Pós-Graduação em Ciência da Computação

Rosangela Maria de Melo

ANÁLISE DE SENSIBILIDADE APLICADA À IDENTIFICAÇÃO DE PONTOS QUE REQUEREM MELHORIA NA DISPONIBILIDADE EM INFRAESTRURA DE *CLOUD*

Tese de Doutorado

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

RECIFE
2017

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

Rosangela Maria de Melo

ANÁLISE DE SENSIBILIDADE APLICADA À IDENTIFICAÇÃO DE PONTOS QUE REQUEREM MELHORIA NA DISPONIBILIDADE EM INFRAESTRURA DE *CLOUD*

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

Orientador: *Paulo Romero Martins Maciel*

RECIFE
2017

Rosangela Maria de Melo
Análise de Sensibilidade aplicada à identificação de pontos que requerem melhoria na disponibilidade em infraestrura de *Cloud*/ Rosangela Maria de Melo. – RECIFE, 2017-
146 p. : il. (algumas color.) ; 30 cm.

Orientador Paulo Romero Martins Maciel

Tese de Doutorado – Universidade Federal de Pernambuco, 2017.

1. Vídeo sob Demanda (VoD). 2. Modelos analíticos. I. Disponibilidade. II. Universidade Federal de Pernambuco. III. Modelos para Análise de Dependabilidade de Arquiteturas de um Serviço de VoD *streaming* na Nuvem

CDU 02:141:005.7

Tese de Doutorado apresentada por **Rosangela Maria de Melo** ao programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **Análise de Sensibilidade aplicada à identificação de pontos que requerem melhoria na disponibilidade em infraestrura de *Cloud***, orientada pelo **Prof. Paulo Romero Martins Maciel** e aprovada pela banca examinadora formada pelos professores:

Prof. Ricardo Massa Ferreira Lima
Centro de Informática/UFPE

Prof. Djamel Fawzi Hadj SadoK
Centro de Informática/UFPE

Profa. Érica Teixeira Gomes de Sousa
Departamento de Informática/UFRPE

Prof. Antônio Alfredo Ferreira Loureiro
Departamento de Ciência da Computação/UFMG

Prof. Giovanni Cordeiro Barroso
Departamento de Física/UFC

RECIFE
2017

Dedico esta tese à minha família, especialmente ao meu irmão, Ivanildo, pelo apoio incondicional e à minha filha, Sarah, pela compreensão da minha ausência na sua vida durante esses anos.

Agradecimentos

Agradeço a Deus por sempre estar presente na minha vida em todos os momentos, sejam eles felizes ou não. Hoje, estar finalizando essa etapa da minha vida representa muito para mim, significa a realização de um sonho almejado durante anos. E isso só foi possível porque Deus me trouxe de novo para a vida e acreditou em mim. Durante esse caminho aconteceram vários desencontros, mas pessoas amigas estavam presentes para ajudar na caminhada, que não foi fácil.

Agradeço ao meu orientador, Prof. Paulo Maciel, pela orientação, pela paciência, pelo suporte e conselhos durante todos esses anos. Sou muito agradecida por tudo. Agradeço também ao Prof. Eduardo Tavares, pela oportunidade de me auxiliar no ingresso dessa caminhada.

Agradeço aos professores Ricardo Massa, Djamel Sadok, Érica Sousa, Giovanni Barroso e Antônio Loureiro por aceitarem ao convite de participação nesta banca de doutorado.

Agradeço aos amigos do grupo MoDCS, especialmente à Jamilson, Clara e Rubens que fizeram parte dessa caminhada e me auxiliaram durante a jornada até aqui.

Agradeço a Vicente Marques, meu amor, pela compreensão, amizade, parceria e paciência comigo durante a finalização deste trabalho. Eu também agradeço por dividir os melhores momentos dessa caminhada.

Agradeço à minha família por todo suporte, amor e paciência durante esses quatro anos. Gostaria de agradecer, em especial, ao meu irmão Ivanildo, que sempre acreditou em mim, à minha mãe Edite que sempre me apoiou em todos os momentos, à minha irmã Leninha, pelo seu exemplo diário de luta pela vida, aos meus queridos sobrinhos João Victor e Beatriz pelo carinho e paciência e à minha filha, Sarah, que sempre entendeu a minha ausência, especialmente nessa fase final.

Agradeço *in Memorian* a Pedro Paulo Marques, um amigo muito especial, pelo apoio, pela atenção, pela preocupação, pelo carinho, e por ter acreditado no meu sonho desde o ínicio.

Agradeço ao Diretor do IFPE Campus Belo Jardim, Francisco Chagas, ao Diretor de Ensino, João Almeida e ao Coordenador de Informática, Fábio Feliciano por toda ajuda oportunizada ao desenvolvimento dessa pesquisa.

Agradeço, principalmente, a Deus, por ter colocado todas essas pessoas no meu caminho, e por elas terem contribuído direta ou indiretamente para que esse trabalho fosse finalizado.

"Tudo tem o seu tempo determinado, e há tempo para todo propósito debaixo do céu."

—ECLESIASTES 3:1.

Resumo

Durante vários anos, os sistemas de computação em nuvem vem gerando um debate e interesse dentro das corporações de TI. Estes ambientes de computação em nuvem fornecem sistemas de armazenamento e processamento que são adaptáveis, eficientes e simples, permitindo assim modificações na infraestrutura de forma rápida, de acordo com a variação da carga de trabalho. Organizações de qualquer tamanho e tipo estão migrando para nuvem suportando soluções baseadas na *Web*. Devido às vantagens do modelo de *pay-per-use* e fatores de escalabilidade, serviços como o de *Streaming* de Vídeo e o MBaaS *OpenMobester*, dependem fortemente dessas infraestruturas de nuvem para oferecer uma grande variedade de conteúdos de multimídia e armazenamento de dados dos dispositivos móveis. Recentes eventos de falha em serviços de *Streaming* de Vídeo, demonstraram a importância fundamental da manutenção da alta disponibilidade em infraestruturas de computação em nuvem. Um dos métodos utilizados para identificar as tendências de ocorrências de falhas em sistemas computacionais, ocorre por meio da aplicação de estratégias de análise de sensibilidade. Cada estratégia de análise de sensibilidade pode obter um *ranking* diferenciado, desse modo sugerimos a utilização para avaliação dos sistemas computacionais, de mais de uma estratégia, com o objetivo de obtermos alta confiabilidade desses sistemas. Esta tese propõe uma metodologia aplicada no domínio dos sistemas computacionais, em particular na computação em nuvem, combinando a proposição e adaptação de estratégias de análise de sensibilidade com métodos já existentes, realizando uma comparação entre elas, com o propósito de estabelecer um índice de sensibilidade a partir da atribuição de pesos, para as posições que os parâmetros ocupam em cada estratégia. Pretende-se obter um *ranking* coerente e com a minimização das discrepâncias entre as estratégias, visando identificar os principais pontos que requerem melhoria na disponibilidade desses ambientes. A metodologia baseia-se na utilização de estratégias de análise de sensibilidade, conjuntamente com a modelagem hierárquica, e com os modelos para representação de mecanismos de redundância visando atuar na *performance* do sistema. A metodologia foi testada ao longo de estudos de casos distintos, no serviço de *Streaming* de Vídeo e no serviço MBaaS *OpenMobester*, desde o nível de infraestrutura básica até a infraestrustrura com redundância. Os estudos de casos mostram que a abordagem proposta é útil para guiar os provedores de serviço de nuvem no processo de tomada de decisões, especialmente para ajustes eventuais e melhorias arquiteturais no serviço.

