar **P1**, a transição imediata **TI1** (simbolizada por um retângulo preto, logo não terá atraso associado) será habilitada. Assim, será disparada logo que estiver habilitada. Quando **TI1** dispara a sub-rede de conversão (**transcode**) é alcançada, enquanto um *token* é retirado do lugar **P1** e **K2** e depositado no lugar **P2**. Desta forma, quando o lugar **P2** estiver com um *token*, significa que o processo de conversão (utilização do recurso) está em operação. A quantidade de *tokens* no lugar **P1** representa o enfileiramento de requisições, o qual ocorre quando não há capacidade disponível para servir a requisição recém-chegada. Se houver unidade de processamento disponível, a transição **TI1** dispara e, posteriormente, a requisição é processada.

O tempo em que os trabalhos permanecem em processo de transcodificação depende da transição temporizada **T2** (tempo para conversão do arquivo de mídia). Esta transição tem a semântica *infinite server*, que representa cada vídeo em processo de transcodificação processado independentemente. É importante notar que o tempo de transcodificação depende do tipo de VM e da quantidade de trabalhos simultâneos realizados.

O modelo da Figura 19 permite computar o tempo de resposta **RT** das requisições de transcodificações de vídeo no sistema, que pode ser calculado seguindo a Lei de Little (LITTLE, 1961), desde que o sistema esteja estável e que se conheça o número médio de trabalhos no sistema e a taxa de chegada. Desta forma, podemos calcular o RT para o modelo seguindo a Equação 5.18.

$$RT = (((E\{\#P1\}) + (E\{\#P2\}))/(1/T1)) \tag{5.18}$$

na qual, $E\#\{P_i\}$ representa o número médio de trabalhos no sistema e $1/T1$ representa a taxa de chegada.

5.4 Modelos de Custo

Os modelos de custo propostos foram formulados com base no custo de aquisição de máquinas físicas, para construção de uma nuvem privada, e custo de aquisição de máquinas virtuais, para aluguel de máquinas virtuais em uma nuvem publica para possíveis comparações de custo entre uma nuvem pública e nuvem privada. Esses modelos de custo são baseados em expressões matemáticas.

Modelo de custo para aquisição de máquinas físicas - A partir do número de máquinas virtuais e de acordo com o tipo de instância virtual que atenda a um *workload* específico, é possível conhecer o número médio de máquinas físicas necessárias para atender a demanda. A organização e divisão de recursos físicos para máquinas virtuais é realizada

pelo *hypervisor*; o *Eucalyptus* faz uso do KVM que faz o direcionamento de recursos com base na quantidade de núcleos disponíveis, somando núcleos físicos e virtuais. Por exemplo, se uma máquina física possui 2 núcleos físicos e 2 virtuais, forma-se um total de 4 núcleos, desta maneira, é possível instanciar até 4 máquinas virtuais com 1 CPU cada, ou 2 máquinas virtuais com 2 CPUs cada, ou uma única máquina virtual com 4 CPUs. A Equação 5.19 apresenta o calculo para o número de VMs necessárias a partir de uma carga de trabalho específica.

$$n_i = \frac{W}{NP_i}, \tag{5.19}$$

onde,

n_i: número de instâncias do tipo i;

W: workload gerado (número de usuários que vão fazer uso do sistema simultaneamente);

NP_i: número de trabalhos suportada pela VM do tipo i (capacidade de usuários em acesso simultâneo da VM do tipo i).

O valor de n_i pode ser definido como uma função teto (*ceiling function*), $N_i = \lceil n_i \rceil$ sendo definida pela Equação 5.20.

$$\lceil n_i \rceil = min\{n \in \mathbb{Z} | n \geq n_i\}, \tag{5.20}$$

Desta forma, é possível calcular o número de CPUs da infraestrutura virtual, informação importante para se se conhecer o número de máquinas físicas para uma infraestrutura privada (ver equação 5.21).

$$N_{CPUV} = \sum_{i=1}^{3} (CPU_i \times n_i) \tag{5.21}$$

onde,

CPU_i : Número de CPUs da VM do tipo i.

A Equação 5.22 apresenta o número de nós físicos necessários (NPC) para uma nuvem privada, com base nas métricas escolhidas.

$$NPC = \frac{N_{CPUV}}{NF_{CPU}} \tag{5.22}$$

onde,

NPC: Quantidade de nós físicos necessários para a infraestrutura;

N_{CPUV}: Quantidade de cores virtuais;

$NFCPU$: Quantidade média de cores ($físicos + threadsvirtuais$) da máquina física.

Tendo conhecimento do número de nós físicos necessários para o ambiente de *cloud* privada é possível modelar o custo de aquisição de máquinas físicas; o modelo de custo é descrito pela Equação 5.23.

$$MCF = NPC \times CF \tag{5.23}$$

onde,

CF: Corresponde ao custo do equipamento físico;

MCF: Modelo de custo de aquisição de máquinas físicas.

Modelo de custo para aquisição de máquinas virtuais - A presente proposta contempla também, infraestruturas de *cloud* pública; para isso, é levado em consideração o valor de aluguel de instâncias virtuais **reservadas**; o custo pode ser definido pela Equação 5.24.

$$CV = \sum_{i=1}^{3}(N_i \times C_i), \tag{5.24}$$

onde,

CV: Custo de aquisição de máquinas virtuais;

C_i: Custo da instância do tipo i;

5.5 Modelos de otimização

O modelo de COA para transcodificação de vídeo em nuvem com diferentes tipos de VM apresenta diversos parâmetros: a quantidade de VMs de cada tipo, a distribuição dos trabalhos para os diferentes tipos de VM e o tamanho da infraestrutura necessária para suportar essas VMs. Cada parâmetro possui diversos valores que podem ser atribuídos, enquanto a combinação das diversas configurações dos parâmetros gera inúmeros valores para as métricas de interesse do modelo. Nosso objetivo é buscar dentro do espaço de soluções aquela que apresente o desempenho mínimo acordado considerando a possibilidade de falha das VMs e *Hardware* que possua o menor custo. Para muitos sistemas, é difícil encontrar uma configuração que maximize (ou minimize) a métrica desejada ao encontrar uma determinada restrição, como custos e métricas mais específicas e o tempo de resposta. Técnicas de otimização são geralmente empregadas para alcançar uma alternativa que, pelo menos, está perto de melhor solução possível. A metodologia proposta, também, é de grande valor considerando esses casos. Utilizamos variações entre alguns parâmetros que levarão as métricas do sistema a uma solução ótima ou quase ótima.

Podemos lidar com problemas de desempenho, dependabilidade e problemas de planejamento de capacidade como casos específicos. Considere que um modelo para um determinado sistema em estudo tem um conjunto de N parâmetros a serem atribuídos, esses parâmetros podem representar o número de equipamentos, sejam eles físicos ou virtuais ou um *workload* gerado para a infraestrutura. O valor atribuído a cada parâmetro pode influenciar os resultados das métricas escolhidas. O processo de otimização deve encontrar a atribuição de melhores parâmetros para uma determinada arquitetura, composta por uma quantidade de máquinas físicas e virtuais que minimizam (ou maximizam) uma função objetivo θ, a qual pode ser disponibilidade orientada à capacidade, disponibilidade

estacionária, confiabilidade, tempo de resposta, custos ou mesmo uma composição de muitas métricas.

O processo de otimização deve fornecer um *vetor* SS^*; cada elemento desse *vetor* é um parâmetro da arquitetura que deverá ser encontrada no processo de otimização. SS_{vm1} representa a proporção da carga de trabalho que vai para a VM do tipo 1, SS_{vm2} representa a proporção da carga de trabalho que vai para a VM do tipo 2 e SS_{vm3} representa a proporção da carga de trabalho que vai para a VM do tipo 3. Da mesma forma, SS_{qvm1} representa a quantidade de VM, do tipo 1, SS_{qvm2} representa a quantidade de VM, do tipo 2 e SS_{qvm3} representa a quantidade de VM, do tipo 3, escolhida para dar suporte no processo de conversão de vídeo. SS_{ti} apresenta o tamanho da infraestrutura em quantidade de máquinas físicas. É importante salientar que, no processo de otimização, uma modificação deste parâmetro pode acarretar na mudança dos demais parâmetros.

Considerando o objetivo da função mencionado anteriormente o problema de otimização pode ser escrito como:

$$mim \quad \theta(SS) \tag{5.25}$$

Sujeito a:

$$\sum_{i=1}^{3} SS_{vm_i} = 1 \ , \ \forall \, SS_{vm_i} \in \mathbb{R} \tag{5.26}$$

$$1 \geq SS_{vm_i} \geq 0 \ , \ \forall \, SS_{vm_i} \in \mathbb{R} \tag{5.27}$$

$$\sum_{i=1}^{3} SS_{qvm_i} \leq NPC \ , \ \forall \, SS_{qvm_i} \in \mathbb{N} \tag{5.28}$$

$$SS_{qvm_i} \geq 0 \ , \ \forall \, SS_{qvm_i} \in \mathbb{N} \tag{5.29}$$

$$f(SS) \leq SLA \tag{5.30}$$

$$\sum_{i=1}^{3} N_i \times C_i \geqslant W \tag{5.31}$$

A primeira restrição (Equação 5.26) indica que a soma dos parâmetros que determina o *workload* pra cada tipo de vm do tipo i deve ser igual a **um**. A segunda restrição (Equação 5.27) indica que os parâmetros associados ao *workload* para cada VM do tipo i está contido entre **zero** e **um**. A terceira restrição (Equação 5.28) indica que a soma da quantidade de VM utilizada deve ser menor ou igual à capacidade dos nós físicos - NPC (ver Equação 5.22), respeitando assim, o número necessário de máquinas físicas para instanciar as VMs. A quarta restrição (Equação 5.29) indica que a quantidade de VMs do tipo i deve ser

maior do que **zero**. A quinta restrição (Equação 5.30) indica que a função objetivo deve respeitar o SLA (limiar de aceitação das arquiteturas). A sexta e última restrição (Equação 5.31) indica que a quantidade de VMs escolhidas, considerando a capacidade de suporte de acesso das mesmas, deve ser maior do que o *workload* gerado, no qual i, é o tipo de máquina virtual; N é a quantidade de máquinas virtuais usadas do tipo i; C é a capacidade de requisições simultâneas suportadas pela VM do tipo i; e W o *workload* gerado.

Para isso, foi implementado a metaheuristica GRASP para o problema, pois esse algoritmo apresenta soluções de boa qualidade, fugindo de mínimos locais, através da geração de soluções iniciais diferentes; através da inserção de um componente aleatório configurável que determina o tamanho da lista de candidatos (*restricted candidate list* (RCL)). Dessa lista serão escolhidos aleatoriamente os melhores elementos de cada parâmetro que irão compor a solução. Portanto, embora não garanta encontrar mínimos globais, apresenta soluções boas o suficiente para aplicações práticas em ambiente com enorme espaço de soluções em um tempo aceitável (MATEUS; RESENDE; SILVA, 2011). Os problemas tratados pelo GRASP geralmente são formulados como

$$mimf(x) \;\; sujeito \;\; a \;\; x \;\; \in \;\; X, \tag{5.32}$$

onde $f(.)$ é uma função objetivo a ser minimizada e X é um conjunto discreto de soluções viáveis. No nosso caso, como os valores de COA, desempenho e custo são de grandezas diferentes, normalizamos essas métricas seguindo a Equação 5.33 (SOUSA, 2015). A normalização das métricas proporciona a uniformidade aos resultados das avaliações das arquiteturas de nuvem.

$$m_{ni} = (m_{sni} - m_{mn})/(m_{mx} - m_{mn}), \tag{5.33}$$

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 $m_i \in \mathbb{R}$; m_{mn} é o valor mínimo da medida, tal que $m_{mn} \in \mathbb{R}$; m_{mx} é o valor máximo da medida, tal que $m_{mx} \in \mathbb{R}$.

Considerando os valores normalizados, a função objetivo tenta minimizar $f(m_{COUA}, m_{RT}, m_C)$ atribuída pela menor distância Euclidiana para a origem (0,0,0), seguindo a Equação 5.34.

$$DistZ = \sqrt{(m_{COUA})^2 + (m_{RT})^2 + (m_C)^2} \tag{5.34}$$

A Implementação do GRASP em pseudocódigo pode ser vista no Algoritmo 1, que tem como entradas o modelo M (da Figura 36); o conjunto de parâmetros PS (quantidade de VMs de cada tipo, a distribuição de trabalho para as VMs e o tamanho da infraestrutura em componente físico) que é formado por todos os valores a serem investigados de cada parâmetro no modelo, o parâmetro α que determina o tamanho da RCL e o SLA para o sistema (o *threshold* um limiar que determina níveis aceitáveis). Como é um algoritmo de minimização de custo, o algoritmo inicia definindo o melhor custo, sendo o infinito

para que seja trocado na primeira solução encontrada, a linha 2 apresenta o critério de parada escolhido, cuja quantidade de iterações são compostas das duas fases do algoritmo, a construção e a busca local nas linhas 3 e 4 respectivamente. A fase de construção gera soluções semi-gulosas válidas a partir da escolha das melhores configurações. Essa solução não precisa ser um mínimo local, pois a fase de busca local irá refinar essa solução inicial. Caso essa nova solução válida (que respeite o SLA) tenha a função objetivo (custo) de menor valor ela tomará o lugar da solução anterior como sendo a de menor custo (BSC) e será atribuída como melhor solução, linhas 6 e 7. Com o fim do algoritmo ele apresentará a melhor solução encontrada em todas as iterações, linha 10.

Algoritmo 1: GRASP Template

```
   Input: M, PS , α, SLA
 1 BSC ← ∞;
 2 while Iterations < maxIterations do
 3 |   S ← Construction(M, PS, α, SLA);
 4 |   S ← Local_Search(S, M);
 5 |   if (f(S) < BSC) then
 6 |   |   BSC ← f(S);
 7 |   |   S* ← S;
 8 |   end
 9 end
10 return S*
   Result: solution*
```

onde, M é o modelo, PS é o conjunto de parâmetros, α irá determinar o tamanho da lista restrita de candidatos e SLA é considerada um limiar para o tempo de resposta aceitável.

A fase de construção é apresentada no Algoritmo 2, que tem como entradas o modelo M, o conjunto de parâmetros PS (exemplo: carga de trabalho para cada VM que poderá variar de 0 a 1, enquanto a soma desse parâmetro deverá ser igual a 1; esse conjunto de parâmetros também pode ser representado pelo tamanho da infraestrutura física, que poderá variar de 1 a 10 em número de máquinas; e por fim, a quantidade de VM de cada tipo), o parâmetro α e o SLA como *threshold*.

Na linha 2 é gerado uma solução aleatória inicial, escolhendo-se um valor para cada parâmetro de forma aleatória e atribuído à variável da solução padrão SS (*standard solution*). O laço entre a linha 4 e a linha 9 irá construir uma solução semi-gulosa, e irá se repetir até que todos os parâmetros do modelo sejam escolhidos; em nosso caso, as quantidades de VM de cada tipo, as 3 probabilidades de distribuição de trabalho e a quantidade de máquinas para suportar as VMs (ver Modelo SPN da Figura 36). A função *evaluateElements* irá avaliar todas as configurações dos valores de um tipo de parâmetro levando em condição suas dependências das melhores soluções dos parâmetros anteriores. No caso da distribuição de VMs vai utilizar valores cuja soma de probabilidades seja 1, no

caso das VMs, irá considerar VMs de acordo com a quantidade de recursos utilizada para cada tipo e a capacidade disponível de *hardware* no momento.

A linha 6 vai selecionar, aleatoriamente, uma das melhores soluções das encontradas na linha 5; a quantidade de soluções será de acordo com o parâmetro α (COLMENAR et al., 2016). A linha 7 adiciona esse parâmetro como melhor solução; a linha 8 altera para o próximo parâmetro do conjunto, assim, até que todos os parâmetros sejam testados. A linha 10 avalia a configuração da solução criada pelos melhores elementos de cada iteração e a linha 11 verifica a validade dessa solução em relação ao SLA. Se a solução for válida ela será retornada para o algoritmo 1 e irá para a busca local; caso seja inválida, será gerada uma nova solução. As etapas do método de seleção da solução apresentado no algoritmo de construção também é apresentado pela Figura 21. Observe que o α é um valor escolhido entre 0 e 1. Se α for zero, qualquer valor para cada parâmetro pode ser escolhido, então, a melhoria é completamente aleatória. Se α for 1, então, somente o melhor valor de parâmetro pode ser escolhido. Na Figura 21, se α é de 0.3, então, de dez valores para o parâmetro SS_{ti} três são testados e será adicionado no RCL os melhores parâmetros para que assim seja testada a solução.

Algoritmo 2: Construction Model

Input: M, PS, α, SLA
1 **repeat**
2 $SS \leftarrow generateRandomCandidate(PS)$;
3 $PI \leftarrow 0$;
4 **while** *is not a complete solution* **do**
5 $ES \leftarrow evaluateElements(M, PS, PI, SS)$;
6 $SE \leftarrow randomRCL(ES, \alpha)$;
7 $SCS[PI] \leftarrow SE$;
8 $PI \leftarrow PI + 1$;
9 **end**
10 $CS \leftarrow Solve(M, SCS)$;
11 **until** $(isInvalid(CS, SLA))$;
12 **return** CS
Result: CS

onde, SS é a solução padrão, ES solução avaliada, SE é a seleção de elementos, SCS seleciona um conjunto de configurações passando alguns parâmetros de entrada PI.