Palavras-chave: Computação em nuvem; disponibilidade; redundância; modelos analíticos; análise de sensibilidade; estratégias.

Abstract

For several years, cloud computing systems have been generating debate and interest within IT corporations. These cloud computing environments provide storage and processing systems that are adaptable, efficient and simple, thus allowing for rapid infrastructure modifications to be made, according to constantly changing workloads. Organizations of any size and type are migrating to the cloud supporting solutions based on Web. Due to the benefits of the pay-per-use model and scalability factors, services such as Video Streaming and MBaaS OpenMobester rely heavily on these cloud infrastructures to deliver a wide variety of multimedia content and Data storage of mobile devices. Recent failure events in video streaming services have demonstrated the critical importance of maintaining high availability in cloud computing infrastructures. One of the methods used to identify the trends of occurrences of failures in computational systems, occurs through the application of strategies of sensitivity analysis. Each strategy of sensitivity analysis can obtain a differentiated ranking, thus it is suggested that we use to evaluate the computational systems, more than one strategy, with the objective of obtaining high reliability of these systems. This thesis proposes a methodology applied in the field of computational systems, in particular in cloud computing, combining the proposition and adaptation of strategies of sensitivity analysis with existing methods, making a comparison between them, with the purpose of establishing a Index of sensitivity from the attribution of weights, the positions that the parameters occupy in each strategies. The aim is to achieve a coherent ranking and to minimize the discrepancies between the strategies, aiming to identify the main points that require improvement in the availability of these environments. The methodology is based on the use of sensitivity analysis strategies, together with the hierarchical modeling, and with the models to represent redundancy mechanisms aiming to perform in the performance of the system. The methodology has been tested over different case studies in the video streaming service and the MBaaS OpenMobester service, from the basic infrastructure to the redundant infrastructures. The case studies show that the proposed approach is useful for guiding cloud service providers in the decision-making process, especially for eventual adjustments and architectural improvements in the service.

Keywords: Cloud computing; availability; redundancy; analytical models; sensitivity analysis; strategies.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1

Introdução

Neste capítulo são apresentados as principais motivações para realização deste trabalho. Na sequência, são apresentados o contexto, a motivação, a justificativa e os objetivos. Para finalizar, a estrutura geral da pesquisa é apresentada.

1.1 Contexto

Os ambientes de computação em nuvem fornecem poder de processamento e capacidade de armazenamento ajustável, bem como, outros recursos computacionais, que permitem o fornecimento rápido de cargas de trabalho variáveis (ARMBRUST et al., 2010). Essa tecnologia ainda provê a infraestrutura, a plataforma e o *software* como serviços sob demanda, com custo associado ao uso (BUYYA; BROBERG; GOSCINSKI, 2010). Por esse motivo, as empresas e os desenvolvedores não precisam realizar grandes investimentos em *hardware* e manutenção para implementar os seus serviços.

Diversos são os serviços que podem ser hospedados em ambientes de nuvem, dentre eles está: o serviço de *Streaming* de Vídeo e o serviço de MBaaS, *Mobile Backend-as-a-Service* (ROWINSKI, 2012). O serviço de *Streaming*, nos últimos anos, vem apresentando um crescimento expressivo, para se ter uma percepção, em 2010, o consumo de vídeo era responsável por 40% de todo o tráfego da *internet*, passando para 50% em 2012 (HWANG et al., 2013) com a expectativa de atingir 80% a 90% até o final do ano de 2019 (INDEX, 2015). A *Netflix*[1] (NETFLIX, 2014) e o Youtube[2] (YOUTUBE, 2014) são os dois maiores provedores de entretenimento de serviços de *streaming* de vídeo que utilizam recursos da *internet* para prover aos usuários os seus serviços.

Sobre o serviço de MBaaS, conhecido como plataforma MBaaS tem sido registrado um crescimento, em virtude da necessiddade de permitir aos desenvolvedores vincularem o

[1] A *Net flix* é uma empresa de *streaming* de vídeo que oferece o serviço através da *internet* e acumula mais de 53 milhões de usuários pagantes distribuídos em cerca de 50 países, que consomem mais de dois bilhões de horas de conteúdo mensalmente (NETFLIX, 2014).

[2] O YouTube é um site que permite que seus usuários carreguem e compartilhem vídeos em formato digital, este site recebe por mês o acesso de mais de um bilhão de usuários únicos, registrando a visualização de mais de seis bilhões de vídeos (YOUTUBE, 2014).

backend de suas aplicações ao armazenamento em nuvem. Fornece recursos de gerenciamento de usuários e dispositivos que estarão aptos a utilizar a aplicação, notificações por *push* e integração com serviços em redes sociais (ROWINSKI, 2012). O negócio MBaaS teve um crescimento de US$ 216,5 milhões de dólares em 2012, estima-se US$ 7,7 bilhões de dólares para 2017 (MARKETSANDMARKETS.COM, 2012).

As infraestruturas de nuvem possuem muitos componentes interconectados, o que torna o gerenciamento e administração dos seus recursos uma atividade complexa, principalmente quando realizada de forma manual pelos seus administradores (TAURION, 2009). É importante evidenciar que essas não estão livres de falhas. Recentemente ocorreram falhas na *Amazon AWS* (NETFLIX, 2014) onde foi interrompido o serviço de prestação de várias empresas importantes, incluindo uma das principais empresas no mercado de VoD, a *Netflix* (NETFLIX, 2014).

Em razão desses eventos, abordar as questões de disponibilidade, qualidade de serviço e desempenho tem sido indispensável, em virtude desses ambientes serem grandes, complexos, dinâmicos e heterogêneos. A modelagem analítica associada à análise de sensibilidade tem sido fortemente aplicada no planejamento e gerenciamento desses recursos de nuvem e provê subsídios que garantem ao sistema funcionamento com níveis de qualidade desejados com relação à disponibilidade e desempenho (MATOS; MACIEL; SILVA, 2015).

Os sistemas podem ser modelados por meio dos formalismos, tais como: diagramas de bloco de confiabilidade (*Reliability Block Diagrams* (RBDs), (BIRNBAUM, 1969)(DANTAS et al., 2012);(BEZERRA et al., 2014); Cadeias de *Markov* de Tempo Contínuo (*Continous Time Markov Chains* (CTMCs) (MACIEL et al., 2011);(KIM; MACHIDA; TRIVEDI, 2009) e Redes de Petri (MALHOTRA; TRIVEDI, 1994).

Os autores enfatizam que a modelagem analítica auxilia no processo de gerenciamento e planejamento de *hardware*, infraestruturas de rede e *software* (MATOS et al., 2012). Seja comparando configurações alternativas antes de implementar um sistema, ou por proporcionar a previsão dos efeitos na disponibilidade e desempenho do sistema após as alterações em seus componentes (MATOS et al., 2012). Esse planejamento é essencial para sistemas com características críticas como a computação em nuvem. Além disso, os prestadores de serviços precisam cumprir acordos de níveis de serviço (SLA – *Service Level Agreement*) estabelecidos, caso contrário pode provocar multas, cancelamento de contratos e outras sanções financeiras ((SATO; TRIVEDI, 2007); (ARAUJO et al., 2011).).

Os autores HAMBY (1994); KUO; ZUO (2003a); SALTELLI (2002);(WILLIAMS et al., 2012) apresentam a análise de sensibilidade como uma estratégia que avalia a variação dos parâmetros de entrada do sistema e identifica o quanto estas variações podem interferir na saída do mesmo. Por meio dessa estratégia é possivel identificar os pontos de criticidade e promover melhoria para o sistema. Complementando, os autores OU; DUGAN (2003), afirmam que análise de sensibilidade é uma fase importante no planejamento e no processo de tomada de decisões, pois ela pode ser utilizada para os seguintes propósitos: i) tomar melhores decisões; ii) decidir quais dados estimados devem ser refinados antes de tomar uma decisão; e iii) concentrar-se nos

elementos críticos durante a implementação.

A análise de sensibilidade pode ser aplicada na fase inicial de implementação, em arquiteturas já operacionais ou na fase de planejamento de arquiteturas computacionais (KUO; ZUO, 2003a). Aplicada nessas fases, permitem detectar potenciais problemas e proporcionam aos decisores a opção de direcionar ações assertivas para manter o sistema em funcionamento (KUO; ZUO, 2003a); (DARADKEH; CHURCHER; MCKINNON, 2008).

São exemplos de estratégias de análise de sensibilidade, a variação um por um, *design of experiments*, índices de importância, diferença percentual e sensibilidade diferencial paramétrica (MATOS et al., 2012);(BEZERRA et al., 2014), essas são utilizadas de forma isolada e, além disso, apresentam algumas limitações como, por exemplo, o cálculo do *ranking* de importância em um determinado parâmetro, sem a possibilidade de variação. Essas estratégias manuseam muitos parâmetros diferentes, e cada parâmetro pode ter um impacto distinto na métrica de interesse, é fundamental conhecer a posição de importância dos parâmetros do modelo, para que possamos decidir o nível apropriado de atenção que se deve diligenciar para cada um (HORST H. HENN , AUTH.).

Pesquisas são realizadas na computação em nuvem, em virtude da complexidade desses sistemas, no entanto, as soluções existentes não têm se preocupado em encontrar os pontos de *upgrade* com o propósito de melhorar a disponibilidade. A aplicação de estratégias de análise de sensibilidade em conjunto com a modelagem hierárquica, podem auxiliar na identificação desses pontos e no planejamento, visando atender os critérios de qualidade que esses ambientes exigem. Elas podem guiar os processos de tomada de decisão, fornecendo informações sobre os pontos de otimização no sistema analisado.

Desse modo, essa pesquisa busca concentrar esforços na proposição e adaptação de estratégias de análise de sensibilidade conjuntamente com métodos existentes para identificar os pontos que requerem melhoria para a disponibilidade dos sistemas computacionais, em particular na computação em nuvem. A combinação dessas estratégias, aliadas ao uso da modelagem hierárquica e ao estudo de modelos para representação dos mecanismos de redundância desses ambientes, compõe uma metodologia para análise de sensibilidade e identificação de pontos críticos que requerem melhoria na disponibilidade do sistema.

A metodologia desenvolvida baseia-se na proposição de estratégias para identificar os pontos críticos de uma infraestrutura de nuvem, tais como o Índice Discreto Médio de Sensibilidade, o qual denominamos de DASI e o Índice de Importância Crítica para a Disponibilidade, denominado de ICD. O método proposto constitui como um roteiro para identificar os pontos que precisam de melhorias, enfatizando na manutenção da métrica disponibilidade dessas infraestruturas.

1.2 Motivação e Justificativa

Para BUYYA; BROBERG; GOSCINSKI (2010), a computação em nuvem é um tipo de sistema paralelo e distribuído, que consiste em uma coleção de computadores interconectados e virtualizados, dinamicamente provisionados e apresentados como um ou mais recursos computacionais e tais recursos devem ser baseados em acordos de níveis de serviço estabelecidos através de negociações entre o provedor de serviços e os clientes.

Ao longo dos últimos anos, o uso de voz e dados na *internet* aumentou significativamente. Esse crescimento está relacionado com a interoperabilidade oferecida por voz, imagem e serviços de dados, de custos baixos. Esses devem ser continuamente fornecidos, mesmo quando eventos como falhas de *hardware* ou *software* acontecerem, afetando a disponibilidade da rede (FURTADO; REGO; LOURAL, 2005).

´Nos últimos anos, os serviços de *Streaming* de Vídeo e o MBaaS, são exemplos de serviços que apresentam um crescimento ascendente. Em abril de 2014, a *Netflix* (ADHIKARI et al., 2014) atraiu mais de 35 milhões de assinantes nos EUA e cerca de 48 milhões em todo o mundo. É a maior fonte de tráfego de *internet*, consumiu 29,7% do tráfego *downstream* [3] em 2011. Por outro lado, o mercado de MBaaS (MARKETSANDMARKETS.COM, 2012) teve um crescimento de US$ 216,5 milhões, o equivalente a R$ 779,2 milhões em 2012 para US$ 7,7 bilhões em 2017, o equivalente a R$ 30,8 bilhões, o que representa uma taxa de crescimento anual composta de 104% entre 2012 e 2017.