A busca local (Algoritmo 3) é realizada para melhorar a solução encontrada na construção. Nós utilizamos o método de primeira melhoria, que ao ser encontrada, em relação ao valor da construção, o algoritmo utiliza esse valor como solução. Essa abordagem foi utilizada, pois a busca exaustiva de toda a vizinhança costuma apresentar resultados semelhantes com maior tempo (GLOVER; KOCHENBERGER, 2006). O critério de parada será um limite de buscas dada pela variável MI; a escolha do vizinho na linha 4, pela função *getNeighbor*, que terá sua função objetivo calculada na linha 5. Caso a solução seja válida e melhor que a solução da construção, ela será retornada para a próxima iteração do

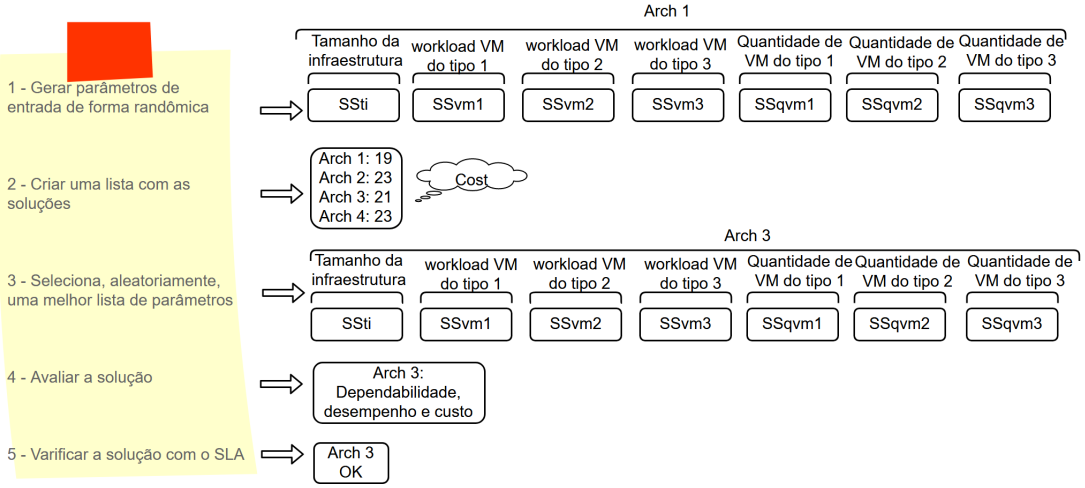


Figura 21 – Exemplo da solução inicial do método de construção

algoritmo. A Figura 22 apresenta um exemplo de dois vizinhos que podem ser encontrados partindo da solução SS. Algoritmo 3 procura em toda a vizinhança até preencher o conjunto de soluções ou até chegar ao número máximo definido de iterações. A vizinhança para os parâmetros do modelo considera uma parcela do intervalo total de valores para cada parâmetro, no qual γ irá definir a quantidade de elementos. A utilização de γ igual a 0.1, de um parâmetro com 10 elementos, significa que a busca local irá variar até um valor do valor encontrado na fase de construção. Ou seja, se o valor encontrado na construção para SS_{vm1} for igual a 0.5 implica em dizer que, com um γ igual a 0.1, os possíveis valores para SS_{vm1} podem ser de 0.4, 0.5 (já avaliado) e 0.6. Observe que, o restante dos parâmetros pode ser alterado respeitando as restrições (indicado pelas setas tracejadas na Figura 22). Observe que, se o valor de SS_{qvm_i} for igual a zero, automaticamente será atribuído zero para a probabilidade de *workload* da VM_i, SS_{vm_i}.

Algoritmo 3: Local Search Model

Input: M, CS, γ, MI

$i \leftarrow 0$;
while $(i < MI)$ **do**
  $i \leftarrow i + 1$;
  $NC[] \leftarrow getNeighbor(CS, \gamma)$;
  $NS \leftarrow Solve(M, NC[])$;
  if $(isValid(NS)\ AND\ f(NS) < f(CS))$ **then**
    return NS;
  end
end
return NS

Result: NS

onde, M é o modelo, CS é a *construction Solution*, γ define a quantidade de elementos a serem pesquisados, MI define um número máximo de interações e NC define a configuração da vizinhança.

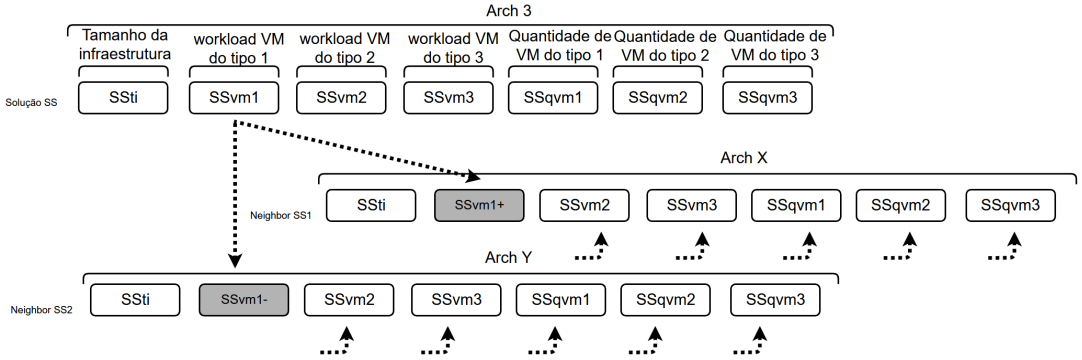


Figura 22 – Exemplo de vizinhança para a solução

Vale ressaltar ainda que, o processo de busca na vizinhança é um processo custoso e demorado, desta forma, todos os parâmetros podem ser modificados no processo de busca na vizinhança e só depois o modelo é avaliado novamente. A avaliação do modelo é realizada através da chamada da API da ferramenta Mercury e a solução dada por simulação estacionária tendo, assim, um novo resultado de uma nova arquitetura.

5.6 Considerações finais

Esse capítulo apresentou as principais contribuições desta tese, os modelos para infraestrutura de nuvem, bem como os pseudocódigos de otimização aplicados aos modelos. Descrevemos inicialmente um modelo proposto para disponibilidade do sistema *VoD* na nuvem, na tentativa de se ter cálculos para infraestruturas maiores. É proposto um modelo escalável na tentativa de suprir o problema de explosão de espaço de estado dos modelos anteriores. Para o serviço de *VoD*, um modelo de transcodificação de vídeo é proposto. Em seguida, apresentamos o modelo de custo para nuvem pública e privada. Então, apresentamos os pseudocódigos das fases de construção e busca local dos mecanismos de otimização aplicados nos modelos. Vale ressaltar que os métodos apresentados aqui foram aplicados com sucesso em vários estudos de caso, demonstrados no Capítulo 6.

6 ESTUDOS DE CASO

Este capítulo apresenta cenários como estudos de caso que demonstram a metodologia utilizada para apoiar o planejamento de infraestruturas de nuvem para serviço de *VoD streaming*: (i) a avaliação de disponibilidade de um sistema *VoD* em uma infraestrutura de nuvem; (ii) a avaliação de disponibilidade e capacidade de uma infraestrutura de nuvem considerando número de máquinas virtuais para um sistema de *VoD*, além de demonstrar a eficácia do modelo matemático escalável comparando-o com um modelo CTMC de mesmo poder computacional (método de validação do modelo escalável); (iii) avaliação de desempenho e performabilidade de um sistema *VoD* considerando tempo de resposta; (iv) criando cenários com o modelo de desempenho, validado, tendo como resultados o descarte (número médio de vídeos não processados por unidade de tempo), tempo de resposta de solicitações de conversão de vídeo e análise de custo; (v) esse estudo cria arquiteturas planejadas para suportar sistemas de vídeo sob demanda encontrando melhores valores de parâmetros. Para esse planejamento é considerado alguns dos modelos mencionados anteriormente em conjunto com o algoritmo GRASP. Este estudo de caso relaciona, também, um menor custo de implantação para o ambiente de nuvem.

6.1 Avaliação de disponibilidade do ambiente de *VoD streaming*

Estudos de casos distintos são apresentados nesta seção para avaliar o ambiente de *VoD streaming* de acordo com os modelos apresentados na Seção 5.2. No primeiro momento, é importante conhecer o ambiente de nuvem que irá trabalhar. Para isso, é necessário uma análise de disponibilidade do ambiente de nuvem considerando *front-end+cluster* em máquinas físicas separadas e *front-end+cluster* em uma única máquina física. A segunda parte da nossa análise está focada no ambiente de *VoD*.

O principal objetivo desse estudo de caso é estimar a disponibilidade do sistema de nuvem e assim, escolher a melhor configuração de ambiente possível para o sistema *VoD*. É importante fazer uma análise primária dessas arquiteturas para que se tenha conclusões sobre que tipo de infraestrutura podemos utilizar, já que a quantidade de máquinas em uma infraestrutura de nuvem privada vai influenciar diretamente no custo de aquisição. Posteriormente, visa-se estimar a disponibilidade do sistema *VoD*.

6.1.1 Parâmetros de entrada

A Tabela 4 apresenta os parâmetros de entrada para o front-end (em uma possível configuração considerando que o sistema de *cluster* é um nó físico individual). Os valores de MTTF e MTTR do *hardware* (hw) e sistema operacional (so) são estimados em (KIM;

MACHIDA; TRIVEDI, 2009). Para os componentes do *Eucalyptus*, como em sua maioria, são baseados em composições *web-services*, os valores de MTTF e MTTR foram retirados de (HU et al., 2010).

Tabela 4 – Parâmetros de entrada para o submodelo do *Front-end* e *cluster*

Componente	**MTTF (h)**	**MTTR (h)**
HW	8760	1.67
SO	2880	1
CLC = CC	788.4	1
SOS = SC	788.4	1

A Tabela 4 mostra ainda os parâmetro MTTF e MTTR para o bloco que representa o componente de *cluster* na infraestrutura de nuvem com um possível *cluster* em uma máquina física individual. É importante ressaltar que, quando se tem máquinas físicas para representar cada componente da nuvem, como é o caso do *front-end* e *cluster*, é possível considerar que as mesmas podem ser idênticas. Neste caso, o MTTF e MTTR do *hardware* e do sistema operacional são os mesmo para ambas as máquinas.

A Tabela 5 apresenta os parâmetros de entrada para o sistema do *Node-service*. A mesma infraestrutura física, os valores de MTTF e MTTR para *hardware*, sistema operacional e os componentes do Eucalyptus permanecem os mesmo dos apresentados anteriormente; os valores de MTTF e MTTR do componente *VM-Service* são extraídos do modelo CTMC apresentado na Seção 5.2.

Tabela 5 – Parâmetros de entrada para o submodelo do *Node-service*

Componente	**MTTF (h)**	**MTTR (h)**
HW	8760	1.67
SO	2880	1
NC	788.4	1
KVM	2990	1
Vm-Service	217.78	0.464

A Tabela 6 mostra os parâmetros de entrada para o modelo CTMC proposto na Seção 5.2. As taxas de falha para a VM (λ_{vm}) e para a aplicação FFMPEG (λ_{ff}) foram estimados de (KIM; MACHIDA; TRIVEDI, 2009). Já a taxa de falha do Apache (λ_{ap}) foram retirados de (HU et al., 2010) por ser um *web server*. As taxas de reparo do Apache (μ_{ap}) e do FFMPEG (μ_{ff}) foram estimadas considerando o fator de pior caso, ter conhecimento do problema da aplicação e ter que instalá-la e configurá-la novamente, que demanda tempo. É relevante ressaltar que para as taxas de reparo das aplicações foi utilizado o mesmo tempo. Para a taxa de reparo da VM (μ_{in}), como mencionado em capítulos anteriores, o sistema foi customizado com todas as aplicações necessárias para o funcionamento e a imagem da

VM armazenada, assim, quando uma nova VM é instânciada as configurações e aplicações são inicializadas sem problemas. Neste caso, é considerada a taxa de instanciação de uma nova VM, retirada por experimento.

Tabela 6 – Parâmetros de entrada para a *VM-Service* (Modelo CTMC)

Componente	**Descrição**	**Valor** (h^{-1})
λ_{ap}	Tempo médio de falha do Apache	1/788.4
λ_{ff}	Tempo médio de falha do FFMPEG	1/336
λ_{vm}	Tempo médio de falha da VM	1/2880
$\mu_{ap} = \mu_{ff} = \mu$	Tempo médio de reparo da aplicação	1/0.5
μ_{in}	Tempo médio de instanciação de uma nova VM	1/0.030

Com os parâmetros definidos, podemos avaliar cada submodelo e, com base nos valores de MTTF e MTTR trazidos por cada conjunto de modelos, chegamos aos valores de entrada para o modelo RBD, considerando o sistema com o *front-end* e *cluster* em máquinas físicas individuais; e o modelo RBD do sistema considerando o *front-end* e *cluster* em uma mesma máquina física. A Tabela 7 mostra esses parâmetros.

Tabela 7 – Parâmetros de entrada para componentes da infraestrutura

Componente	**MTTF (h)**	**MTTR (h)**	**Descrição**
Front-end	333.54	1.03	Máquina física individual
Cluster	333.54	1.03	Máquina física individual
Front-end+Cluster	180.67	1.07	Mesma máquina física
Nó	150.24	0.64	Configuração do nó e serviço *VoD*

6.1.2 Resultados dos modelos

O estudo inicial se baseia na disponibilidade da infraestrutura de *VoD streaming* na tentativa de combinar os componentes da infraestrutura e ter conclusões com base em que tipo de infraestrutura usar para a continuidade do trabalho. Os resultados preliminares são descritos na Tabela 8, na qual os valores são descritos como disponibilidade para o subsistema de *cluster* integrado ao *front-end* (***front-end+cluster***) e a disponibilidade para um subsistema de *cluster* em uma máquina física independente (***Cluster* independente**). Vale destacar a diferença de mais de 4 horas de *downtime* entre os cenários. Esta diferença é esperada devido à presença de outro componente físico na arquitetura com um *cluster* independente. Como consequência, a disponibilidade do sistema aumenta significativamente de 98% no cenário com um *cluster* físico independente para 99% no cenário em que temos um *Front-end* e um *cluster* juntos em um mesmo meio físico (***Front-end+Cluster***). É importante salientar que isso não implica em dizer que uma infraestrutura com *cluster*

separado seja pior do que outra com *cluster* integrado ao *front-end*, haja vista que, uma infraestrutura que ofereça suporte a *clusters* separados ofereça diversas possibilidades de configurações, podendo ser subdivididas em *clusters* localmente distribuídos ou *clusters* geograficamente distribuídos.

Tabela 8 – Resultado de disponibilidade do subsistema de *Cluster* (integrado x independente)

Medidas	**Front-end+Cluster**	***Cluster* independente**
MTTF (horas)	82.028	79.037
MTTR (horas)	0.814	0.827
Disponibilidade (%)	99.017	98.964
Disponibilidade (9's)	2.007	1.985
Uptime (horas/ano)	8679.666	8674.999
Downtime (horas/ano)	86.147	90.813

A análise para sistemas com *cluster* distribuídos pode-se considerar uma arquitetura com nove nós subdivididos em três *cluster* distintos e compará-la a uma outra arquitetura com nove nós subdivididos em um sistema de um único *cluster*. A Figura 23 apresenta o modelo RBD para um sistema com nove nós e o *cluster* trabalhando em conjunto com o subsistema *front-end* (*front-end+cluster*). O bloco *node-service 1/9* representa uma organização dos blocos para o serviço *VoD* e o nó em *k-out-of-n*. Essa representatividade mostra a quantidade de nós k necessárias para o funcionamento do sistema de um total de n. Desta forma, o sistema está funcionando se ao menos um nó estiver funcionando de um total de nove.

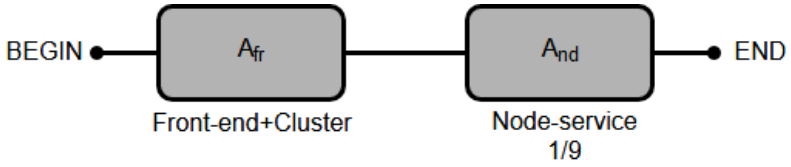


Figura 23 – Modelo RBD com nove nós e um subsistema de *Front-end+cluster*

A Equação 6.1 apresenta a equação de forma fechada para o cálculo da disponibilidade do sistema A_{sys}.

$$A_{sys} = A_{fr} \times \sum_{i=1}^{9} \binom{9}{1} A_{nd}^{1}(1 - A_{nd})^{9-1} \tag{6.1}$$

A Figura 24 apresenta o modelo RBD para um sistema com nove nós. Nesta arquitetura o subsistema de *cluster* trabalha independente de forma distribuída. Para cada subsistema de *cluster* existem três *nodes-service*. Cada bloco *node-service 1/3* representa uma organização dos blocos para o serviço *VoD* e o nó em *k-out-of-n*. Essa representatividade mostra a quantidade de nós k necessárias para o funcionamento do sistema de um total de n. Desta

forma, o sistema está funcionando se ao menos um nó estiver funcionando de um total de três para cada subsistema de *cluster*.

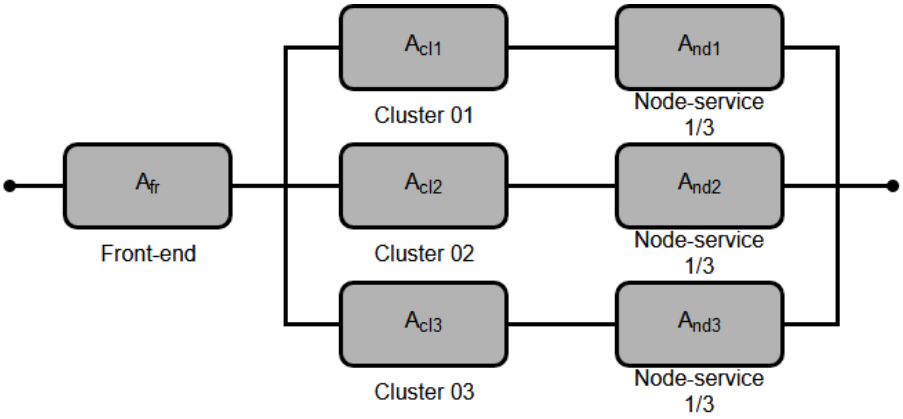


Figura 24 – Modelo RBD com nove nós subdividido em três subsistemas de *cluster*

A Equação 6.2 apresenta a equação de forma fechada para o cálculo da disponibilidade para essa configuração de sistema A_{sys2}.

$$A_{sys2} = A_{fr} \times (1 - (1 - (A_{cl1}) * (A_{nd1})) * (1 - (A_{cl2}) * (A_{nd2})) * (1 - (A_{cl3}) * (A_{nd3}))) \quad (6.2)$$

onde,

$A_{nd_i} = \sum_{i=1}^{3} \binom{3}{1} A_{nd_i}^{1} (1 - A_{nd_i})^{3-1}$

Os resultados são descritos na Tabela 9 como valores em disponibilidade para o subsistema de *cluster* integrado ao *front-end* (*front-end+cluster* integrado) e a disponibilidade para um subsistema de *cluster* em uma máquina física independente (*Cluster* independente).

Tabela 9 – Resultado de disponibilidade do subsistema de *Cluster* (integrado x independente) nove *nodes-services*

Medidas	**Front-end+Cluster** integrado	***Cluster* independente**
MTTF (horas)	155.463	179.329
MTTR (horas)	0.874	0.552
Disponibilidade (%)	99.441	99.693
Disponibilidade (9's)	2.252	2.513
Uptime (horas/ano)	8716.790	8738.917
Downtime (horas/ano)	49.022	26.896

Os resultados mostram que com uma arquitetura com *clusters* independentes temos melhores resultados em comparação à arquitetura com *clusters* integrado ao *Front-end*, porém, quando envolvemos o custo de aquisição de máquinas físicas para a infraestrutura privada, podemos ter outra conclusão com base nos resultados apresentados. A Tabela

10 apresenta os resultados de custo e disponibilidade das arquiteturas analisadas, sendo arquitetura **A1** - *Front-end+Cluster* integrado e arquitetura **A2** - *Cluster* independente.

Tabela 10 – Resultado de custo e disponibilidade do subsistema de *Cluster* (integrado x independente)

Medidas	Custo (US$)	Disponibilidade (%)
A1	24,871.10	99.441
A2	32,332.43	99.693

Para determinar as arquiteturas de melhor relação entre custo x disponibilidade, mas, por serem de grandezas diferentes, é necessário normalizá-los e colocá-los dentro de um mesmo intervalo: 0, 1. Processo descrito no Capítulo 5 Seção 5.5.

De posse dos valores normalizados do custo e disponibilidade, é preciso relacioná-los de alguma forma; utilizamos, então, da distância euclidiana e a arquitetura com menor distância da origem tende a ser a de melhor custo benefício. A Equação 6.3 mostra como é realizado o cálculo das distâncias considerando esse caso em específico.

$$DistZ = \sqrt{(m_{Custo})^2 + (m_{Dist})^2} \tag{6.3}$$

onde,

Z representa as arquiteturas: A1 e A2;

m_{Custo} representa o custo normalizado;

m_{Dist} a disponibilidade normalizada.

A Figura 25 apresenta, graficamente, a relação custo X benefício entre as arquiteturas, enquanto a arquitetura A1 apresenta a menor distância para a origem fazendo com que seja a arquitetura escolhida como melhor, nesse caso. Por esse motivo, será o modelo de arquitetura que escolhemos para as infraestruturas futuras (*Front-end+Cluster* integrado).

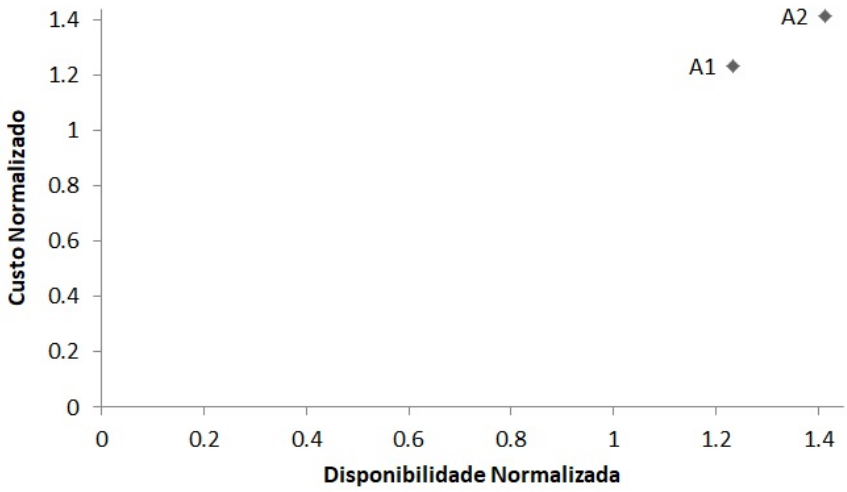


Figura 25 – Relação entre custo e disponibilidade

6.2 Estimando disponibilidade e capacidade do sistema de *VoD streaming*

Essa seção apresenta o processo de validação e resultados do modelo matemático escalável para estimar a capacidade e disponibilidade de sistemas de IaaS. A validação dar-se-á pela análise comparativa de uma infraestrutura de nuvem avaliada através de um modelo CTMC e a mesma infraestrutura comparada ao modelo matemático escalável apresentado no Capítulo 5.

6.2.1 Modelo CTMC para infraestrutura de nuvem

Seguindo o exemplo descrito na seção 5.2, foi concebido um modelo CTMC que representasse uma infraestrutura capaz de realizar comparações com os resultados do modelo matemático, sem que houvesse a explosão de estados. O modelo considera duas máquinas físicas, cada uma hospeda até duas máquinas virtuais. A infraestrutura suporta um total de até quatro máquinas virtuais executando no sistema. Desta forma, o modelo CTMC apresentado na Figura 26 tenta quantificar a disponibilidade do sistema, downtime e capacidade. O modelo apresenta doze estados. Os detalhes de cada estado pode ser descrito na Tabela 11. Os estados Sombreados $UD0$, $UU0$, $DU0$ e $DD0$ apresentam estado indisponíveis, em outras palavras, o sistema não tem máquinas virtuais disponíveis.

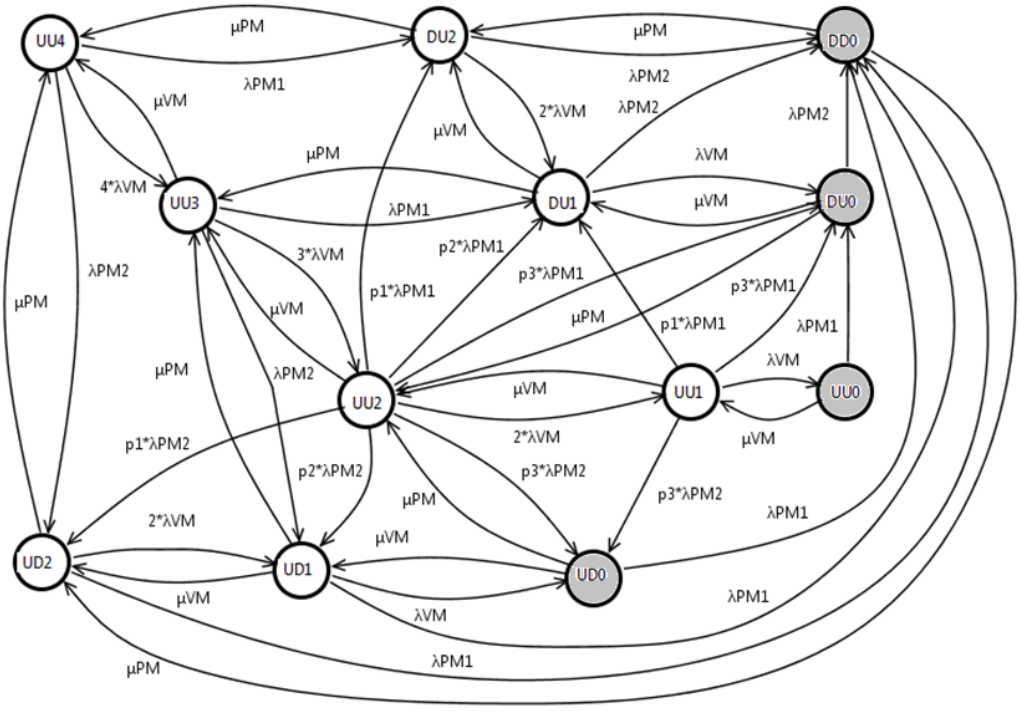


Figura 26 – Modelo CTMC para o exemplo com duas máquinas físicas

A notação representa a condição do sistema para cada estado. O nó de computação pode estar up (U) ou down (D). Se existir dois nós de computação disponíveis, o sistema

Tabela 11 – Descrição para os estados do modelo CTMC

Rótulo dos estados	**Descrição**
UU4	Dois nós de computação UPs e quatro VMs UPs
UU3	Dois nós de computação UPs e três VMs UPs
UU2	Dois nós de computação UPs e duas VMs UPs
UU1	Dois nós de computação UPs e uma VM UP
UU0	Dois nós de computação UPs e zero VMs UPs
UD2	Um nó de computação UP e duas VMs UPs
UD1	Um nó de computação UP e uma VM UP
UD0	Um nó de computação UP e zero VMs UP
DU2	Um nó de computação UP e duas VMs UPs
DU1	Um nó de computação UP e uma VM UP
DU0	Um nó de computação UP e zero VMs UP
DD0	Sem serviço

pode ter quatro, três, dois, um ou zero máquinas virtuais executando (UU4, UU3, UU3, UU1 ou UU0, respectivamente). Se existir apenas um nó de computação disponível no sistema, o sistema pode ter dois, um ou zero máquinas virtuais em execução (UD2, UD1 e UD0 ou DU2, DU1 e DU0, a depender de qual máquina física irá falhar). O estado DD0 representa a falha de todos os nós de computação, consequentemente, não há a possibilidade de instanciação de máquinas virtuais no sistema. É considerado atividade de falha e reparo do sistema, assim como atividade de falha e reparo das instâncias virtuais.

A falha do nó de computação é um evento que ocorre quando algum componente, seja ele *hardware* ou *software*, não funciona da forma correta ocasionando uma falha na entrega correta do serviço (mal funcionamento). As taxas de falha relacionada aos nós de computação são representadas por λ_{PM} e λ_{PM2}, enquanto que μ_{PM} representa a taxa de reparo dos nós de computação. Assumindo que as máquinas virtuais são idênticas, as respectivas taxas de falha e reparo podem ser expressas por λ_{VM} and μ_{VM}.

A Tabela 12 descreve os parâmetros de entrada para o modelo CTMC descrito anteriormente. Esses valores foram obtidos de (DANTAS et al., 2012) (DANTAS et al., 2015) (DANTAS et al., 2016).

Tabela 12 – Parâmetros de entrada para o modelo CTMC

Parâmetros	**Descrição**	**Valores (h^{-1})**
$\lambda_{PM1} = \lambda_{PM2}$	Taxa de falha das máquinas físicas	1/8760
λ_{VM}	Taxa de falha da máquina virtual	1/2880
μ_{PM}	Taxa de reparo das máquinas físicas	1
μ_{VM}	Taxa de reparo da máquina virtual	1

O modelo apresentado na Figura 26 permite obter uma equação de forma fechada para a disponibilidade da infraestrutura da nuvem, devido à quantidade de estados representados por esse sistema. Seria impraticável apresentá-la aqui e utilizá-la devido ao tamanho grande da equação (é preciso aproximadamente três páginas de tamanho A4 para apresentá-la). Tal fórmula tem pouco uso prático, pois, é difícil de manipular, tanto pelos humanos quanto por um solucionador simbólico (como a ferramenta Mathematica (MATHEMATICA, 2017)). Por outro lado, a equação de forma fechada generalizada proposta, conforme apresentado na Seção 5.2, é mais útil e viável para sistemas realmente grandes.

6.2.2 Resultados

Com os modelos apresentados, foram calculadas as medidas de disponibilidade e capacidade para a arquitetura representada. Em primeiro momento, é realizado uma análise comparativa entre o modelo CTMC apresentado anteriormente e o modelo matemático escalável apresentado na Seção 5.2. A Tabela 13 mostra os valores de disponibilidade no estado estacionário e disponibilidade orientada à capacidade - COA, permitindo a comparação entre o modelo CTMC e a equação em forma fechada.

Tabela 13 – Resultado da validação (Modelo CTMC x Equação de forma fechada)

Descrição do modelo	**Disponibilidade**	**COA**
Modelo CTMC	0.999999987	0.9995384300
Equação de forma fechada	0.999999987	0.9995384342

Outro resultado significativo é a comparação do tempo de execução entre o modelo CTMC e a equação de forma fechada. O tempo de execução representa o tempo gasto para calcular o COA para cenários específicos. Para esse estudo, foi utilizado dois cenários específicos. Vale ressaltar que os cenários escolhidos foram cenários pequenos, pois para o modelo CTMC ou outro formalismo, poderá ocasionar explosão de estados. O *script* utilizado para calcular o COA e o tempo de execução do modelo CTMC é apresentado no Apêndice A; a ferramenta Mercury auxilia o usuário na construção do modelo em um script, considerando o formalismo SPN, e calcula as probabilidades de cada estado usando o modelo CTMC. O primeiro (S1) é composto por uma máquina física e quatro VMs por máquina física, totalizando quatro máquinas virtuais no sistema. O segundo (S2) consiste em duas máquinas físicas e quatro VMs por máquina física, totalizando oito VMs no sistema. O terceiro cenário (S3) consiste em três máquinas físicas e quatro VMs por máquina física, totalizando doze VMs no sistema.

A Figura 27 mostra a diferença entre o modelo CTMC e a equação de forma fechada para o tempo de execução. Três cenários simples representam as principais discrepâncias em comparação com a equação de forma fechada. É importante ressaltar que o outro cenário possível com quatro máquinas físicas e quatro VMs por máquina física, totalizando

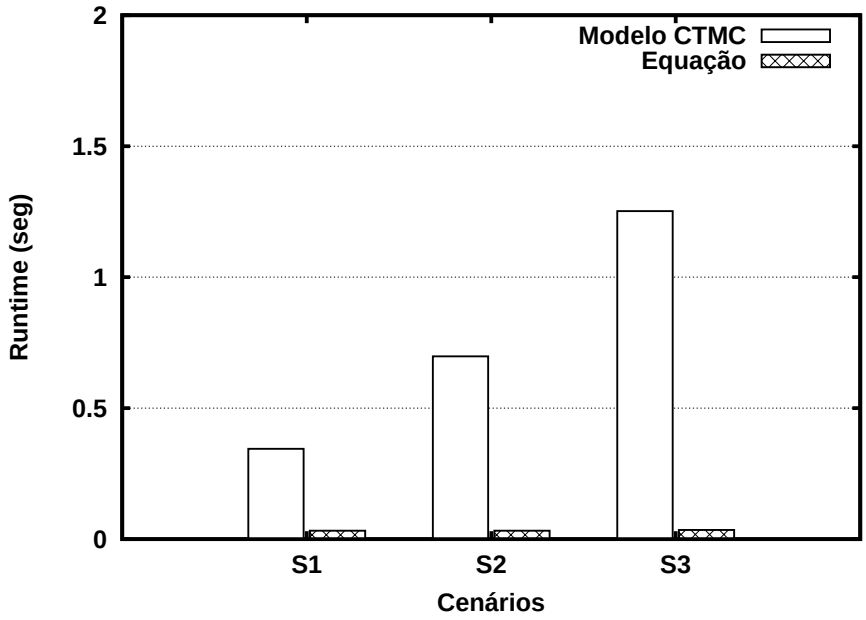


Figura 27 – Resultado do tempo de Execução

dezesseis VMs, teve um resultado de tempo de execução de 76.418 segundos no modelo CTMC, enquanto que o mesmo cenário tem 0,05 segundos no tempo de execução na equação de forma fechada. Esse resultado nos dá quase 80% de aumento no tempo de execução quando usamos um modelo CTMC. Portanto, podemos inferir que a equação gerada em nossa abordagem tem precisão equivalente, mas em um tempo de execução muito menor do que um CTMC para representar sistemas de grande escala.

A fim de fazer uso da equação de forma fechada proposta, é possível aplicar a cenários complexos com uma maior gama de possíveis tamanhos de infraestrutura; estimamos o COA para cenários com 2, 10, 50, 100 e 200 nós, cada nó com 2, 4 e 10 VMs por máquina física. A Figura 28 denota que o COA diminui à medida que o número de nós aumenta, ou seja, apesar de haver mais capacidade absoluta implantada, existe uma porcentagem de redução da capacidade real disponível para uso. Este comportamento é esperado devido ao maior número de possíveis pontos de falha. Um fenômeno semelhante é verificado para o número de VMs por nó, devido a razões semelhantes.

Vale ressaltar que, considerando o número total de VMs (ou seja, número de nodes multiplicado pelo número de VMs por node), o COA é maior para configurações com mais nodes e menos VMs por node. Um exemplo disso é verificado quando comparamos infraestruturas distintas com 200 VMs. O cenário com 50 nodes e 4 VMs por node produz um COA de 0.999513068, enquanto que 100 nodes com 2 VMs fornecem um COA de 0.999513068. Esse tipo de resultado pode ser especialmente valioso para a tomada de decisões de *designers* e administradores de sistemas de nuvem, reforçando a aplicabilidade do método de avaliação proposto. Outro resultado significativo é quando aumentamos a infraestrutura computacional para duzentas máquinas físicas e seis máquinas virtuais por

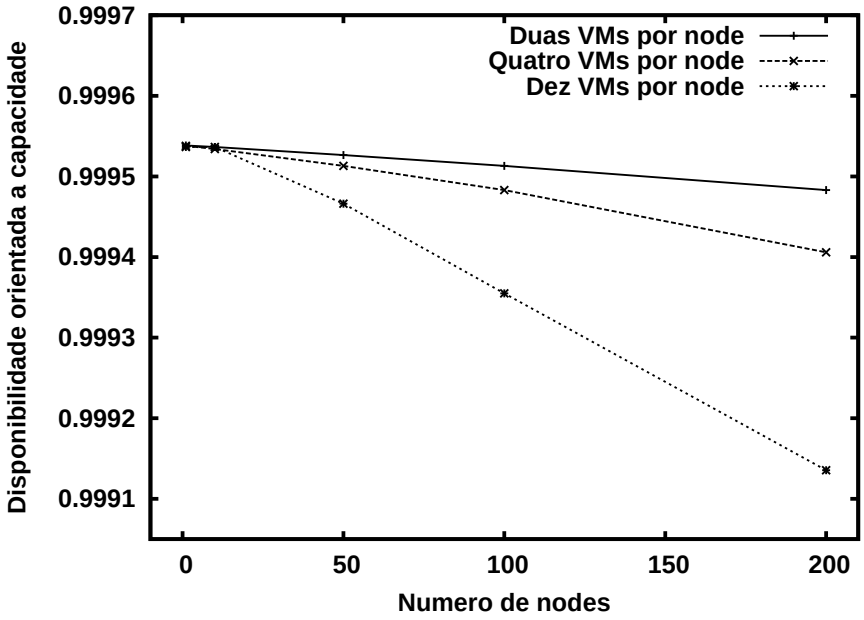
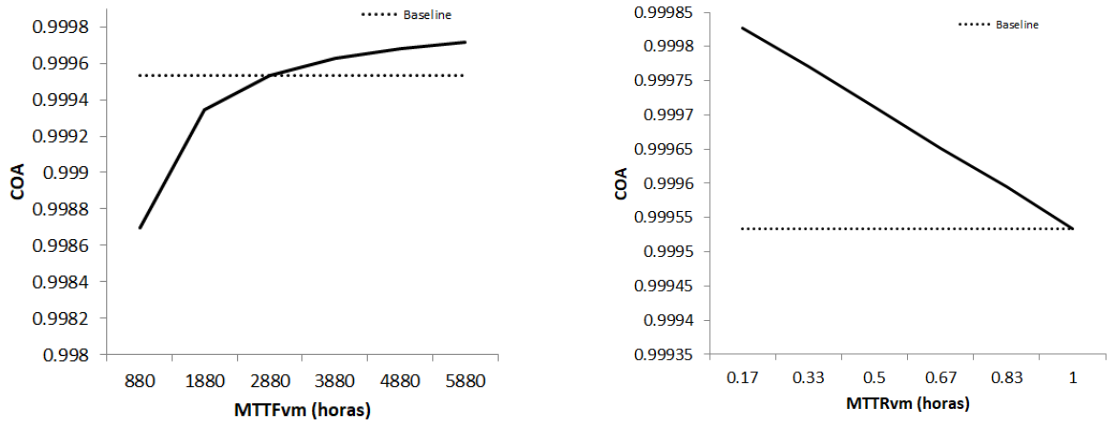


Figura 28 – Disponibilidade orientada a capacidade: Resultado por node



(a) MTTFvm (b) MTTRvm

Figura 29 – Variação dos tempos de falha e reparo da VM - COA

máquina física, totalizando mil e duzentas VM. Ao usar a equação de forma fechada, a avaliação do sistema produz um COA de 0.999291945, com um tempo de execução de apenas 13.55 horas. Um modelo CTMC com os parâmetros mencionados seria inviável para calcular devido à explosão do espaço de estados.