Alinhado a esses cenários de crescimento, o perfil do usuário também mudou. Atualmente temos um tipo de usuário que requer conexão 24 horas através de diversos tipos de dispositivos móveis ou de computadores pessoais. O interesse de tráfego desses usuários deixou de ser apenas voz, para ser dados, como imagens, músicas, vídeo sob demanda. Os acessos aos serviços de video sob demanda, como *Netflix, YouTube* e *iTunes*, são maiores do que os da TV aberta.

Nesse contexto, garantir os níveis requeridos de disponibilidade para esses serviços hospedados na computação em nuvem não tem sido uma tarefa trivial (PEIXOTO, 2012). A ocorrência de eventuais falhas nesses serviços pode ocasionar a degradação dos tempos de respostas e a interrupção do atendimento da requisição devido à indisponibilidade do serviço ou do recurso solicitado (BAUER; ADAMS, 2012). A paralisação desses serviços pode ser ocasionada pela ocorrência de eventos de falha no *hardware* ou no *software*, nos sistemas de energias, do sistema de refrigeração e de rede de dados (BAUER; ADAMS, 2012).

A análise de sensibilidade vem sendo utilizada, para avaliar o comportamento de tendências à ocorrência de eventos de falhas (IMAN; JOHNSON; WATSON, 2005). Por meio dos resultados da análise de sensibilidade, podemos de forma significativa, direcionar os recursos para as variáveis de grande influência, quando se objetiva melhorias para a disponibilidade do sistema (BRUNI; FAMÁ; SIQUEIRA, 1998).

Desse modo, a proposição de estratégias de análise de sensibilidade combinadas com

[3] *Downstream* é o fluxo de dados no sentido servidor > usuário.

métodos já existentes podem auxiliar no estabelecimento dos pontos que requerem melhoria na disponibilidade desses ambientes. Aliadas às estratégias de análise de sensibilidade, as técnicas de modelagem hierárquicas podem ser utilizadas para representar aspectos de dependabilidade dos sistemas configurados na nuvem (JAIN, 1991). A utilização de modelos para representar o funcionamento de sistemas permite resultados confiáveis a um custo muito baixo, visto que não é necessário construir um sistema real para analisá-los.

1.3 Objetivos

O objetivo desse trabalho é a proposição e adaptação de estratégias de análise de sensibilidade e adoção de uma metodologia baseada em modelagem hierárquica e estudo de mecanismos de redundância, para a avaliação de infraestruturas em nuvem, visando garantir os seus critérios de qualidade. A metodologia adotada possibilita a detecção de pontos de melhoria da disponibilidade.

Para alcançar o objetivo geral, temos os objetivos específicos:

- Proposição e adaptação de estratégias de análise de sensibilidade (AS) para identificar pontos de implementação de mecanismos de redundância.
- Proposição de procedimento para aplicar estratégias de análise de sensibilidade em modelos hierárquicos.

1.4 Estrutura do documento

Essa tese está estruturada da seguinte maneira: No segundo capítulo, são abordados a fundamentação teórica do trabalho proposto, apresentando uma visão geral sobre a computação de nuvem, dependabilidade, modelos de confiabilidade e disponibilidade, e análise de sensibilidade. No terceiro capítulo são apresentados os trabalhos relacionados a tese proposta. Esse capítulo está dividido em trabalhos relacionados a dependabilidade e análise de sensibilidade e na comparação dos trabalhos relacionados. No quarto capítulo são apresentados a proposição de estratégias de análise de sensibilidade, *Discrete Averaged Sensitivity Index* (Índice Discreto Médio de Sensibilidade) (DASI) e Índice de Importância Crítica para a Disponibilidade (ICD). No quinto capítulo são apresentados e descritos a metodologia usada para aplicar as estratégias de análise de sensibilidade nos ambientes de computação em nuvem. No sexto capítulo são apresentados os modelos para representação de mecanismos de redundância *cold e warm standby* e ativo-ativo. No sétimo capítulo são apresentados três estudos de casos que foram utilizados para verificar a aplicabilidade da abordagem proposta, bem como para demonstrar seus benefícios e limitações. No oitavo capítulo finaliza o documento, são apresentados um resumo dos resultados alcançados com essa pesquisa de tese, além das contribuições e os trabalhos futuros.

2

Fundamentos

Neste capítulo são apresentados conceitos que norteiam o desenvolvimento desta pesquisa. Sua estrutura é da seguinte maneira: primeiramente, é introduzido o conceito de computação em nuvem, onde é abordada a classificação dos serviços oferecidos, bem como, os seus principais tipos. Na sequência, as definições de dependabilidade e dos modelos de confiabilidade e disponibilidade, evidenciando os principais conceitos sobre Redes de Petri Estocásticas (SPN), Diagramas de Blocos de Confiabilidade (*Reliability Block Diagrams* (RBD)) e Cadeia de *Markov* (TRIVEDI et al., 1996), (MACIEL; LINS; CUNHA, 1996), (KUO; ZHU, 2012), (BIRNBAUM, 1969),(MACIEL et al., 2011). Em seguida, enfatizamos as principais técnicas de análise de sensibilidade (AS). O capítulo finaliza com a descrição das considerações finais.

2.1 Computação na Nuvem

O sistema de Computação em Nuvem (CN) (em inglês *cloud computing*) é composto pela união de recursos virtuais prontamente utilizáveis e acessíveis, tais como *hardware*, *software*, plataformas de desenvolvimento e serviços (TAURION, 2009), (BUYYA; BROBERG; GOSCINSKI, 2010), (ARMBRUST et al., 2009). Esses recursos podem ser alocados de forma dinâmica e redimensionados para ajustar-se a uma carga de trabalho variável, possibilitando a otimização do uso dos recursos (TAURION, 2009). Esse conjunto de recursos é tipicamente explorado, por meio de um modelo que o cliente é cobrado pela utlização do serviço ou recurso de acordo com uso, e de acordo com a autorização do provedor mediante acordos de nível de serviços (CARDOSO, 2010), (BUYYA; BROBERG; GOSCINSKI, 2010), (ARMBRUST et al., 2009).

De acordo com Giordanelli e Mastroianni (2010), o uso da CN proporciona uma série de informações em relação às quais os usuários não precisam se preocupar com a sua localização, além disso, precisam saber apenas que os seus dados podem ser acessados de qualquer lugar do mundo e que devem possuir conexão com a *internet* GIORDANELLI; MASTROIANNI (2010). A CN é importante para a distribuição e acesso de recursos compartilhados, oferecendo vantagens tais como: escalabilidade, segurança, disponibilidade estacionária e confiabilidade

(TAURION, 2009), (ARMBRUST et al., 2009), (BUYYA; BROBERG; GOSCINSKI, 2010).

O NIST (*The National Institute of Standards and Technology*) define a computação em nuvem como um modelo que permite, de forma conveniente, o acesso ubíquo sob demanda a rede, a um *pool* compartilhado de recursos computacionais configuráveis, por exemplo: redes, servidores, armazenamento, aplicações e serviços. Esses podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011). Inclusive, a CN também define três modelos de serviço para a arquitetura são eles: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS) (GONG et al., 2010), (EUCALYPTUS, 2014a). Esses modelos estão detalhados em (COUTINHO et al., 2013), (LINDEMANN, 2015) e (LINDEMANN, 2015).

Uma das maneiras de caracterizar uma nuvem computacional é a partir do ponto de vista do usuário da nuvem e de quem provê a infraestrutura (ZHANG; CHENG; BOUTABA, 2010). Desse modo, é possível classificar segundo o *NIST*, uma nuvem por meio de três tipos nuvem pública, nuvem privada e nuvem híbrida, as quas estão descritas em (ZHANG; CHENG; BOUTABA, 2010), (STANDART; TECHNOLOGY'S, 2013).

A computação em nuvem tem proporcionado para o usuário final a possibilidade de não se preocupar com a aquisição de novas infraestruturas, bem como, a manutenção da mesma. Isso ocorre, em virtude do que, o usuário pode utilizar a infraestrutura de nuvem para armazenamento dos seus dados com segurança. disponibilidade, escalabilidade. Pagando apenas pelo recurso que utilizar.

Os diferentes tipos de infraestruturas de nuvens públicas e privadas têm oferecido serviços para o usuário com o mesmo nível de qualidade, diferenciando-se entre si no que diz respeito à segurança da informação e ao investimento necessário para estabelecer esse ambiente. As nuvens privadas possuem um controle maior sobre a segurança da informação em relação às nuvens públicas.

Desse modo, para a condução do trabalho utilizamos uma infraestrutura de nuvem privada *Eucalyptus*, contendo o serviço de *Streaming* de Vídeo hospedado na nuvem e uma outra infraestrutura com o serviço *Mobile Backend-as-a-Service* (MBaaS) (ROWINSKI, 2012), (FERNANDO; LOKE; RAHAYU, 2013), também hospedado no ambiente de nuvem *Eucalyptus*.

A escolha de utilizar uma arquitetura em nuvem privada deu-se em função das características desse tipo de nuvem que propicia o controle, consolidação e gerenciamento dos recursos da nuvem (NURMI et al., 2009), (COUTINHO et al., 2013). No entanto, a escolha da plataforma de nuvem privada *Eucalyptus* ocorreu por causa dos seguintes fatos:

- Possui um ambiente de computação em nuvem bem planejado (CHAGANTI, 2009);
- É um sistema de núcleo aberto com uma arquitetura de *software* baseado em *Linux*, que facilita a implementação Infraestrutura como Serviço (IaaS) de nuvens privadas e híbridas (NURMI et al., 2009),(EUCALYPTUS, 2014b);

- Foi desenvolvida para hospedar comunicação e pesquisa para plataformas de computação em nuvem (CHAGANTI, 2009);
- Está disponível gratuitamente na forma de código fonte, facilitando o entendimento do código e a elaboração de extensões para a plataforma (CHAGANTI, 2009);
- Possui compatibilidade com a *Application Programming Interface* (API) da *Amazon* EC2 (*Elastic Compute Cloud*) (NURMI et al., 2009), (EUCALYPTUS, 2014b). Essa funcionalidade permite que o usuário possa criar instâncias de máquinas virtuais (Virtual Machine (VM)) usando imagens de diferentes sistemas operacionais, como várias distribuições Linux; *Microsoft Windows Server* 2003, 2008 e 2012; *OpenSolaris* e *FreeBSD*, além disso cada máquina física passa a ser um nó *Eucalyptus* que pode executar VM sobre diferentes tipos de hipervisores, entre eles, *Xen*, VMware e KVM (NURMI et al., 2009), (EUCALYPTUS, 2014b).
- Possui flexibilidade, suportando a migração das aplicações de uma nuvem para a outra prontamente (CHAGANTI, 2009),(FURHT; ESCALANTE, 2010);
- Oportuniza a criação de nuvens híbridas que usam nuvens privadas e públicas concomitantemente (NURMI et al., 2009),(EUCALYPTUS, 2014b)

Na Figura 2.1 apresentamos uma arquitetura de nuvem na plataforma *Eucalyptus* com o serviço de *Streaming* de Vídeo hospedado (MELO et al., 2015). Esse serviço, pela sua natureza e abrangência, irá usufruir das características do ambiente, tais como a escalabilidade, mobilidade e disponibilidade (COUTINHO et al., 2013).

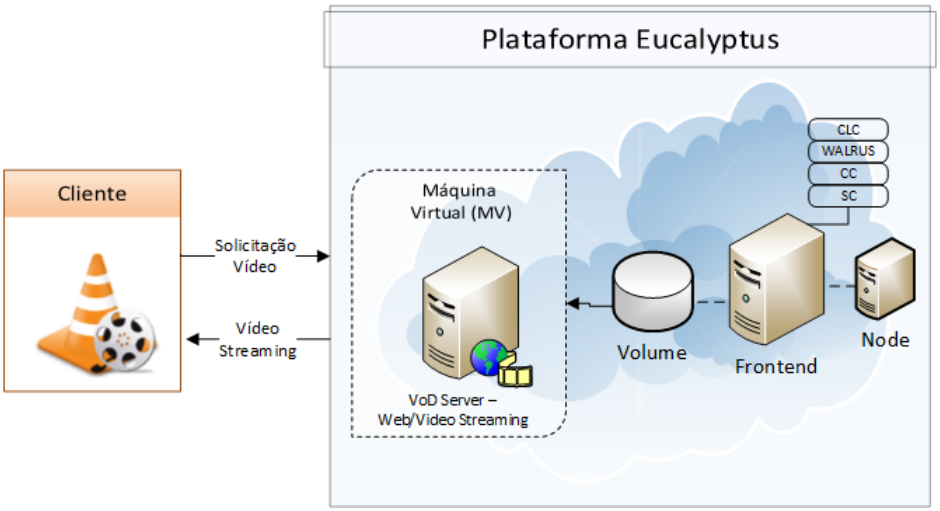


Figura 2.1: Arquitetura de nuvem *Eucalyptus*

As VMs instanciadas possuem as aplicações VLC, FFMPEG e Apache instaladas para realização do processo de *upload* dos vídeos. Esses *players* permitem que os vídeos possam ser codificados para diversos formatos como MPEG, H.264, WVM e MP4 entre outros.