A fim de explorar melhor a utilização de tal equação e compreender o impacto de determinados parâmetros do modelo, utilizamos uma variação do MTTF e MTTR com compõe a VM. A arquitetura escolhida para essa análise é composta por 10 máquinas físicas e quatro VMs por máquina física, totalizando 40 VMs na infraestrutura. A Figura 29 mostra uma variação no tempo médio para falhas da VM (do inglês, *Mean Time to Failure Virtual Machine* (MTTFvm)) e o tempo médio de reparo da VM (do inglês, *Mean Time do repair Virtual Machine* (MTTRvm)).

Tomando como base o cenário, chamado aqui de *baseline*, os resultados mostram uma mudança significativa em relação ao COA quando se tem VMs mais confiáveis. Desta forma, é possível observar uma diferença de dois noves, considerando 880 horas de MTTF e três noves quando se tem 1880 horas de MTTF da VM (ver Figura 29a). Da mesma forma, quando se tem políticas de reparo bem elaboradas podemos alcançar melhores resultados de COA (ver Figura 29b), por exemplo, considerando um reparo de 10 minutos (0.17 horas), tempo suficientemente adequado para situações em que necessite, no pior dos casos, a reinicialização de todo o serviço/VM, o ambiente apresenta o melhor resultado. Considerando arquitetura um pouco mais realista, em se tratando de sistemas de nuvem, com 100 nós de computação e 2000 VMs, sendo vinte VMs por nó de computação, a diferença entre o tempo de reparo observando a capacidade do ambiente em números de VMs disponíveis é de 1999.543 para um tempo de reparo de 0.17 horas e de 1998.894 para um tempo de reparo de uma hora. Portanto, podemos notar que o COA do sistema é mais afetada quando se tem um tempo de reparo longo, chegando a perder, aproximadamente, uma VM no sistema. Isso destaca a importância de dar enfoque nas atividades de reparo do sistema já que, aumentar a confiabilidade da VM não é uma tarefa trivial.

6.3 Validação do modelo de desempenho do sistema de *VoD streaming*

Aqui, será descrito os experimentos computacionais utilizados para verificar a precisão do modelo de transcodificação de nuvem pública. Os resultados visam garantir que o modelo represente o sistema real e seja capaz de prover resultados corretos com 95% de confiança. A avaliação foi realizada em um ambiente de nuvem privada controlado, Eucalyptus. O modelo de nuvem escolhido permite que as configurações de máquinas virtuais utilizadas possam ser estendidas para nuvens públicas como a Amazon.

6.3.1 Arquitetura de testes

A infraestrutura física para validação possui dois nós de computação com CPU Xeon E3-1220V3: 4 x 3.10 GHz, 8 MB *Cache*, Memória RAM de 32 GB, 2 *NIC Gigabit Ethernet*. Foi instalado o controlador de nó *Eucalyptus* (ver Capítulo 5 nos servidores, que são usados para suportar as VMs). O ambiente também possui dois *Storage* com quatro HDs em *RAID* 5, que mantém os vídeos transcodificados. Como mostrado nos resultados apresentados na Subseção 6.1, o modelo de arquitetura escolhido para os testes corresponde a uma arquitetura com um gerenciador da nuvem (*front-end*) e os nós de computação. Desta forma, para o gerenciador da nuvem utilizamos um servidor Core i7 CPU, 4 x 3.40 GHz, 8 MB cache, 4 GB RAM, 1 *Gigabit Ethernet NIC*, juntamente com os componentes do *Eucalyptus CLC, CC, SC* e *SOS* instalados e configurados. É utilizado também, um *Switch*

com 16 portas *Gigabit Ethernet* e capacidade máxima de 32 Gbps. Como nó cliente para requisitar as transcodificações, foi utilizado um computador com processador i7 , 4 x 3.40 GHz, 8 MB Cache, 4 GB RAM, NIC 1 *GigabitEthernet*. O *software* gerador de requisições utilizado foi o *JMeter* (APACHE, 2017b). A arquitetura pode ser vista na Figura 30.

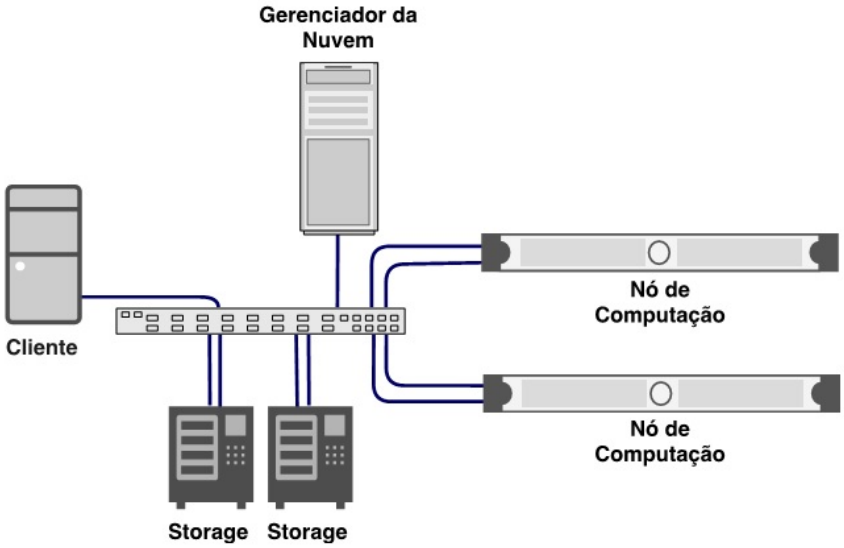


Figura 30 – Infraestrutura de testes

O sistema de transcodificação recebe as requisições do *Front-end* (gerenciador da nuvem), realiza as transcodificações e informa o estado atual para o *Front-end*. Esse subsistema usa FFMPEG (FFMPEG, 2017) para transcodificar os vídeos recebidos nos mais diversos formatos para o container MP4 com codec h264, um dos formatos recomendados para disponibilização na *Web* (DAOUST et al., 2010). Os vídeos transcodificados são armazenadas no *storage*.

O cenário de conversão de vídeo faz uso de arquivos de vídeos do mesmo tamanho, com duração de 00:04:55, *frames* do tamanho (640x356), 25 *frames* por segundo e taxa de 200kbps. Foram definidos os mesmos parâmetros, tanto para o sistema quanto para o modelo SPN (ver Figura 19).

6.3.2 Resultados experimentais e validação

A arquitetura mencionada anteriormente é suficiente para executar o sistema real, mas o modelo necessita de um importante valor: o tempo de transcodificação para uma VM do tipo **m1.small**. A fim de identificar a duração e a distribuição do tempo para realizar transcodificações, utilizamos o Jmeter para gerar a carga no sistema de transcodificação de vídeo na nuvem. O Jmeter foi configurado para enviar 60 trabalhos de conversão intercalados por 1 minuto de pausa depois do fim de cada transcodificação, então, nós obtivemos o valor do tempo de transcodificação para a transição **T1** (ver Figura 19) sem que haja influência do enfileiramento do servidor. Os resultados trouxeram um tempo médio de transcodificação de vídeo de 22.34 segundos (valor atribuído à transição **T2**).

Depois de obter o tempo médio de transcodificação de vídeo, necessário para o modelo SPN, a validação foi feita usando um tempo entre chegada de requisições, tanto para o modelo quanto para o sistema. Foi empregado um tempo entre chegada exponencialmente distribuído com 35 segundos. O *Jmeter* fez solicitações de conversões com o tempo entre chegada, para a execução de 100 vídeos que foram convertidos. Portanto, os resultados representam o comportamento do sistema considerando um total de 100 transcodificações. A Tabela 14 mostra o resultado da validação do modelo com o sistema real, para o tempo de resposta, o modelo proposto também possui resultados equivalentes ao comportamento do sistema real. O tempo entre chegadas de requisições foi atribuído à transição **T1**.

Tabela 14 – Validação por análise numérica

Taxa de entrada (s)	**Modelo (s)**	**Sistema real (s)**	**I.C. do Sistema (s)**
35	58.849	61.957	(53.312; 62.237)

6.4 Explorando o modelo de desempenho do sistema de *VoD*

O modelo concebido mostrado na Figura 19 nos dá a oportunidade de explorá-lo, criar cenários específicos para ambientes de nuvem reais. Desta forma, este estudo de caso amplia o cenário de transcodificação validado anteriormente para um cenário que comporte diferentes entradas de codecs de vídeo para conversão no sistema de *VoD* na nuvem, tendo em vista que, grande parte dos usuários desse tipo de sistema possuem câmeras e aparelhos de gravação de áudio e vídeo que salvam o arquivo bruto em diversos formatos.

A arquitetura considerada nesse estudo é a mesma apresentada na Figura 13, onde temos as entradas de arquivos de vídeo no sistema em algum formato específico (por exemplo, .vp9, .h265, .vp8). O conteúdo será convertido para o formato de vídeo H.264/MPEG-44. Todo vídeo que chegar à infraestrutura de nuvem será convertido para o formato especificado anteriormente. A infraestrutura pode fazer uso de diferentes tipos de VMs, as utilizadas nesta tese com suas respectivas configurações e diferenças, podem ser vista na Tabela 15.

Tabela 15 – Especificações das instâncias de máquinas virtuais

Tipo	**CPU**	**Memória (MB)**	**HD (GB)**
m1.Small	1	256	5
m1.Medium	1	512	5
m1.Large	2	1024	10

Para esta análise, foi necessário identificar os tempos para transcodificação de cada tipo de VM. Neste cenário, obtivemos os tempos de transcodificação para cada arquivo de

vídeo por meio de experimentos específicos. Os vídeos utilizados estão no formato **vp9**, **vp8** e **h.265** e tem duração média de 00:04:55 minutos. Para que tivéssemos o mesmo arquivo de vídeo (mesma duração e conteúdo), foi realizada a conversão do arquivo bruto **h.265** para os formatos de vídeo **vp9** e **vp8**. O resultado das transcodificações podem ser vistas na Tabela 16. Os valores da coluna, **Tempo de Transcodificação**, representam valores em segundos de conversões no sistema. Desta forma, o Jmeter foi configurado para enviar 100 trabalhos de conversão intercalados por 1 minuto de pausa, para que não houvesse fila e o sistema realizasse uma conversão por vez para cada tipo de vídeo.

Tabela 16 – Tempos das transições temporizadas para cada tipo de vídeo

Tipo de Vídeo	**Tipo de VM**	**Tempo de Transcodificação (s)**
vp9	m1.Small	28.192
	m1.Medium	25.915
	m1.Large	20.258
vp8	m1.Small	30.072
	m1.Medium	26.288
	m1.Large	20.021
h.265	m1.Small	22.340
	m1.Medium	20.021
	m1.Large	17.980

Para esse estudo de caso, foi necessário adaptar o modelo de desempenho apresentado na Seção 5.3. Desta forma, o modelo apresentado na Figura 31 apresenta a arquitetura de transcodificação de vídeos considerando um tipo de VM e os três tipos de vídeo mostrado anteriormente.

O modelo é dividido em duas sub-redes: *Arrival*, considera a chegada de trabalhos na infraestrutura; e *Transcode*, responsável pela fila e processo de conversão de vídeos. A sub-rede *Arrival* é composta por dois lugares **P1** e **K0**, que representam a espera entre chegada de trabalhos a depender do tamanho do *buffer* no lugar **K0**, em conjunto com a transição **T1**. Quando a transição **T1** está habilitada, um *token* é consumido do lugar **K0** e é depositado no lugar **P1**, representando a chegada de vídeo para ser convertido. A transição **T1** representa uma transição temporizada com tempos entre chegadas de arquivos de mídia exponencialmente distribuídos. Essa suposição pode ser modificada, alterando essa distribuição. Outra consideração importante é que o tempo associado à transição **T1** leva em conta apenas o tempo que as transações entram no sistema, ou seja, não são levados em conta as perdas provenientes da rede.

Ao chegar trabalhos a serem convertidos no lugar **P1**, a transição imediata **TI1** (simbolizada por um retângulo preto, logo não terá atraso associado) será habilitada. Assim, será disparada, logo que estiver habilitada. Quando **TI1** dispara a sub-rede de conversão (**transcode**) é alcançada, um *token* é retirado do lugar **P1** e **K2**, representando

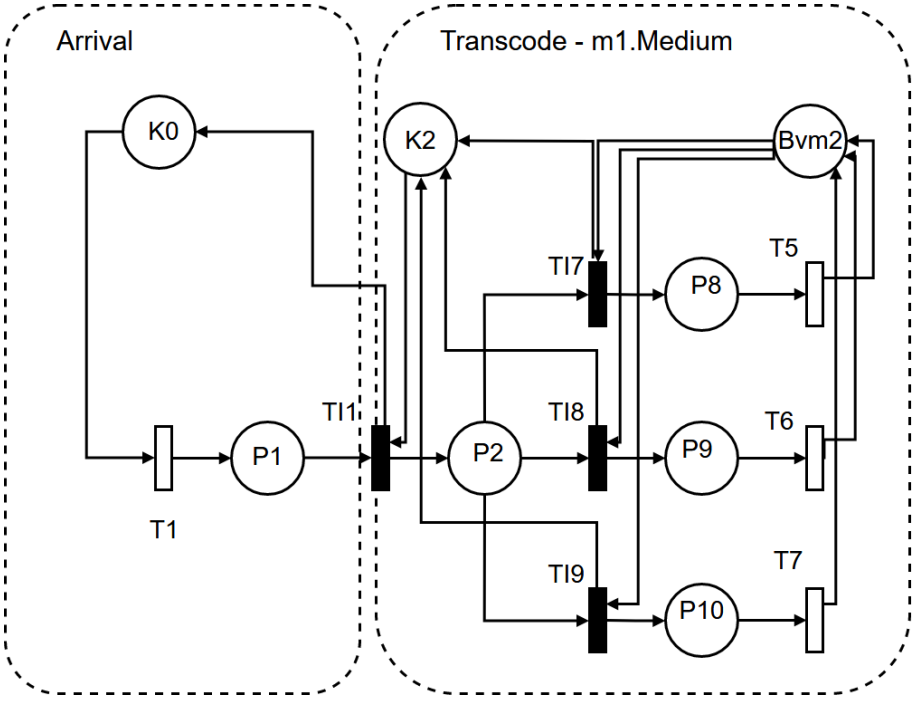


Figura 31 – Modelo SPN para transcodificação de diferentes tipos de vídeo

a chegada do arquivo de vídeo a ser convertido e a fila do sistema de transcodificação, respectivamente, e depositado no lugar **P2** e **K0**, o vídeo chegou ao sistema de conversão e a retomada do tamanho do *buffer* no sistema, possibilitando a chegada de novos vídeos. As transições imediatas **TI7**, **TI8** e **TI9** representam probabilidades do usuário enviar um vídeo de um determinado tipo (vp8, vp9 e h.265 respectivamente). Esses formatos de vídeo foram escolhidos devido ao atual cenário de gravações de vídeos em altas definições. Desta forma, as três transições estarão habilitadas e o disparo da transição será realizado a partir da probabilidade atribuída a cada transição e tão logo houver *tokens* nos lugares **P2** e **Bvm2** (seguindo as condições de disparo). Assim que **TI7** é disparada, um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P8** e **K2** que representa o processo de conversão para o formato de vídeo vp8 e a retomada do tamanho da fila do sistema. Assim que **TI8** é disparada, um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P9** e **K2** que representa o processo de conversão para o formato de vídeo vp9 e a retomada do tamanho da fila do sistema. Assim que **TI9** é disparada um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P10** e **K2** que representa o processo de conversão para o formato de vídeo h.265 e a retomada do tamanho da fila do sistema, sempre respeitando a política de disparo da transição. A quantidade de *tokens* no lugar **P2** representa o enfileiramento de requisições, o enfileiramento ocorre quando não há capacidade disponível para servir a requisição recém-chegada, representada pelo lugar **K2**. Se houver unidade de processamento, representada pelo lugar **Bvm2**, disponível, as transições **TI7**, **TI8** e **TI9** podem disparar e posteriormente a requisição é processada. As transições **T5**, **T6** e **T7** representam transições temporizadas referente ao *delay* no

processo de conversão para cada tipo de arquivo de vídeo.

6.4.1 Parâmetros de entrada

A Tabela 17 apresenta, de forma condensada, os parâmetros de entrada para o modelo SPN apresentado anteriormente. Esses parâmetros levam em consideração o tipo de VM **m1.small**, **m1.medium** e **m1.large**. Os valores de probabilidade podem ser estimados, para os formatos de vídeo **vp9** e **vp8** do *google* como sendo valores com usabilidade inferior, considerando câmeras e dispositivos que gravem vídeos nesse formato, do que formatos de vídeos usuais como **h.265**, **h.264**, por exemplo, usamos probabilidades menores do que o **h.265**.

Tabela 17 – Parâmetros de entrada para as transições temporizadas

Transição	**m1.small (s)**	**m1.medium (s)**	**m1.large (s)**	**Server semantics**
T1 (chegada)	30, 35 e 40	30, 35 e 40	30, 35 e 40	Single server
T5 (vp9)	28.192	25.915	20.258	Infinite server
T6 (vp8)	30.072	26.288	20.021	Infinite server
T7 (h.265)	22.340	20.021	17.980	Infinite server

Assim como as transições temporizadas, temos as transições imediatas. A Tabela 18 apresenta uma descrição com os valores de entrada para cada transição imediata. Vale destacar que os valores para as transições imediatas não sofrem alterações de acordo com o tipo de VM utilizada. O tempo de resposta pode ser calculado seguindo a Equação 5.18 apresentada na seção anterior.

Tabela 18 – Parâmetros de entrada para as transições imediatas

Transição	**Peso**	**Prioridade**
TI1	1	1
TI7 (vp8)	0.15	1
TI8 (vp9)	0.35	1
TI9 (h.265)	0.50	1

6.4.2 Resultados

Foi calculado o tempo de resposta para transcodificação dos vídeos em diversas situações. Consideramos o tempo entre chegadas de solicitações de vídeo para conversão e tempo entre tanscodificações, os valores apresentados anteriormente para os três tipos de VM configuradas, **m1.small**, **m1.medium** e **m1.large**. A Figura 32 apresenta o resultado de transcodificação fazendo uma comparação entre os tipos de VM e considerando o tempo

entre chegada de conversão. Como era de se esperar, esse resultado nos mostra a VM do tipo **m1.large** como sendo a melhor no processo de transcodificação dos vídeos devido à configuração superior.

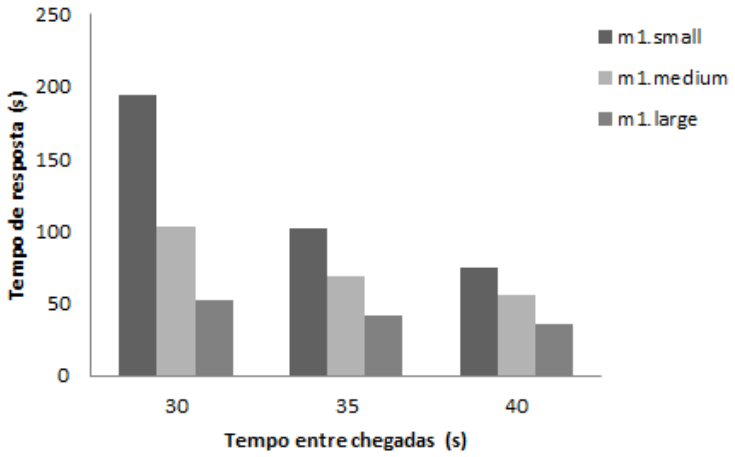


Figura 32 – Tempo de transcodificação considerando tempo entre chegadas e VMs diferentes

Com base nesses resultados e com a formulação matemática para o cálculo da disponibilidade apresentada na Equação 5.14, podemos conhecer o número médio de trabalhos descartados devido a atividades de falha do nó de computação ou da VM. Consideramos como tempo entre chegadas de vídeo o valor de 35 segundos e a VM do tipo **m1.small**. Desta forma, foram utilizados os parâmetros de entrada apresentados na Tabela 12 para os valores de falha e reparo do sistema; variamos o tempo de falha da VM em 300 horas partindo de um MTTF de 780 horas até um MTTF de 2880 horas. De acordo com as Equações 6.4 e 6.5, é possível conhecer o descarte médio para um período de tempo determinado.

$$\lambda_D = THx(1 - A) \tag{6.4}$$

$$DM = \lambda_D x Time \tag{6.5}$$

onde, λ_D representa a taxa de descarte de requisições de vídeo, enquanto **TH** o throughput do sistema, **A** representa a disponibilidade do sistema, **Time** o tempo de observação no sistema e **DM** representa o descarte médio de solicitações de vídeo no sistema.

A Figura 33 apresenta o resultado considerando o descarte médio do sistema para um período de um mês, seis meses e um ano de observação. É importante observar o comportamento do sistema devido à atividade de falha da VM; com a variação do MTTF da VM, observamos que o descarte tende a estacionar. É possível concluir que a adoção de sistemas mais confiáveis pode diminuir o número médio de descarte no sistema.

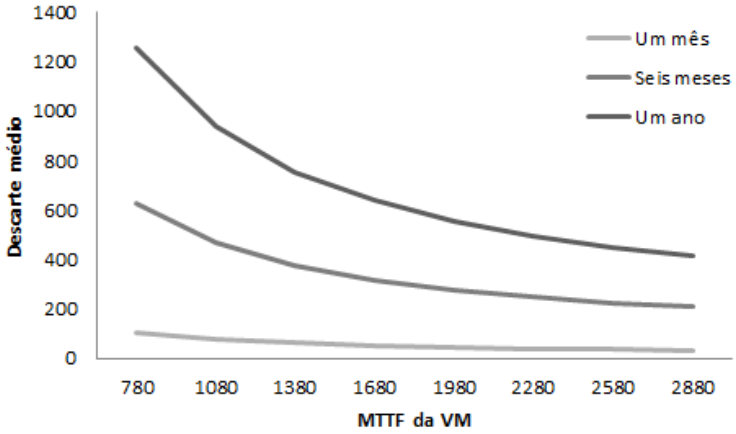


Figura 33 – Descarte médio de transações de vídeo

Análise de custo

A fim de comparar os custos relacionados à arquitetura de nuvem, realizamos uma comparação entre uma arquitetura de nuvem *pública* versus uma nuvem *privada*. A Tabela 19 apresenta os custos de uma nuvem pública considerando instâncias do tipo reservada na Amazon.

Tabela 19 – Custo médio de instâncias do tipo reservadas

Tipo de VM	**1 ano (reservada US$)**	**3 anos (reservada)**
m1.small	137	411
m1.medium	275	825
m1.large	549	1,647

Para os custos de uma nuvem privada, usamos os custos relacionados a um servidor PowerEdge R430 Rack Server com CPU Intel Xeon E5-2650 V4 2.2GHz 30M Cache e memória de 16GB (8x8) RDIMM a um custo de US$ 2,487.11. Uma máquina desse tipo, de acordo com a infraestrutura de nuvem estabelecida, pode suportar até 2 VMs do tipo **m1.large**, 4 VMs do tipo **m1.medium** e 8 VMs do tipo **m1.small**.

Desta forma, assumindo um SLA de 30 segundos de tempo de resposta para quaisquer que seja o tipo de VM utilizada na infraestrutura e um tempo entre chegadas de requisições de 10 segundos, é possível estimar a quantidade de VM utilizada na infraestrutura, bem como, os custos relacionados. A Tabela 20 apresenta a quantidade de VM utilizada para o processo de transcodificação, assim como os custos relacionados. Embora, para esse pequeno cenário, a VM do tipo **m1.small** apresente um menor custo, ela apresenta o pior tempo de resposta e considera o uso de mais instâncias reservadas.

De acordo com a Seção 6.1, podemos verificar que para uma infraestrutura privada são necessários ao menos dois servidores (um *Front-end* e um nó), a depender da quantidade de VMs a ser utilizada que acarreta o crescimento da quantidade de nós. Levando em

Tabela 20 – Custo médio de instâncias reservadas considerando processo de transcodificação

Tipo de VM	**Quantidade VM**	**Tempo de resposta médio**	**1 ano (reservada US$)**
m1.Large	2	21.41	1098
m1.medium	4	26.58	1100
m1.Small	5	27.13	685

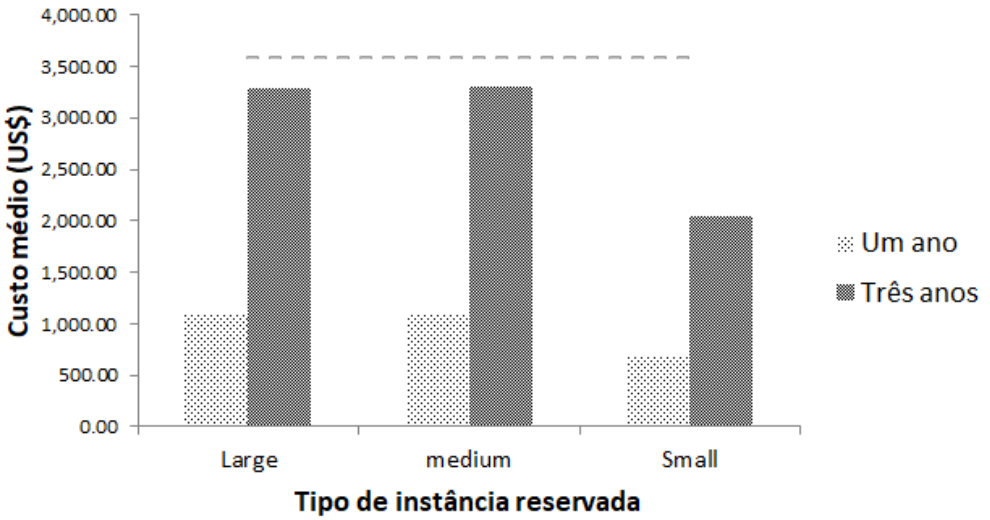


Figura 34 – Comparação entre os custos médios de uma nuvem privada vs nuvem pública

consideração a quantidade de VMs que podem ser instânciadas no servidor apresentado anteriormente, uma única máquina física desse tipo é suficiente para a quantidade de VMs da Tabela 20. O servidor que funciona como *Front-end* pode ter características e configurações menores do que o servidor que funciona como nó. Desta forma, o servidor *Front-end* é um PowerEdge R330 Rack Server com CPU Intel Xeon E3-1220 v5 3.0GHz e 8M cache e memória de 8GB, 500GB HD a um custo de US$ 1,099.00. A Figura 34 apresenta uma relação do custo médio por tipo de instância reservada fazendo uma comparação com os custos de uma infraestrutura privada. Podemos notar que para uma pequena infraestrutura de computação é preferível manter um aluguel de uma infraestrutura pública por até três anos do que manter uma infraestrutura privada por igual período (fato comprovado através da linha tracejada no gráfico que representa a infraestrutura privada). Vale ressaltar que os custos relacionados para infraestrutura privada leva em consideração apenas os custos com aquisição de máquinas físicas, deixando de envolver vários outros custos relacionados.

6.5 Planejamento da infraestrutura de *VoD* na nuvem

Este estudo de caso tem como objetivo planejar uma infraestrutura de nuvem, seja ela pública ou privada, considerando um ambiente de pequeno porte, como é o caso de instituições EaD ou pequenas empresas que fazem uso de sistemas de vídeo sob demanda. Para esse estudo, usamos como base os resultados apresentados anteriormente seguindo a metodologia de apoio descrita na Seção 4.1.

A arquitetura usada pode ser vista na Figura 35. Para o planejamento da infraestrutura de nuvem é considerado três tipos de VMs (**m1.small**, **m1.medium** e **m1.large**) mostrado anteriormente. A infraestrutura contém um **gerente da nuvem** e N **nodes**. Para cada nó pode existir até, 8 VMs do tipo **m1.large**, 4 VMs do tipo **m1.medium** e 2 VMs do tipo **m1.small** instanciadas. Para o planejamento da infraestrutura de nuvem consideramos métricas como: **disponibilidade**, **desempenho** e **custo**. A disponibilidade é considerada quando se tem uma infraestrutura de nuvem privada e os custos levados em consideração são apenas os relacionados à aquisição de equipamento físico (nuvem privada) e aluguel de instâncias reservadas (nuvem pública).

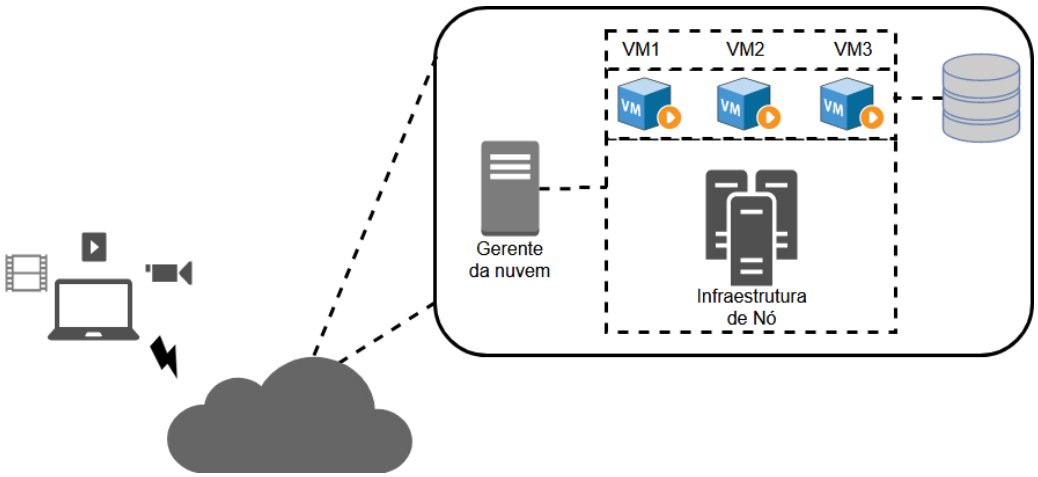


Figura 35 – Arquitetura de desempenho considerando três tipos de VMs

Inicialmente, foram realizados experimentos com o objetivo de identificar a quantidade de vídeos que uma VM é capaz de processar de forma simultânea, ou seja, sem que aconteça o descarte natural do sistema ou o travamento da aplicação de conversão de vídeo devido ao grande número de solicitações (número de *threads* simultâneas é suficientemente grande para processar uma quantidade média de vídeos no processo de conversão).

6.5.1 Modelo de desempenho para o planejamento

É proposto um modelo SPN com as características dos modelos apresentados na Seção 5, o qual foi reestruturado para o problema proposto. Neste cenário, temos uma composição dos três tipos de VMs no sistema e consideramos a probabilidade de utilização das VMs (**transcode - m1.large**, **transcode - m1.medium** e **transcode - m1.small**). A Figura

36 apresenta esse modelo cujo propósito é calcular o tempo médio de resposta para a atividade de conversão de vídeo, entretanto, nada impede que outras métricas sejam obtidas. Os significados dos nomes utilizados no modelo estão na Tabela 21 e a descrição dos atributos das transições na Tabela 22. Os parâmetros da distribuição das transições temporizadas devem ser providos pelo usuário do modelo, o qual representa o processo de transcodificação considerando três tipos de VMs. Nesse caso, cada tipo de VM possuirá diferentes tempos para transcodificação e os valores podem ser obtidos através de medidas em um sistema já hospedado na nuvem, ou considerando cargas de trabalho esperadas em diferentes VMs em um sistema privado.

Tabela 21 – Descrição dos places e transições temporizadas para o modelo SPN - 2

Transição	**Descrição**
T1	Tempo entre chegada de trabalhos
P1	Espera pela disponibilidade da fila da VM
P2, P3 e P4	Trabalhos na fila (para VM1, VM2 e VM3)
P5 a P13	Transcodificações sendo realizadas
Bvm1, Bvm2 e Bvm3	Recursos de processamento disponíveis (por VM)
T2 a T10	Tempo para transcodificação (diferente por tipo de vídeo e VM)
K1, K2 e K3	Capacidade máxima da fila por VM
K0	Capacidade máxima da fila no sistema

Para as transições imediatas que recebem **Peso = dinâmico**, implica em dizer que os valores referente ao peso das transições pode variar, a depender do método de avaliação ou dos cenários e perfis de usuário que fazem uso do sistema.

Tabela 22 – Descrição das transições para o modelo SPN - 2

Transição	**Tipo**	**Server Semantic**	**Peso**	**Prioridade**
T1	Temporizada	Single Server	-	-
T2 a T10	Temporizada	Infinite Server	-	-
TI1	Imediata	-	Dinâmico	1
TI2	Imediata	-	Dinâmico	1
TI3	Imediata	-	Dinâmico	1
TI4=TI7=TI10=VP9	Imediata	-	Dinâmico	1
TI5=TI8=TI11=VP8	Imediata	-	Dinâmico	1
TI6=TI9=TI12=h264	Imediata	-	Dinâmico	1

O modelo é dividido em duas sub-redes: *Arrival*, considera a chegada de trabalhos na infraestrutura; enquanto *Transcode* por tipo de VM é responsável pela fila e processo de conversão de vídeos para cada tipo de VM. A sub-rede *Arrival* é composta por dois lugares **P1** e **K0**, que representam a espera entre chegada de trabalhos e a aceitação desse trabalho

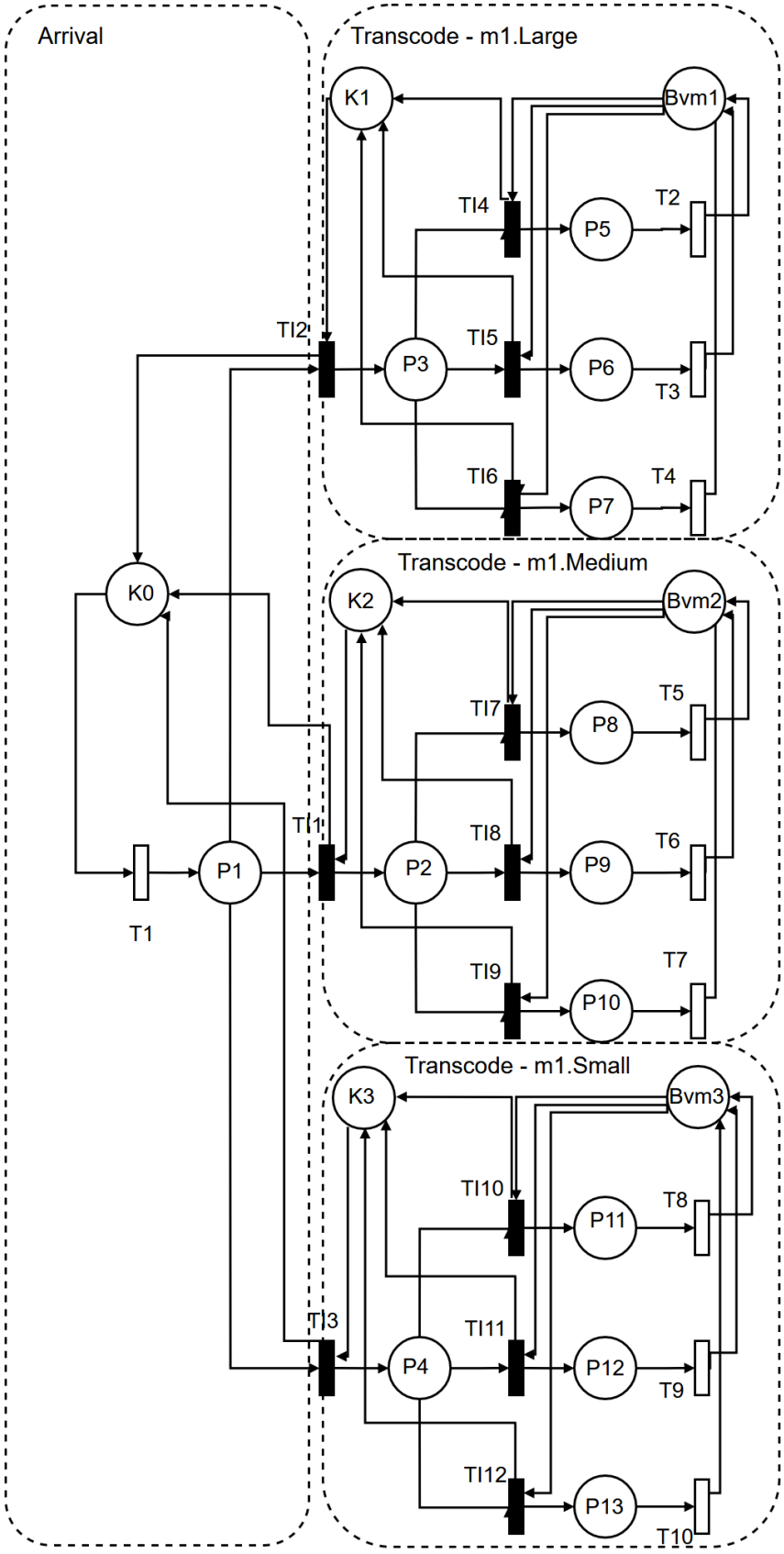


Figura 36 – Modelo de desempenho do sistema *VoD* para o planejamento de infraestrutura de Nuvem

na fila, respectivamente, em conjunto com a transição **T1**. Quando a transição **T1** está habilitada, um *token* é consumido do lugar **K0** e depositado no lugar **P1**, representando a chegada de vídeo para ser convertido. A transição **T1** representa uma transição temporizada com tempos entre chegadas de arquivos de mídia exponencialmente distribuídos. Essa suposição pode ser modificada, alterando essa distribuição. Outra consideração importante é que o tempo associado à transição **T1** leva em conta apenas o tempo que as transações entram no sistema, ou seja, não são levados em conta as perdas provenientes da rede.