Além da arquitetura de nuvem *Eucalyptus* com o serviço de *Streaming* de Vídeo, a Plataforma do *MBaaS OpenMobster* (ROWINSKI, 2012), (FERNANDO; LOKE; RAHAYU, 2013), presente na Figura 2.2 foi igualmente analisada. Possui a mesma infraestrutura de nuvem *Eucalyptus* apresentada anteriormente, com componente *Frontend*, Node, VM com MBaaS, que contém a plataforma MBaaS e o NAS, o sistema de armazenamento. É uma plataforma *open source* que provê integração entre aplicações móveis e serviço de *cloud*. Ela faz *interface* com os clientes móveis, fornecendo a estes, serviços de *backend* que podem abranger diversas categorias, desde aplicações de *e-mail* e serviços *Web* até mobilização de banco de dados.

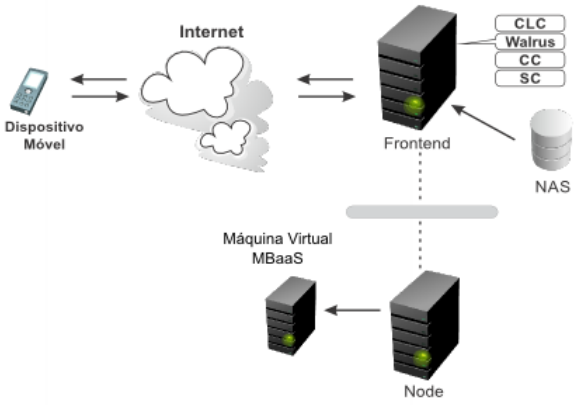


Figura 2.2: Arquitetura da Plataforma *OpenMobster*

Na próxima seção, abordaremos o conceito de dependabilidade, apresentando os seus atributos, suas principais caracteríscas e as equações que podem ser utilizadas para obter essa métrica.

2.2 Dependabilidade

Dependabilidade está intimamente relacionada com áreas de tolerância a falhas e de confiabilidade (WEBER, 2003). Weber em (WEBER, 2003) e (MALHOTRA; TRIVEDI, 1994), Laprie em (LAPRIE; AVIZIENIS; KOPETZ, 1992) afirmam que a dependabilidade é um conceito genérico que inclui disponibilidade, confiabilidade, segurança, integridade e manutenção.

Os três principais eixos da dependabilidade são: atributos, meios e ameças (WEBER, 2003). O eixo **atributo** é composto por (WEBER, 2003): **Disponibilidade** - compreende que o sistema está operacional para ser usado; **Confidencialidade** - corresponde à ausência de divulgação desautorizada de informação; **Confiabilidade** - corresponde à continuidade do fornecimento do serviço até o momento em que ocorre a falha; **Segurança** - corresponde à ausência de consequências graves; **Manutenabilidade** - corresponde a predisposição à manutenção; **Integridade** - corresponde à ausência de alterações no sistema. Por outro lado, os **meios** são formados por remoção de falhas, tolerância a falhas, prevenção e previsão de falhas, em contrapartida as **ameaças** que são agrupadas por erros, falhas e defeitos (AVIZIENIS et al., 2004), (AVIZIENIS

et al., 2001).

Embora os conceitos de disponibilidade e confiabilidade estejam intimamente ligados, há uma distinção sutil entre eles. Considerando que a confiabilidade do sistema em t é a probabilidade de que o sistema executa suas funções sem falha até o instante de tempo t que ocorre uma falha (KUO; ZUO, 2003a), (XIE; DAI; POH, 2004). Por outro lado, a disponibilidade (A) é a probabilidade de que o sistema esteja funcionando e pronto para uso em um dado instante de tempo t, expressa pela Equação 2.1 (KUO; ZUO, 2003a), (XIE; DAI; POH, 2004), (MACIEL et al., 2012):

$$A = \frac{E[Uptime]}{(E[Uptime] + E[Downtime])}, \tag{2.1}$$

Em que:

- A é a disponibilidade estacionária do sistema;
- E [*Uptime*] é o valor esperado de funcionamento do sistema quando está operacional;
- E [*Downtime*] é o valor esperado do sistema quando o mesmo está não operacional;

De maneira análoga, a indisponibilidade (UA) é obtida a partir das Equações 2.2 e 2.3.

$$UA = \frac{E[Downtime]}{(E[Uptime] + E[Downtime])} \tag{2.2}$$

Assim:

$$UA = 1 - A. \tag{2.3}$$

A disponibilidade pode ser expressa em função do MTTF e do MTTR do sistema, conforme apresentado na Equação 2.4 (KUO; ZUO, 2003a), (XIE; DAI; POH, 2004).

$$A = \frac{MTTF}{(MTTF + MTTR)} \tag{2.4}$$

Em que,

- *Mean Time To Failure* (MTTF) - : representa o tempo médio até apresentar falhas no sistema;
- *Mean Time To Repair* (MTTR) - : representa o tempo médio para reparo do sistema.

A partir da métrica indisponibilidade do sistema é possível obter duas outras métricas, sendo elas o ***downtime* (D) anual** e o **número de noves**. O número de 9 (noves) corresponde à quantidade de algarismos consecutivos iguais a 9 após a virgula e é expressso por meio da Equação 2.5.

$$N_9 = |log_{10}(1-A)| \tag{2.5}$$

Esse número representa o quanto o sistema está próximo de 100% da disponibilidade desejada (OLIVEIRA; MACIEL, 2014). A medida em que o número de noves aumenta, a indisponibilidade e *downtime* anual diminui e consequentemente, para manter esses resultados é necessário implementar mecanismos de redundância, tais como: *warm* e *cold standby* ou ativo-ativo (OLIVEIRA; MACIEL, 2014). O *downtime* anual corresponde ao tempo em que o sistema passa indisponível durante o ano. A Equação 2.6 apresenta o cálculo desse elemento:

$$D = UA \times 8760(horas) \tag{2.6}$$

A natureza de um sistema em nuvem de funcionar com o mínimo de interrupção possível, alinha-se à temática de dependabilidade que possui vários atributos que visam manter um ambiente computacional operacional com disponibilidade, confiabilidade e segurança. Desse modo, os modelos analíticos são usados como ferramentas de planejamento que auxiliam na representação e simulação de falhas nesses ambientes.