Ao chegar trabalhos a serem convertidos no lugar **P1**, as transições imediatas **TI1, TI2 e TI3** (simbolizadas por um retângulo preto, logo não terá atraso associado) estarão habilitadas e será disparada com maior frequência aquela com **Peso** maior ou probabilidade de utilização maior, de uma VM de determinado tipo. A transição que será disparada é a transição que representa uma maior probabilidade de utilização de uma VM de um determinado tipo (**TI1** - m1.Medium, **TI2** - m1.Large, etc.), por exemplo, se **TI1** dispara, a sub-rede de conversão (**transcode - m1.medium**) é alcançada, um *token* é retirado do lugar **P1** e **K2** e depositado no lugar **P2**. Vale lembrar que um token é devolvido ao lugar **K0**, representando a fila do sistema. As transições imediatas **TI7**, **TI8** e **TI9** representam probabilidades do usuário enviar um vídeo de um determinado tipo (vp9, vp8 e h.265 respectivamente). Esses formatos de vídeo foram escolhidos devido ao atual cenário de gravações de vídeos em altas definições. Desta forma, as três transições estarão habilitadas e o disparo da transição será realizado a partir da probabilidade atribuída a cada transição e tão logo se houver *tokens* nos lugares **P2** e **Bvm2** (seguindo as condições de disparo). Assim que **TI8** é disparada, um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P9**, que representa o processo de conversão para o formato de vídeo vp8. Assim que **TI8** é disparada, um *token* é devolvido ao lugar **K2**, representando a composição da fila no sistema. Assim que **TI7** é disparada, um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P8** que representa o processo de conversão para o formato de vídeo vp9. Assim que **TI9** é disparada, um *token* é consumido dos lugares **P2** e **Bvm2** e depositado no lugar **P10** que representa o processo de conversão para o formato de vídeo h.265, sempre respeitando a política de disparo da transição. A quantidade de *tokens* no lugar **P2** representa o enfileiramento de requisições para o tipo de VM **m1.medium**; o enfileiramento ocorre quando não há capacidade disponível para servir a requisição recém-chegada, representada pelo lugar **K2**. Se houver unidade de processamento disponível, as transições **TI7**, **TI8** e **TI9** podem disparar e posteriormente a requisição é processada. O processo de conversão para os tipos de VM **m1.large** e **m1.small** seguem o mesmo padrão de processamento. As transições de **T2** à **T10** representam transições temporizadas referente ao *delay* no processo de conversão para cada tipo de arquivo de vídeo e para cada tipo de VM associada.

6.5.2 Parâmetros de entrada

Como as transições temporizadas do tempo de transcodificação dependem do tipo de VM e da quantidade de trabalhos simultâneos que essa VM é capaz de realizar, necessitamos medir esses valores para as VMs estudadas em um sistema real. Utilizamos a infraestrutura apresentada anteriormente para medir em cada tipo de VM os tempos de transcodificação, que foram obtidos com 60 requisições de transcodificações intervaladas de 1 minuto. Dessa maneira, a chegada de 60 processos é simulada quase que simultaneamente, na infraestrutura.

A quantidade de *threads*, em processamento simultâneo no sistema, foi o limiar para estabelecer a quantidade de trabalhos que uma VM é capaz de processar. A Tabela 23 apresenta a quantidade de processos que uma VM de um determinado tipo é capaz de processar com os seus respectivos valores de tempo de resposta em segundos; os valores foram obtidos por meio de experimentos específicos. O critério de parada para estabelecer o número correspondente à quantidade de trabalhos que uma VM é capaz de processar de forma simultânea é o conjunto de *threads* de conversão e a fila no sistema. Ao chegar trabalhos de conversão, o FFMPEG faz uma subdivisão de tarefas e realiza as conversões de forma simultânea, com ajuda do paralelismo e da capacidade de processamento do processador.

Desta forma, se o sistema estiver com todas as *threads* ocupadas e um outro processo na fila de espera, esse será estabelecido como quantidade de trabalhos que uma VM suporta em processos simultâneos. A quantidade de processos simultâneo para VM do tipo **m1.medium** e **m1.small** é quase o mesmo valor devido à diferença da VM ser apenas a quantidade de memória. Esses valores devem ser atribuídos aos lugares **Bvm1**, **Bvm2** e **Bvm3** (ver o modelo da Figura 36). Devido às questões do paralelismo do sistema operacional e a quantidade de núcleos (unidade de processamento) da VM, o tempo de resposta para uma quantidade de vídeo no processo de conversão tende a crescer. Para os valores apresentados na Tabela 23 referem-se ao tempo de processamento do vídeo h.265, vp8 e vp9 que são atribuídas às transições de **T2** a **T10**.

Tabela 23 – Quantidade de trabalhos simultâneos por tipo de VM

Tipo de VM	**Quantidade de trabalhos**	**h.265 (s)**	**vp8 (s)**	**vp9 (s)**
m1.Large	18	119	131	127
m1.Medium	10	138	144	142
m1.Small	8	146	161	155

A Tabela 24 apresenta os valores para as transições imediatas que representam as probabilidades de chegada de vídeo de um determinado tipo para serem convertidos pela VM. Para isso, foram construídos cenários com base em perfis de usuário, nos quais, para um determinado perfil, podemos ter maior incidência de envio de vídeos de um determinado

tipo. A tabela apresenta seis perfis de usuário, bem como, os respectivos pesos para as transições, considerando as transições imediatas de **TI4** a **TI12**.

Tabela 24 – Parâmetro de entrada para as transições imediatas considerando perfis de usuário

Perfis de usuário	**Transições**	**Peso h.265**	**Peso vp8**	**Peso vp9**
I	TI4 a TI12	0.50	0.35	0.15
II	TI4 a TI12	0.85	0.1	0.05
III	TI4 a TI12	0.50	0.50	0.00
IV	TI4 a TI12	0.50	0.25	0.25
V	TI4 a TI12	0.25	0.25	0.50
VI	TI4 a TI12	0.33	0.33	0.33

Para a transição temporizada **T1** é atribuído o valor de 20 segundos considerando o tempo entre chegada de trabalhos a serem convertidos na infraestrutura de nuvem. Para as transições imediatas **TI1**, **TI2** e **TI3**, os valores de probabilidade serão atribuídos pelo algoritmo GRASP descrito na Seção 5.5. Esse processo é importante para o algoritmo, pois é ele quem irá determinar a melhor configuração e tipo de VM para a infraestrutura, processo que seria lento se realizado manualmente.

6.5.3 Resultados

Este resultado demonstra a combinação do modelo SPN, modelo matemático para o COA, modelo de custo e o modelo de otimização apresentados no Capítulo 5. O algoritmo de otimização irá explorar o espaço de configuração de três parâmetros (dependentes da probabilidade de utilização para cada tipo de VM). Portanto o GRASP identificará: o tipo de VM a ser usado; a quantidade de VMs reservadas a serem contratadas; a quantidade de computadores físicos a serem utilizadas na infraestrutura privada (dependente das VMs reservadas) e o *step-size* que devem ser configurados para minimizar o custo enquanto também cumpre o SLA definido pelo tempo médio de resposta.

Nós consideramos um cenário onde a carga de trabalho esperada é composta pelos vídeos de mesmas durações e formatos diferentes descritos anteriormente, com tempo entre requisições de 20 segundos (exponencialmente distribuídos). O processo de otimização deverá identificar a configuração que deverá ser utilizada na nuvem para minimizar o custo do sistema considerando um SLA de 150 segundos e um *range* de máquinas físicas de 1 a 10. Um SLA de 150 segundos permite que vários trabalhos possam ser convertidos em uma única VM como visto na Tabela 23.

A Tabela 25 apresenta o resultado considerando o número de VMs utilizadas por perfil de usuário (ver perfis na Tabela 23). Considerando uma nuvem privada, os perfis **I**, **IV**, **V** e **VI** apresentaram os mesmos tamanhos (*Size Infra*) em número de máquinas físicas. É

possível conhecer o *Size Infra* para cada perfil considerando os cálculos apresentados no Capítulo 5, na Seção 5.4.

Tabela 25 – Resultados em número de VMs para cada perfil de usuário - planejado

Tipo de VM	**I**	**II**	**III**	**IV**	**V**	**VI**
m1.Small	30	0	0	26	28	36
m1.Medium	3	0	0	5	4	0
m1.Large	1	6	4	1	1	1
Size Infra	5	3	2	5	5	5

A Figura 37 apresenta os resultados do custo considerando os perfis de usuário. O **perfil III** apresenta um melhor resultado, pois faz uso de apenas duas máquinas físicas (como nó de computação). O algoritmo achou mais vantajoso fazer uso de seis máquinas virtuais do tipo **m1.large** para esse perfil que faz uso de, 50% de vídeos do tipo **h.265**, 50% do tipo **vp8** e 0% do tipo **vp9**.

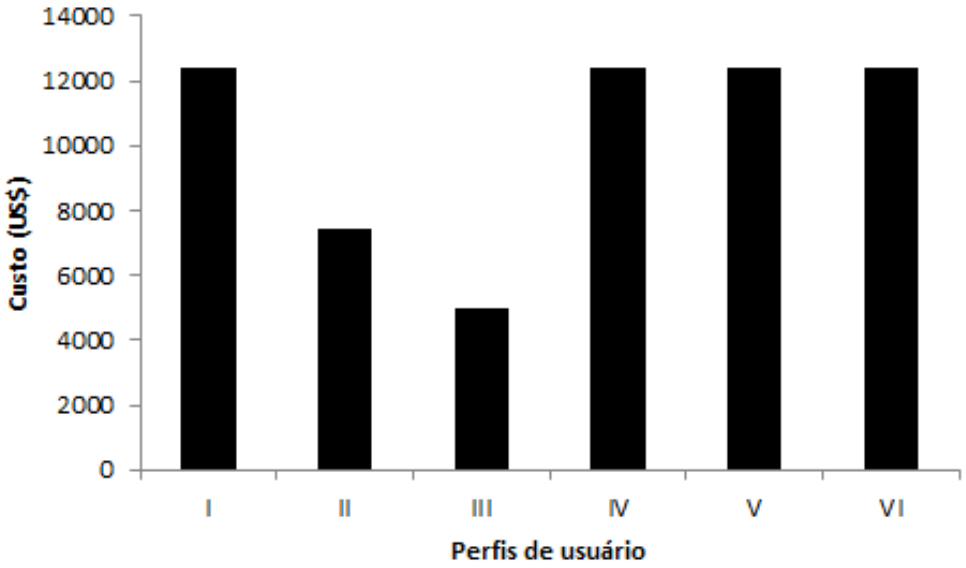


Figura 37 – Custo da infraestrutura privada

Fazendo uso dos valores da Tabela 19, na qual apresenta os custos relacionados às instâncias do tipo reservadas em uma nuvem pública, podemos fazer uma relação entre o custo médio em se alugar uma nuvem pública com os custos médios em se ter uma infraestrutura privada. É relevante lembrar que os custos relacionados à infraestrutura privada levam em consideração, apenas, os custos com aquisição de máquinas físicas. A Figura 38 apresenta essa relação de custos.

Os resultados indicam que o custo de nuvens públicas é mais atraente para os serviços que se destinam a trabalhar por um curto período de tempo. O **perfil III** se mantém com o perfil mais vantajoso no quesito custo. Assim como o **perfil III**, o **perfil II** apresenta o

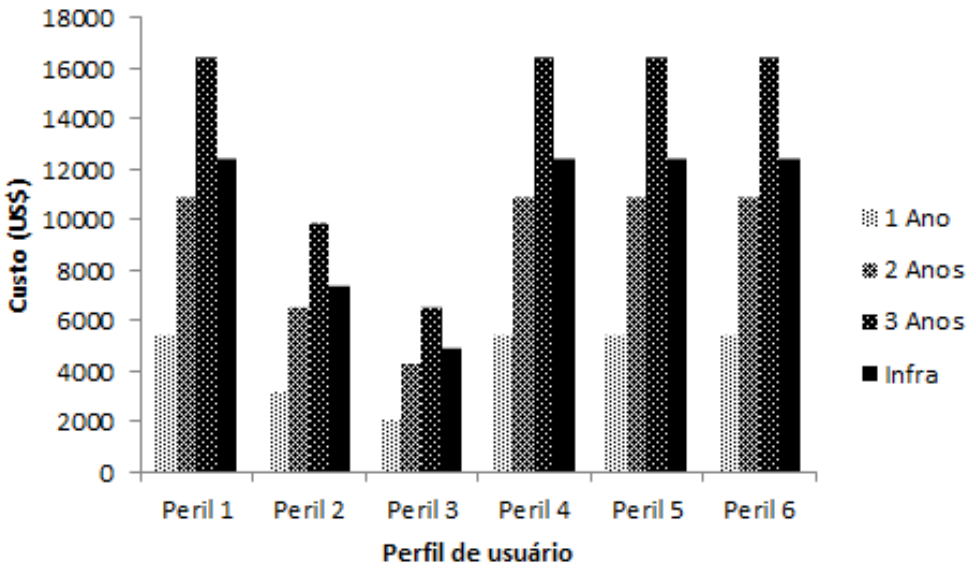


Figura 38 – Custos: Nuvem pública x nuvem privada

segundo lugar no *ranking* dos custos, isso porque o **perfil II** chega a enviar uma maior quantidade de vídeos do tipo **h.265** e uma menor quantidade dos tipos **vp8** e **vp9**, o que exige um maior tempo de processamento.

O algoritmo de otimização nos auxilia no planejamento de uma infraestrutura considerando um ambiente de *VoD streaming*. Com as características e resultados apresentados nas seções anteriores, é possível notar que com um conjunto de parâmetros e modelos complexos o algoritmo chega a resultados otimizados e satisfatórios, diferentemente da avaliação realizada manualmente, o que tornaria o método custoso e lento. O Algoritmo de otimização tenta organizar uma melhor combinação de máquinas virtuais, a depender do perfil de usuário, e montar uma melhor infraestrutura de máquinas físicas que atendam a quantidade de VMs esperada. Assim, de acordo com a Tabela 25 é possível notar a quantidade de instâncias virtuais para cada perfil de usuário. Desta forma, podemos verificar através da Figura 39 uma comparação dos **custos** com o **tempo de resposta** das infraestruturas, considerando um ambiente planejado (*planning*) com o algoritmo de otimização e uma outra infraestrutura não planejada, considerando apenas, o número de máquinas virtuais necessário no processo de transcodificação de vídeo por perfil.

Para esse processo, nós consideramos um único conjunto de VMs de determinado tipo. Por exemplo, como mostrado na Tabela 25, para o **perfil I** são necessárias 30 máquinas do tipo *m1.small*, três do tipo *m1.medium* e uma do tipo *m1.large*, fazendo uso de cinco máquinas físicas para o **Size Infra**, considerando uma infraestrutura planejada (*planning*), escolhida através do algoritmo de otimização. Assim, para uma infraestrutura não planejada, o usuário do **perfil I** pode optar por ter até 34 VMs de um mesmo tipo. A Tabela 26 apresenta os valores correspondentes à quantidade de VM escolhida por tipo e o **Size Infra** para cada infraestrutura **não planejada**.

Tabela 26 – Resultados em número de VMs para cada perfil de usuário - não planejado

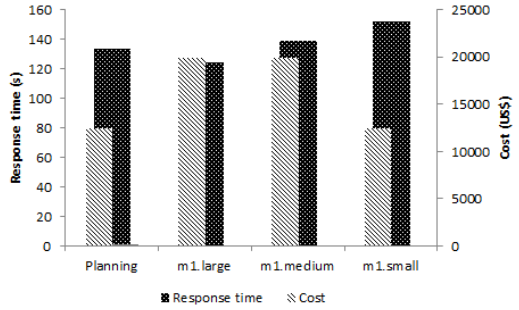
Tipo de VM	**I**	**II**	**III**	**IV**	**V**	**VI**
(A) m1.small	34	16	12	32	32	38
(B) m1.medium	30	12	8	28	28	34
(C) m1.large	15	6	4	14	14	16
Size Infra (A/B/C)	5/8/8	2/3/3	2/2/2	4/7/7	4/7/7	5/9/8

Em todos os cenários, a arquitetura planejada por meio do algoritmo de otimização leva vantagem quando comparada às arquiteturas que não tenham um planejamento adequado, seja essa vantagem no custo ou no tempo de resposta. A Figura 39b apresenta resultados semelhantes para o **perfil II** considerando as arquiteturas *planning* e m1.large, já que representam-se como mesma arquitetura, tanto para o resultado da planejada quanto para o resultado de uma arquitetura em que se escolha tipo único de VMs. Para os resultados apresentados em 39d e 39e o gráfico tem o mesmo resultado devido ao conjunto de máquinas físicas que não se alteram para as arquiteturas planejadas e não planejadas, considerando os dois perfis. No quesito custo, o **perfil III** é o mais vantajoso e os **perfis I** e **IV** apresentam piores tempos de resposta, por esse motivo, é necessário maior tempo de processamento, já que faz uso dos tipos de vídeo **vp8** e **vp9**.

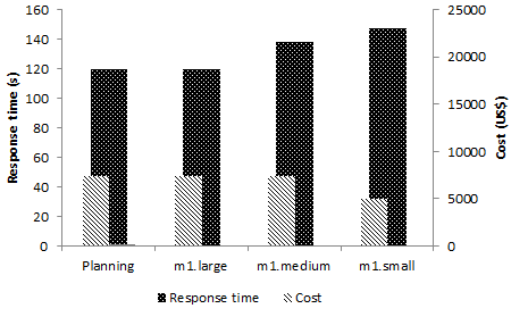
Com o objetivo de comprovar a viabilidade do algoritmo proposto, foi realizado um estudo preliminar com base no menor ambiente possível. Um único ambiente pode gerar várias arquiteturas com base na quantidade e tipo de VMs que nela será apresentada. As arquiteturas serão compostas por uma máquina física, na qual, comporta até oito máquinas virtuais do tipo *small*, ou quatro máquinas virtuais do tipo *medium*, ou duas máquinas virtuais do tipo Large, ou ainda, uma mistura dos três tipos de máquinas virtuais possíveis de acordo com a capacidade da máquina física. A Tabela 27 apresenta a distribuição de VMs para as arquiteturas.