Identificar os atributos da dependabilidade, como, por exemplo, a disponibilidade, é uma tarefa que requer aplicação de técnicas coerentes a fim de minimizar o tempo de paralisação e o tempo de resposta desse sistema. A métrica disponibilidade, nessa pesquisa, foi calculada a partir dos tempos médios para falhar e para reparar MTTF e MTTR respectivamente, presentes em cada componente de *hardware* e *software* (LAPRIE; AVIZIENIS; KOPETZ, 1992), (OLIVEIRA; MACIEL, 2014),(MELO et al., 2015) das arquiteturas representadas pelas Figuras 2.1 e 2.2, (vide páginas 24 e 25).

Na próxima seção, abordaremos os modelos de confiabilidade e disponibilidade, apresentando as suas principais características.

2.3 Modelos de Confiabilidade e Disponibilidade

A modelagem analítica[1] utiliza um conjunto de equações e funções matemáticas para reproduzir o comportamento de um sistema (MELO, 2014), (TAY, 2010). Durante a construção dos modelos, devemos levar em consideração a complexidade e praticidade dos sistemas (TAY, 2010).

Os modelos analíticos possibilitam uma análise em relação aos efeitos causados pelos parâmetros definidos nas equações sobre a aplicação (JAIN, 1991). Além disso, é possível determinar os prováveis relacionamentos entre cada um dos parâmetros considerados (MELO, 2014). A validação dos modelos analíticos pode ser verificada mediante a comparação dos

[1] consite em um modelo que é uma abstração do sistema que consegue capturar, dentre os inúmeros detalhes do sistema, aqueles que são essenciais para o seu comportamento. Podemos dizer ainda que refere-se à representação matemática do sistema (TAY, 2010), (RAEDER et al., 2011).

valores reais do sistema com os valores medidos em testes experimentais (JAIN, 1991).

Existem vários tipos de modelos que podem ser utilizados para a avaliação analítica de dependabilidade tais como (OLIVEIRA; MACIEL, 2014): Diagramas de bloco de confiabilidade - RBD (BIRNBAUM, 1969), árvores de falhas, redes de Petri Estocásticas e Cadeia de *Markov* de Tempo Contínuo (CTMC), os mesmos tem sido usados para representar sistemas tolerantes a falhas e avaliar medidas de dependabilidades diferentes (MALHOTRA; TRIVEDI, 1994). Esses tipos de modelos diferem de um para outro, não só na facilidade de utilização para uma aplicação em particular, mas em termos de poder de modelagem (MALHOTRA; TRIVEDI, 1994).

A cadeia de *Markov* de Tempo Contínuo encontra-se detalhada em MENASCE et al. (2004), KIM; MACHIDA; TRIVEDI (2009), SOUZA (2009), JULIAN MENEZES ARAUJO; ROMERO MARTINS MACIEL (2009), MACIEL et al. (2011) DANTAS et al. (2012), OLIVEIRA; MACIEL (2014). Assim como as técnicas RBD e rede de Petri encontram-se detalhadas em (BIRNBAUM, 1969) (TRIVEDI et al., 1996), FIGUEIRÊDO; MACIEL (2011),(ČEPIN, 2011), HAJIAN-HOSEINABADI; GOLSHAN (2012), KUO; ZHU (2012) e MARSAN et al. (1994) MACIEL; LINS; CUNHA (1996), BALBO (2001), MÁRIO LINS GALDINO; ROMERO MARTINS MACIEL (2009), respectivamente.

Os ambientes de nuvens possuem um grande número de equipamentos, e gerenciar esses recursos não é uma tarefa fácil. Por isso, a modelagem analítica auxilia no processo de gerenciamento. A modelagem é capaz de retratar o funcionamento de ambientes computacionais reais. Essa característica da modelagem é relevante, ao passo que é possível atuar em ambientes de testes e verificar como eles podem se comportar a partir de um evento real.

A fidelidade de representação dos modelos analíticos pode ser verificada por meio de um processo de validação (COSTA, 2015). A validação dos modelos tem o propósito de mostrar que um modelo elaborado é compatível com o ambiente pretendido (COSTA, 2015). Nessa pesquisa, utilizamos o método do intervalo de confiança da disponibilidade proposto por (KEESEE, 1965a), para realizar a validação dos modelos.

Os tipos de modelos como redes de Petri, Cadeias de *Markov* e RBD, são usados para representação de sistemas como o ambiente de computação em nuvem. Convém evidenciar que a aplicação do tipo de modelagem a ser utilizada depende do nível de detalhes que desejamos representar.

Os modelos de alto nível RBD, retratam uma representação global do sistema sem apresentar o relacionamento existente entre os componentes. Estes tipos representam o funcionamento do sistema de forma independente e em alguns casos, esses modelos trazem outro modelos associados, ou seja, outros tipos de modelos que servem como dados de entrada para este modelo. Isso é o que denominamos de modelagem hierárquica utilizada nessa pesquisa (MELO, 2014).

Inicialmente, os modelos RBDs foram utilizados para representar as arquiteturas analisadas em alto nível, possibilitando uma visão geral da infraestrutura. Além disso, foram utilizados os componentes que são representados por módulos independentes, onde cada um pode ser

facilmente representado por um bloco de disponibilidade (MELO, 2014).

No entanto, identificamos a necessidade de modelar esses sistemas em conjunto com outros modelos, de forma hierárquica. Isso ocorreu, em razões de particularidades e detalhes desses sistemas, não sendo possível representá-los exclusivamente pelos modelos RBD (MELO et al., 2015), (COSTA, 2015). Desse modo, as Cadeias de *Markov* atendem perfeitamente essas exigências e foram usadas nesses sistemas. No entanto, ao utilizá-las nos deparamos com situações em que a quantidade excessiva de número de estados pode, inviabilizar o seu uso (SILVA et al., 2006).

O modelo de redundância ativo-ativo foi inicialmente confeccionado por meio de uma cadeia de *Markov*, em que nos deparamos com a explosão de espaço de estados, em consequência disso, adotamos um outro tipo de modelo para representar essa arquitetura, sendo a rede de Petri a modelagem escolhida, em vista de possuir uma representação gráfica, com métodos poderosos de análise formal (SILVA et al., 2006). Registra-se ainda que o formalismo de rede de Petri é utilizado principalmente em sistemas que possam apresentar atividades paralelas, assíncronas, concorrentes e não-determinísticas (MELO, 2014).