Tabela 27 – Distribuição de VMs considerando uma máquina física

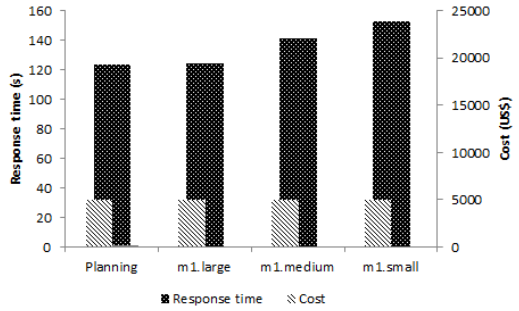
Arquitetura	Quantidade de VM (por tipo)		
	Large	Medium	Small
A1	2	0	0
A2	1	2	0
A3	1	1	2
A4	1	0	4
A5	0	4	0
A6	0	3	2
A7	0	2	4
A8	0	1	6
A9	0	0	8



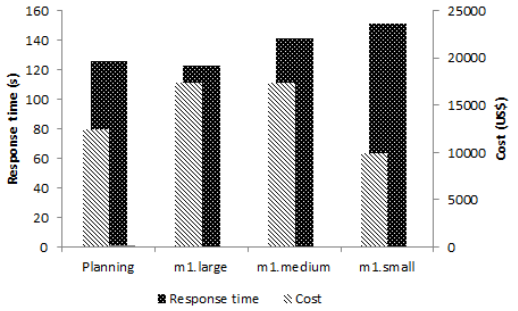
(a) Comparação da nuvem planejada - Perfil I



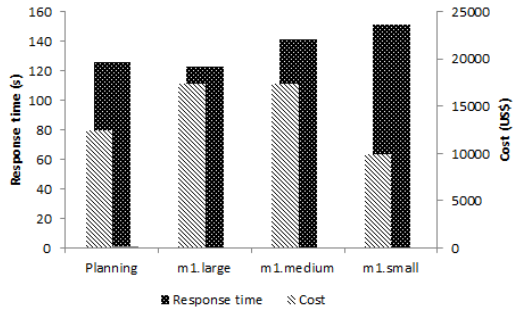
(b) Comparação da nuvem planejada - Perfil II



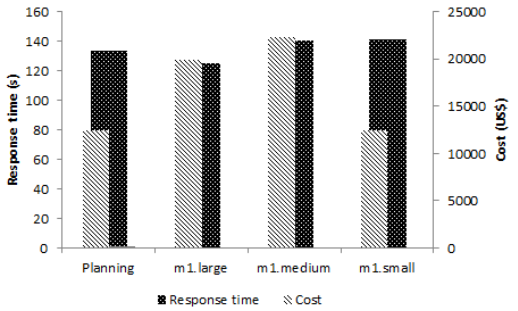
(c) Comparação da nuvem planejada - Perfil III



(d) Comparação da nuvem planejada - Perfil IV



(e) Comparação da nuvem planejada - Perfil V



(f) Comparação da nuvem planejada - Perfil VI

Figura 39 – Comparação entre custo e tempo de resposta - Nuvem Privada otimizada e não otimizada

Desta forma, é possível estimar o **cost** (o custo relacionado as métricas de disponibilidade, COA, desempenho e custo de aquisição de máquinas físicas com valores normalizados e calculado com a distância euclidiana (ver Seção 5.5)) de cada arquitetura. A Tabela 28 apresenta o resultado para cada arquitetura com base nos modelos apresentados nas seções anteriores. Da mesma forma, a Figura 40 apresenta o resultado considerando o método de otimização proposto.

Embora, o ambiente analisado proporcione uma pequena quantidade de combinações, é possível observar que o método proposto garante um bom resultado com base nas arquiteturas apresentadas. O **cost** para a arquitetura **A1** apresenta um melhor resultado considerando as métricas de disponibilidade, COA, desempenho e custo.

Com isso, foi possível observar que o método proposto auxilia administradores de

Tabela 28 – Resultado da melhor arquitetura com base na Disponibilida, COA, desempenho e custo

Arquitetura	Quantidade de VM (por tipo)			Cost
	Large	Medium	Small	
A1	2	0	0	1.297731
A2	1	2	0	1.333822
A3	1	1	2	1.355007
A4	1	0	4	1.358437
A5	0	4	0	1.371245
A6	0	3	2	1.399725
A7	0	2	4	1.399899
A8	0	1	6	1.398707
A9	0	0	8	1.425107

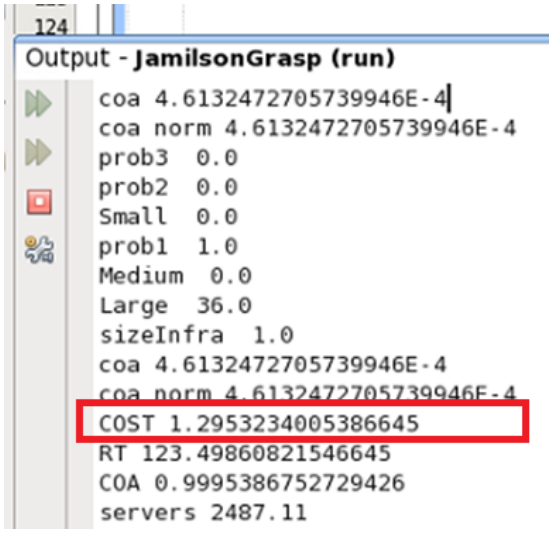


Figura 40 – Resultado: Algoritmo de otimização

sistemas com base nas arquiteturas de nuvens para serviços de VoD *streaming* a encontrarem melhores combinações para esse tipo de serviço. É importante ressaltar que, quando não temos o auxílio de um método como o apresentado nessa tese, o trabalho para analisar e obter resultados se torna cansativo, pois, é necessário avaliar, com base em cada métrica, normalizar os dados e assim calcular a distância euclidiana para obter um resultado conclusivo para cada arquitetura.

6.6 Considerações finais

Este capítulo apresentou cinco estudos de caso com o objetivo de planejar infraestruturas de nuvens conforme a metodologia e o método de otimização propostos. O primeiro estudo proporcionou a avaliação de arquiteturas de nuvem para ambiente de *VoD streaming*

considerando arquiteturas com *cluster* de nuvem integrado em uma única máquina física ou o *cluster* de nuvem separado (em uma máquina física independente). Para o estudo de caso 2, elaboramos uma equação matemática para estimar a disponibilidade e disponibilidade orientada à capacidade (COA) de infraestruturas de nuvens computacionais. Um modelo CTMC foi concebido para ser utilizado no processo de validação e comparação com o modelo matemático elaborado. O estudo de caso 3 proporciona a validação do modelo de desempenho para o sistema de *VoD streaming*. Com o modelo validado é possível explorá-lo para expandir seus resultados e apresentá-los no Estudo de caso 4. O Estudo de caso 5 proporciona a seleção de infraestruturas de nuvens considerando fatores como: disponibilidade, desempenho e custo.

7 CONCLUSÕES E TRABALHOS FUTUROS

A computação em nuvem vem sendo amplamente utilizada em pesquisas (BRILHANTE et al., 2014) (SOUSA et al., 2014) (GHOSH et al., 2014) (RIBAS et al., 2015), assim como por empresas que adotam esse sistema tipo de tecnologia. Esse fato é consequência da possibilidade de utilização da internet como fonte distribuidora dos dados nela contidos. Dessa forma, a computação em nuvem permite o compartilhamento dos recursos computacionais, tais como, poder de processamento, memória e disco através da Internet.

Porém, desafios são encontrados quando empresas querem fazer uso dessa tecnologia. O planejamento de uma infraestrutura, seja ela de nuvem ou não, exige cuidados e estudos para que não haja falhas, perda de receita e custos elevados na elaboração do projeto. Tratando-se de empresas que fornecem serviços de vídeo sob demanda, características como o processo de transcodificação e o tipo de vídeo a ser enviado para a infraestrutura podem levar a empresa ao fracasso.

Técnicas como modelagem ou simulação têm ajudado no processo de avaliação de sistemas. Essa tese propôs uma solução integrada para o planejamento de infraestruturas de nuvens privadas, considerando os aspectos de desempenho, de disponibilidade e disponibilidade orientada à capacidade e de custo de aquisição de computadores para serviços de vídeo sob demanda. Esse trabalho representa técnicas de modelagem hierárquica para avaliação das infraestruturas de nuvens privadas. Outras técnicas como, redes de Petri estocásticas, equações de custo e equações de disponibilidade orientada à capacidade são apresentados como modelos auxiliares no processo de avaliação e planejamento.

Esta tese alcançou uma série de resultados nas áreas que explorou e sua maior contribuição é o conjunto de modelos para planejamento de infraestruturas de nuvem para sistemas de *VoD*. Esta tese também fornece guias para melhorar os sistemas de forma contínua, servindo como suporte em um processo de planejamento da infraestrutura. As técnicas foram testadas por meio de estudos de casos distintos e produziram diversas outras contribuições.

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

7.1 Limitações

Mesmo a proposição de diversos modelos e os resultados alcançados, esta tese possui algumas limitações.

A principal limitação é que os modelos de desempenho do sistema de *VoD* tendem a crescer quando avaliados em cenários distintos. Desta forma, o modelo passa a não ser avaliado analiticamente e começar a ser resolvido somente por simulação, devido à explosão de estados, que faz com que possa ter resultados imprecisos. Outra limitação é quanto à geração de modelos de disponibilidade para a infraestrutura de nuvem. Os modelos poderiam ser gerados automaticamente, já que o usuário leigo, sem conhecimento ou com pouco conhecimento dos modelos apresentados, poderia fazer uso de um *framework* no qual faria um modelo de alto nível. Assim, seriam feitas interligações entre os componentes do sistema (*hardware*, *software* e componentes de interconexão), através da comunicação com uma API de algum software de modelagem que desse suporte ao formalismo adotado, como o Mercury (SILVA et al., 2015), por exemplo, no qual fosse feita a geração dos modelos e avaliação das métricas adotadas de forma automatizada.

Outra limitação é quanto aos modelos e técnicas utilizadas nessa tese. O usuário deve ter conhecimento básico para a parametrização das variáveis no sistema. Uma forma de facilitar o método de avaliação é criar um sistema que ajude o usuário final nesse processo, sistema esse que pode ser incluído como um trabalho futuro e não como limitação.

7.2 Contribuições

Como resultado das atividades desenvolvidas nesta tese, podemos identificar as seguintes contribuições:

- Elaboração de modelos de dependabilidade para infraestrutura de nuvem privada considerando sistema de *VoD* streaming. Esses modelos representam também o sistema computacional, a máquina virtual, os módulos de gerenciamento da plataforma de nuvem e o sistema de *VoD streaming* escolhido.

- A formulação matemática para o cálculo da disponibilidade orientada à capacidade. Esse modelo permite estimar o COA para grandes infraestruturas de nuvem privada. O processo de avaliação e validação permitiu identificar as vantagens de se ter um modelo matemático para este fim.

- Elaboração de um modelo de desempenho para infraestrutura privada de transcodificação de vídeo: O modelo gerado foi validado através de experimentos específicos. Com base na validação e avaliação do modelo de desempenho, permitiu-se a geração de cenários e combinação de diferentes cargas de trabalho na infraestrutura de transcodificação de vídeo na nuvem.

- 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 que respeite um SLA específico baseado no tempo de

resposta para transcodificação de vídeos e encontre um limiar de minimização dos custos e maximização da disponibilidade do sistema.

- Planejamento de infraestruturas de nuvens privadas: esta tese possibilita o planejamento de infraestruturas de nuvens privadas que atendam aos requisitos de custo estabelecidos.

7.3 Trabalhos futuros

Embora esta tese tenha alcançado diversos resultados e coberto alguns pontos relacionados ao planejamento de infraestruturas de nuvem considerando sistemas de *VoD streaming*, há muitas possibilidade de estender o trabalho atual. Algumas dessas possibilidades podem ser implementados em trabalhos futuros, os quais estão listadas abaixo:

A proposição de modelos específicos no processo de avaliação de desempenho de vídeos na infraestrutura de nuvem, considerando fatores como, *jitter*, frames por unidade de tempo e até mesmo avaliação do protocolo *MPEG-DASH* (MPEG-DASH, 2018) que permite a entrega dos vídeos em alta qualidade sobre o convencional HTTP. Além disso, também seria possível avaliar métricas de disponibilidade, confiabilidade, desempenho e consequentemente, performabilidade do sistema. Possivelmente, seria necessária a execução de experimentos em ambientes controlados para obtenção de alguns dos parâmetros de entrada, mas, também, é possível obter parte desses dados a partir da literatura, ou ainda a partir de estimativas, já que tais valores dependem de diversas questões de infraestrutura, portanto, não podem ser considerados somente como valores fixos e arbitrários.

Como mencionado anteriormente, é possível construir um sistema, como por exemplo, um sistema web que permita a entrada dos parâmetros dos modelos apresentados e que gere as combinações possíveis com base nos dados que foram também apresentados. Esse sistema, pode fazer uso de outros modelos e métricas, assim como, especificidades de métricas de mídias transmitidas via Internet. Outra característica interessante, é a elaboração de outros modelos que representem sistemas de *live streaming*. Tais sistemas podem fazer uso dos modelos apresentados ou realizar pequenas mudanças para o ambiente de transmissão de vídeo ao vivo.

Também é possível estender os resultados dessa tese com o desenvolvimento de algoritmos de otimização mais eficientes, além de um estudo mais aprofundado na busca de configurações melhores em algoritmos de otimização aplicados à modelagem estocástica e melhorar e/ou comparar algoritmos de busca mais eficientes, comparando os resultados e o tempo de avaliação no critério de busca por melhores soluções. Além disso, técnicas de análise de sensibilidade, como variação paramétrica e derivada parcial, também podem ser utilizadas para propor melhorias nos resultados. Dessa maneira, seria possível testar inúmeras novas possibilidade sob uma diferente perspectiva.

REFERÊNCIAS

AHMAD, I.; WEI, X.; SUN, Y.; ZHANG, Y.-Q. Video transcoding: an overview of various techniques and research issues. *IEEE Transactions on multimedia*, IEEE, v. 7, n. 5, p. 793–804, 2005.

AMAZON. *Amazon Elastic Compute Cloud - EC2*. 2017. Amazon. Available in: http://aws.amazon.com/pt/ec2/.

AMAZON. *Amazon Elastic Compute Cloud (Amazon EC2)*. 2017. Disponível em <http://calculator.s3.amazonaws.com/index.html>.

APACHE. *Apache CloudStack - Open Source Cloud Computing*. 2017. Cloudstack. Available in: https://cloudstack.apache.org/.

APACHE. *Apache JMeter*. 2017. Available on <http://jmeter.apache.org/>.

ARAÚJO, C. J. M. *Avaliação e modelagem de desempenho para planejamento de capacidade do sistema de transferência eletrônica de fundos utilizando tráfego em rajada*. [S.l.]: Recife, 2009.

ARAúJO, J. C. T. de. *Software Aging Monitoring Strategies and Rejuvenation Policies for Eucalyptus Cloud Computing Platform*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Mar. 2012.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I. et al. *Above the Clouds: A View of Cloud Computing*. [S.l.], 2010.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>.

AVIZIENIS, A.; LAPRIE, J.; RANDELL, B. et al. Fundamental concepts of dependability. *Technical Report Series-University Of Newcastle Upon Tyne Computing Science*, UNIVERSITY OF NEWCASTLE UPON TYNE, 2001.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. E. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, v. 1, n. 1, p. 11–33, 2004.

BALBO, G. Introduction to stochastic petri nets. *Lectures on Formal Methods and PerformanceAnalysis*, Springer, p. 84–155, 2001.

BALBO, G. *Introducation to stochastic Petri nets, Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*. [S.l.]: Springer-Verlag New York, Inc., New York, NY, 2002.

BEZERRA, M. C.; MELO, R.; DANTAS, J.; MACIEL, P.; VIEIRA, F. Availability modeling and analysis of a vod service for eucalyptus platform. In: IEEE. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 3779–3784.

BOLCH, G.; GREINER, S.; MEER, H. de; TRIVEDI, K. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. [S.l.]: Wiley-Interscience, 2006.

BRILHANTE, J.; SILVA, B.; MACIEL, P.; ZIMMERMANN, A. Dependability models for eucalyptus infrastructure clouds considering vm life-cycle. In: IEEE. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 1336–1341.

CALLOU, G.; MACIEL, P.; TUTSCH, D.; ; ARAUJO, J. Models for dependability and sustainability analysis of data center cooling architectures. In: *The 2nd International Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology (DCDV 2012) in conjunction with The 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. Boston, MA, USA: [s.n.], 2012.

CAROLAN, J.; GAEDE, S. Introduction to cloud computing architecture. *SUN Microsystems Inc., pp.–1-40*, 2009.

CASSANDRAS, C.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Kluwer academic publishers, 1999. v. 11.

CHAISIRI, S.; LEE, B.-S.; NIYATO, D.; MILONE, M. Optimization of resource provisioning cost in cloud computing. *Services Computing, IEEE Transactions on*, IEEE, v. 5, n. 2, p. 164–177, 2012.

CHUOB, S.; POKHAREL, M.; PARK, J. S. Modeling and analysis of cloud computing availability based on eucalyptus platform for e-government data center. In: IEEE. *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*. [S.l.], 2011. p. 289–296.

COLMENAR, J. M.; GREISTORFER, P.; MARTÍ, R.; DUARTE, A. Advanced greedy randomized adaptive search procedure for the obnoxious p-median problem. *European Journal of Operational Research*, Elsevier, v. 252, n. 2, p. 432–442, 2016.

CORPORATION, M. *Windows Azure*. 2017. Microsoft. Available in: https://azure.microsoft.com/pt-br/.

COSTA, I.; ARAUJO, J.; DANTAS, J.; CAMPOS, E.; SILVA, F. A.; MACIEL, P. Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform. *Quality and Reliability Engineering International*, Wiley Online Library, 2015.

CUOMO, A.; MODICA, G. D.; DISTEFANO, S.; PULIAFITO, A.; RAK, M.; TOMARCHIO, O.; VENTICINQUE, S.; VILLANO, U. An sla-based broker for cloud infrastructures. *Journal of grid computing*, Springer, v. 11, n. 1, p. 1–25, 2013.

DANTAS, J.; MATOS, R.; ARAUJO, J.; MACIEL, P. An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. [S.l.], 2012. p. 1664–1669.

DANTAS, J.; MATOS, R.; ARAUJO, J.; MACIEL, P. Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. *Computing*, Springer, v. 97, n. 11, p. 1121–1140, 2015.

DANTAS, J.; MATOS, R.; ARAUJO, J.; OLIVEIRA, D.; OLIVEIRA, A.; MACIEL, P. Hierarchical model and sensitivity analysis for a cloud-based vod streaming service. In: IEEE. *Dependable Systems and Networks Workshop, 2016 46th Annual IEEE/IFIP International Conference on*. [S.l.], 2016. p. 10–16.

DAOUST, F.; HOSCHKA, P.; PATRIKAKIS, C. Z.; CRUZ, R. S.; NUNES, M. S.; OSBORNE, D. S. Towards video on the web with html5. *NEM Summit*, p. 6, 2010.

DELGADO, G. D.; FRÍAS, V. C.; IGARTUA, M. A. Video-streaming transmission with qos over cross-layered ad hoc networks. In: IEEE. *Software in Telecommunications and Computer Networks, 2006. SoftCOM 2006. International Conference on*. [S.l.], 2006. p. 102–106.

DESROCHERS, A. A.; AL-JAAR, R. Y. *Applications of Petri nets in manufacturing systems: modeling, control, and performance analysis*. [S.l.]: IEEE, 1995.

DÍAZ-SÁNCHEZ, D.; ALMENAREZ, F.; MARÍN, A.; PROSERPIO, D.; CABARCOS, P. A. Media cloud: an open cloud computing middleware for content management. *Consumer Electronics, IEEE Transactions on*, IEEE, v. 57, n. 2, p. 970–978, 2011.

DILLON, T.; WU, C.; CHANG, E. Cloud computing: Issues and challenges. In: IEEE. *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. [S.l.], 2010. p. 27–33.

DOCUMENTATION, E. *HPE Helion Eucalyptus*. Goleta, CA, 2017.

EBELING, C. *An introduction to reliability and maintainability engineering*. [S.l.]: Tata McGraw-Hill Education, 2004.

EC2, A. *Virtual Server Hosting*. 2016. Amazon. Available in: https://aws.amazon.com/ec2/.

ENGINE, G. app. *Google app engine*. 2017. Google.com. Available in: https://cloud.google.com/appengine/docs/.

EUCALYPTUS. *Eucalyptus Cloud Computing Platform - Administrator Guide*. [S.l.], 2010. Version 1.6.

EUCALYPTUS. *The Open Source Cloud Platform*. 2012. Eucalyptus Systems. Available in: http://open.eucalyptus.com/.

FFMPEG. *FFmpeg*. 2017. FFmpeg. Available in: https://www.ffmpeg.org/.

FOX, A.; GRIFFITH, R.; JOSEPH, A.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, v. 28, p. 13, 2009.

FURHT, B.; ESCALANTE, A. *Handbook of cloud computing*. [S.l.]: Springer, 2010.

GARCIA, A.; KALVA, H.; FURHT, B. A study of transcoding on cloud environments for video content delivery. In: ACM. *Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing*. [S.l.], 2010. p. 13–18.

GERMAN, R.; KELLING, C.; ZIMMERMANN, A.; HOMMEL, G. Timenet: a toolkit for evaluating non-markovian stochastic petri nets. *Performance Evaluation*, Elsevier, v. 24, n. 1, p. 69–87, 1995.

GHOSH, R.; LONGO, F.; FRATTINI, F.; RUSSO, S.; TRIVEDI, K. S. Scalable analytics for iaas cloud availability. *Cloud Computing, IEEE Transactions on*, IEEE, v. 2, n. 1, p. 57–70, 2014.

GHOSH, R.; LONGO, F.; NAIK, V. K.; TRIVEDI, K. S. Modeling and performance analysis of large scale iaas clouds. *Future Generation Computer Systems*, Elsevier, v. 29, n. 5, p. 1216–1234, 2013.

GHOSH, R.; NAIK, V. K.; TRIVEDI, K. S. Power-performance trade-offs in iaas cloud: A scalable analytic approach. In: IEEE. *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*. [S.l.], 2011. p. 152–157.

GHOSH, R.; TRIVEDI, K. S.; NAIK, V. K.; KIM, D. S. End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach. In: IEEE. *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*. [S.l.], 2010. p. 125–132.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of metaheuristics*. [S.l.]: Springer Science & Business Media, 2006. v. 57.

GOGRID. *Gogrid cloud hosting*. 2017. GoGrid.com. Available in: https://www.rackspace.com/datapipe.

GONG, C.; LIU, J.; ZHANG, Q.; CHEN, H.; GONG, Z. The characteristics of cloud computing. In: IEEE. *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*. [S.l.], 2010. p. 275–279.

GOOGLE. *Google Compute Engine*. 2017. Google. Available in: https://cloud.google.com/compute/.

GOOGLE. *Google Drive*. 2017. Google.com. Available in: https://drive.google.com/.

GOOGLE. *Serviços de e-mail do google - Gmail.com*. 2017. Gmail.com. Available in: http://www.gmail.com.

GUIMARAES, A. P.; OLIVEIRA, H.; BARROS, R.; MACIEL, P. Availability analysis of redundant computer networks: a strategy based on reliability importance. In: *Proceedings of 3rd IEEE International Conference on Communication Software and Networks (ICCSN 2011)*. [S.l.: s.n.], 2011. p. 328–332.

HEIMANN, D.; MITTAL, N.; TRIVEDI, K. Availability and reliability modeling for computer systems. *Advances in Computers*, v. 31, p. 175–233, 1990.

HERZOG, U. Formal methods for performance evaluation. *Lectures on Formal Methods and PerformanceAnalysis*, Springer, p. 1–37, 2001.

HIREL, C.; TUFFIN, B.; TRIVEDI, K. S. Spnp: Stochastic petri nets. version 6.0. In: *Computer Performance Evaluation. Modelling Techniques and Tools*. [S.l.]: Springer, 2000. p. 354–357.

HU, T.; GUO, M.; GUO, S.; OZAKI, H.; ZHENG, L.; OTA, K.; DONG, M. Mttf of composite web services. In: *Parallel and Distributed Processing with Applications (ISPA), 2010 Int. Symp. on*. [S.l.: s.n.], 2010. p. 130 –137.

JAIN, R. *The art of computer systems performance analysis*. [S.l.]: John Wiley & Sons, 2008.

KARTSON, D.; BALBO, G.; DONATELLI, S.; FRANCESCHINIS, G.; CONTE, G. *Modelling with generalized stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 1994.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. Availability modeling and analysis of a virtualized system. In: *Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim Int. Symp. on*. [S.l.: s.n.], 2009. p. 365–371.

KOREN, I.; KRISHNA, C. M. *Fault-tolerant systems*. [S.l.]: Morgan Kaufmann, 2010.

KRISHNAPPA, D. K.; ZINK, M.; SITARAMAN, R. K. Optimizing the video transcoding workflow in content delivery networks. In: ACM. *Proceedings of the 6th ACM Multimedia Systems Conference*. [S.l.], 2015. p. 37–48.

KUO, W.; ZUO, M. *Optimal reliability modeling: principles and applications*. [S.l.]: Wiley, 2002.

LI, X.; LI, Y.; LIU, T.; QIU, J.; WANG, F. The method and tool of cost analysis for cloud computing. In: IEEE. *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*. [S.l.], 2009. p. 93–100.

LILJA, D. J. *Measuring computer performance: a practitioner's guide*. [S.l.]: Cambridge University Press, 2005.

LITTLE, J. D. A proof for the queuing formula: L= λ w. *Operations research*, INFORMS, v. 9, n. 3, p. 383–387, 1961.

LIU, T.; SONG, H. Dependability prediction of high availability oscar cluster server. In: *Proceedings of the 2003 Int. Conf. on Parallel and Distributed Processing Techniques and Applications*. [S.l.: s.n.], 2003.

LUDWIG, H.; KELLER, A.; DAN, A.; KING, R. P.; FRANCK, R. Web service level agreement (wsla) language specification. *IBM Corporation*, p. 815–824, 2003.

MACIEL, P.; TRIVEDI, K.; MATIAS, R.; KIM, D. Performance and dependability in service computing: Concepts, techniques and research directions, ser. *Premier Reference Source. Igi Global*, 2011.

MACIEL, P.; TRIVEDI, K. S.; MATIAS, R.; KIM, D. S. Dependability modeling. In: *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*. Hershey: IGI Global, 2011.

MALHOTRA, M. Power-hierarchy of dependability model types. *IEEE Trans. on Reliability*, v. 43, n. 2, p. 493–502, Sept. 1994.

MARSAN, M. A.; BALBO, G.; CONTE, G. Performance models of multiprocessor systems. *The MIT Press, Cambridge, MA*, 1986.

MATEUS, G. R.; RESENDE, M. G.; SILVA, R. M. Grasp with path-relinking for the generalized quadratic assignment problem. *Journal of heuristics*, Springer, v. 17, n. 5, p. 527–565, 2011.

MATHEMATICA, W. *The world's definitive system for modern technical computing*. 2017. Disponível em <https://www.wolfram.com/mathematica/>.

MATOS, R.; MACIEL, P. R.; SILVA, R. M.; MILONE, M. Sensitive grasp: combinatorial optimisation of composite web services guided by sensitivity analysis. *International Journal of Web and Grid Services*, Inderscience Publishers (IEL), v. 12, n. 1, p. 63–80, 2016.

MELL, P.; GRANCE, T. The nist definition of cloud computing (draft). *NIST special publication*, v. 800, p. 145, 2011.

MELO, R. M. D.; BEZERRA, M. C.; DANTAS, J.; MATOS, R.; FILHO, I. J. D. M.; MACIEL, P. Redundant vod streaming service in a private cloud: Availability modeling and sensitivity analysis. *Mathematical Problems in Engineering*, Hindawi Publishing Corporation, v. 2014, 2014.

MENASCE, D. A.; ALMEIDA, V. A.; DOWDY, L. W.; DOWDY, L. *Performance by design: computer capacity planning by example*. [S.l.]: Prentice Hall Professional, 2004.

MOLLOY, M. Performance analysis using stochastic petri nets. *Computers, IEEE Transactions on*, IEEE, v. 100, n. 9, p. 913–917, 1982.

MPEG-DASH. *MPEG-DASH*. 2018. Available on <https://www.dacast.com/support/>.

MUÑOZ, V. M.; RAMO, A. C.; ALBOR, V. F.; DIAZ, R. G.; ARÉVALO, G. M. Rafhyc: An architecture for constructing resilient services on federated hybrid clouds. *Journal of Grid Computing*, Springer, v. 11, n. 4, p. 753–770, 2013.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989.

NETFLIX. *Lessons Netflix Learned from the AWS Outage*. 2016. The Netflix Tech Blog. Available on <http://techblog.netflix.com/2011/04/lessons-netflix-learned-from-aws-outage.html>.

OLIVEIRA, D.; BRINKMANN, A.; MACIEL, P. Advanced stochastic petri net modeling with the mercury scripting language. In: *ValueTools, 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. [S.l.: s.n.], 2017.

OPENNEBULA. *The Simplest Cloud Management Experience*. 2017. Available on <https://opennebula.org/>.

OPENSTACK. *OpenStack - Cloud Solution*. 2017. Openstack. Available in: https://www.openstack.org/.

PACKARD, H. *HPE Helion Eucalyptus, Deploying a basic cloud with FastStart*. HPE Helion, 2016.

PARHAMI, B. From defects to failures: a view of dependable computing. *ACM SIGARCH Computer Architecture News*, ACM, v. 16, n. 4, p. 157–168, 1988.

PETERSON, J. Petri nets. *ACM Computing Surveys (CSUR)*, ACM, v. 9, n. 3, p. 223–252, 1977.

POWERED, T. *FFMPEG*. 2017. A complete, cross-platform solution to record, covert and stream audio anda video. Available in: https://www.ffmpeg.org/.

QIU, X.; LI, H.; WU, C.; LI, Z.; LAU, F. Dynamic scaling of vod services into hybrid clouds with cost minimization and qos guarantee. In: IEEE. *Packet Video Workshop (PV), 2012 19th International*. [S.l.], 2012. p. 137–142.

REIBMAN, A.; SMITH, R.; TRIVEDI, K. Markov and markov reward model transient analysis: An overview of numerical approaches. *European Journal of Operational Research*, Elsevier, v. 40, n. 2, p. 257–267, 1989.

REISIG, W. *Petri nets: an introduction, volume 4 of EATCS monographs on theoretical computer science*. [S.l.]: Springer-Verlag, 1985.

RESNICK, R. *A modern taxonomy of high availability*. [S.l.]: Interlog, Tech. Rep, 1996.

RIBAS, M.; FURTADO, C.; BARROSO, G.; LIMA, A. S.; SOUZA, N.; MOURA, A. Modeling the use of spot instances for cost reduction in cloud computing adoption using a petri net framework. In: IEEE. *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. [S.l.], 2015. p. 1428–1433.

SAHNER, R. A.; TRIVEDI, K. S. Reliability modeling using sharpe. *IEEE Transactions on Reliability*, IEEE, v. 36, n. 2, p. 186–193, 1987.

SALESFORCE. *The social enterprise platform*. 2016. SalesForce.com. Available in: http://www.salesforce.com/platform.

SERVICES, A. web. *Amazon Elastic Block Store - EBS*. 2016. Amazon. Available in: https://aws.amazon.com/ebs/.

SERVICES, A. web. *Amazon Simple Stora Service - S3*. 2016. Amazon. Available in: https://aws.amazon.com/s3/.

SILVA, B.; MATOS, R.; CALLOU, G.; FIGUEIREDO, J.; OLIVEIRA, D.; FERREIRA, J.; DANTAS, J.; LOBO, A.; ALVES, V.; MACIEL, P. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: *Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN*. [S.l.: s.n.], 2015.

SOUSA, E.; SILVA, E.; LINS, F.; TAVARES, E.; MACIEL, P. Dependability evaluation of cloud infrastructures. In: IEEE. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 1282–1287.

SOUSA, E. d. *Avaliação do impacto de uma política de manutenção na performabilidade de sistemas de transferência eletrônica de fundos*. Tese (Doutorado) — Dissertação, Universidade Federal de Pernambuco, Centro de Informática, 2009.

SOUSA, E. T. G. d. *Modelagem de desempenho, dependabilidade e custo para o planejamento de infraestruturas de nuvens privadas*. [S.l.]: UNIVERSIDADE FEDERAL DE PERNAMBUCO, 2015.

SOUSA, E. Teixeira Gomes de; LINS, F. A. A.; TAVARES, E. A. G.; MACIEL, P. R. M. Performance and cost modeling strategy for cloud infrastructure planning. In: IEEE. *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. [S.l.], 2014. p. 546–553.

STEWART, W. J. *Introduction to the numerical solutions of Markov chains*. [S.l.]: Princeton Univ. Press, 1994.

STEWART, W. J. *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. [S.l.]: Princeton University Press, 2009.

STOCKHAMMER, T. Dynamic adaptive streaming over http–: standards and design principles. In: ACM. *Proceedings of the second annual ACM conference on Multimedia systems*. [S.l.], 2011. p. 133–144.

SUN, D.; CHANG, G.; GUO, Q.; WANG, C.; WANG, X. A dependability model to enhance security of cloud environment using systemlevel virtualization techniques. In: *Proc. First Int. Conf. on Pervasive Computing, Signal Processing and Applications (PCSPA 2010)*. Harbin: [s.n.], 2010.

SUN, H.; VETRO, A.; XIN, J. An overview of scalable video streaming. *Wireless Communications and Mobile Computing*, Wiley Online Library, v. 7, n. 2, p. 159–172, 2007.

TRIVEDI, K.; HUNTER, S.; GARG, S.; FRICKS, R. Reliability analysis techniques explored through a communication network example. In: *International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability*. [S.l.: s.n.], 1996. p. 2–3.

VIRT-MANAGER. *VM Virtual Machine Manager*. 2016. Virt-manager. Available in: https://virt-manager.org/.

WU, Y.; WU, C.; LI, B.; QIU, X.; LAU, F. Cloudmedia: When cloud on demand meets video on demand. In: IEEE. *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. [S.l.], 2011. p. 268–277.

XIE, M.; DAI, Y.-S.; POH, K.-L. *Computing system reliability: models and analysis*. [S.l.]: Springer Science & Business Media, 2004.

XIONG, K.; PERROS, H. Service performance and analysis in cloud computing. In: IEEE. *Services-I, 2009 World Conference on*. [S.l.], 2009. p. 693–700.

YANG, M.; CAI, J.; ZHANG, W.; WEN, Y.; FOH, C. H. Adaptive configuration of cloud video transcoding. In: IEEE. *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. [S.l.], 2015. p. 1658–1661.

YOUTUBE. *Share your videos with friends, family, and the world.* 2017. Available on <https://www.youtube.com/>.

YOUTUBE. *YouTube estatísticas.* 2017. Disponível em <https://www.youtube.com/intl/pt-BR/yt/about/press/>.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, Springer, v. 1, n. 1, p. 7–18, 2010.

ZHU, W.; LUO, C.; WANG, J.; LI, S. Multimedia cloud computing. *Signal Processing Magazine, IEEE*, IEEE, v. 28, n. 3, p. 59–69, 2011.

APÊNDICE A – SCRIPT PARA AVALIAÇÃO DO COA CONSIDERANDO UM SERVIDOR E QUATRO VMS

Este apêndice apresenta em detalhes o *script* utilizado para avaliação do modelo de disponibilidade e verificação do tempo de execução do modelo CTMC 26. Vale ressaltar que o modelo CTMC foi convertido para um modelo SPN em linguagem de script traduzido pela ferramenta Mercury (SILVA et al., 2015) (OLIVEIRA; BRINKMANN; MACIEL, 2017).

Listing A.1 – COA Evaluation Script

```
%\lstset{style=mystyle}
%\lstset{caption={COA Evaluation Script},label=redundantgc}
%\begin{lstlisting}

MTTFPM = 8760;
MTTFvm = 2880;
NVM = 4;
mttr = 1;
p = 1;

SPN Model{

    place vmsup( tokens = NVM * p );

    for i in range(1, p){

        place PM1_D_#($i);
        place PM1_U_#($i)( tokens= 1 );
        place VMsS1_D1_#($i);
        place VMsS1_D2_#($i);
        place VMsS1_U_#($i)( tokens= NVM );


        immediateTransition TI0_#($i)(
            guardExpression = #VMsS1_D2_#($i)>0,
            inputs = [VMsS1_D2_#($i)(#VMsS1_D2_#($i))],
            outputs = [VMsS1_U_#($i)(#VMsS1_D2_#($i)), vmsup(#VMsS1_D2_#($i))],
            inhibitors = [PM1_D_#($i)]
        );

        immediateTransition TI1_#($i)(
            guardExpression = #VMsS1_U_#($i)>0,
            inputs = [VMsS1_U_#($i)(#VMsS1_U_#($i)), VMsS1_D1_#($i)(#VMsS1_D1_#($i))
                , vmsup(#VMsS1_U_#($i))],
            outputs = [VMsS1_D2_#($i)((#VMsS1_U_#($i)+#VMsS1_D1_#($i)))],
            inhibitors = [PM1_U_#($i)]
        );
```

```
37
        timedTransition TE0_#($i)(
39          inputs = [VMsS1_D1_#($i)],
            outputs = [VMsS1_U_#($i), vmsup],
41          delay = mttr
        );
43
        timedTransition TE1_#($i)(
45          inputs = [VMsS1_U_#($i), vmsup],
            outputs = [VMsS1_D1_#($i)],
47          delay = MTTFvm,
            serverType = "InfiniteServer"
49      );

51      timedTransition TE4_#($i)(
            inputs = [PM1_D_#($i)],
53          outputs = [PM1_U_#($i)],
            delay = mttr
55      );

57      timedTransition TE5_#($i)(
            inputs = [PM1_U_#($i)],
59          outputs = [PM1_D_#($i)],
            delay = MTTFPM
61      );
    }
63
    reward r(
65      true -> (#VMsS1_U_1 + #VMsS1_U_2) / (NVM * 2)
    );
67
    reward r2(
69      true -> (#vmsup) / (NVM * p)
    );
71
    metric COA = stationaryReward( rewardRate = r2 );
73
}
75
main {
77  coa = solve( Model, COA );
    println(coa);
79 }
```