Na próxima seção, abordaremos o conceito sobre a análise de sensibilidade, apresentando suas principais caracteríscas e aplicação, bem como os tipos de estratégias de sensibilidade utilizadas nessa pesquisa.

2.4 Análise de Sensibilidade

Nesta seção apresentamos alguns conceitos sobre análise de sensibilidade (AS) sob o ponto de vista de diferentes autores. Abordamos as estratégias de AS: i) variação um por um, ii) sensibilidade diferencial paramétrica, iii) diferença percentual, iv) *design of experiments* e v) importância para a confiabilidade.

Para SALTELLI et al. (2004), o termo análise de sensibilidade é interpretado em diferentes comunidades técnicas e aplicado em diferentes configurações de problemas. Assim, até pouco tempo, a AS foi concebida e muitas vezes definida como uma medida do efeito local de um determinado dado de entrada em relação ao dado de saída (SALTELLI et al., 2004). O autor complementa que a AS tem por objetivo mapear os elos fracos dos sistemas computacionais e, a partir desse ponto, busca aplicar um conjunto de estratégias que visam melhorias para esses sistemas em diferentes contextos.

De acordo com JUNIOR et al. (2011), é comum em sistemas computacionais termos vários componentes, que não possuem necessariamente a mesma importância para o seu desempenho. Conhecer bem essa infraestrutura e identificar quais são os elementos relevantes para o seu desempenho, a fim de que possamos direcionar ações específicas para cada componente.

Para MARVIN RAUSAND (2003), um estudo prático de confiabilidade de um sistema complexo, como por exemplo um sistema de computação em nuvem, é uma tarefa que consome tempo para encontrar estimativas adequadas para os parâmetros de entrada (taxas de falha,

taxas de reparo, etc.) e verificar o impacto desses parâmetros na disponibilidade do sistema (MARVIN RAUSAND, 2003). Em alguns casos, podemos começar com estimativas grosseiras, nesse sentido as estratégias de AS podem auxiliar esse processo de refinamento e de identificação dos componentes mais importantes do sistema, bem como na identificação dos componentes que possuem um efeito insignificante associado à disponibilidade do sistema (MARVIN RAUSAND, 2003).

Sem a AS muitas vezes os analistas colocam muita ênfase sobre os resultados obtidos nas suas análises, apresentando-as como verdade. No entanto, não possui evidências de como alcançou os resultados (JAIN, 1991). Na ausência da AS adequada, não se têm certeza dos resultados obtidos, desse modo surgem alguns questionamentos, tais como: será que as conclusões estão corretas? Será que as conclusões poderiam ser diferentes, ou ainda, o que aconteceria se a análise tivesse sido realizada em um cenário ligeiramente diferente? Além disso, não considerando a AS, dificulta a identificação de quais são os parâmetros relevantes, principalmente se o sistema possui muitos parâmetros (JAIN, 1991). Desse modo, a AS traz a segurança necessária e pode direcionar os resultados dentro da perspectiva pré-estabelecida pelos administradores do sistema.

Vários métodos de análise de sensibilidade estão disponíveis na literatura tais como: análise de correlação, análise de regressão e análise de perturbação (PA), análise diferencial paramétrica, variação um por um , simulação de Monte Carlo, correlação de *Pearson*, correlação de *Spearman*, análise de variância (ANOVA), *fourier amplitude sensitivity test- (FAST)*, *design of experiments (DoE)*, diferença percentual, importância para confiabilidade e disponibilidade e o método de Sobol.

A escolha de qual o método adotar para realizar a análise de sensibilidade, é um passo difícil. Essa escolha precisa ser consoante às questões a serem abordadas, aos recursos computacionais disponíveis, e às características dos problemas abordados (CAMPOLONGO; TARANTOLA; SALTELLI, 1999);(PIANOSI et al., 2016).

2.4.1 Variação um por um

Existem diversas formas de se conduzir uma análise de sensibilidade (SOUZA MATOS JÚNIOR, 2011). Algumas delas podem ser devidamente utilizadas nos modelos analíticos, como cadeias de *Markov*, visto que outras abordagens são mais adequadas para medições experimentais baseadas em análises (SOUZA MATOS JÚNIOR, 2011). O método mais simples, numa visão conceitual, é o de variar um parâmetro por vez, enquanto se mantém os outros parâmetros sem variação (HAMBY, 1994). Quando aplicado este método, o *ranking* de sensibilidade é obtido a partir da observação das correspondentes mudanças na saída do modelo.

Relacionamentos inesperados entre variáveis de entrada e saída podem também ser revelados a partir dessa abordagem, desencadeando a necessidade de outras investigações, para isso pode-se realizar uma nova análise utilizando diferentes estratégias e comparando o resultado entre elas (HAMBY, 1994).

Para essa estratégia, uma das maneiras de variar os parâmetros de entrada do modelo é utilizar uma determinada percentagem do valor médio do parâmetro de entrada, que pode ser utilizada para incrementar a entrada e observar o comportamento do sistema (DOWNING; GARDNER; HOFFMAN, 1985). Além disso, o parâmetro pode ser também incrementado em $\pm 20\%$ a partir do seu desvio padrão, quando essa informação é conhecida (DOWNING; GARDNER; HOFFMAN, 1985),(HAMBY, 1995), ou ainda podemos variar o valor de cada parâmetro de entrada em $\pm 10\%$ ou $\pm 20\%$ do valor nominal do parâmetro (MATOS et al., 2015), (HAMBY, 1995).

Contudo, variar um parâmetro por vez, às vezes pode não ser a forma mais prática de realizar uma análise. Quando lidamos com um grande número de parâmetros, as análises das dispersões dos pontos tornam-se mais difíceis, principalmente devido às proximidades das curvas (MAIER; NORTON; CROKE, 2005). A diferença em termos de magnitude é outro possível fator crítico (MAIER; NORTON; CROKE, 2005), (HENDERSON-SELLERS; HENDERSON-SELLERS, 1996);(CAMPOLONGO et al., 2000); (SALTELLI et al., 2000)) uma vez que, todos os parâmetros não podem ser visualizados no mesmo gráfico, inibindo interpretações precisas das diferenças entre as influências dos parâmetros.

2.4.2 Análise de Sensibilidade Diferencial Paramétrica

Análise de sensibilidade diferencial paramétrica (SDP) também conhecida como análise diferencial ou ainda como método direto é a base de muitas outras técnicas de AS (HAMBY, 1994). Geralmente, ela pode ser realizada de maneira computacionalmente eficiente em modelos analíticos, podendo ser utilizada nas análises de desempenho e dependabilidade. Esse método fornece um coeficiente de sensibilidade único que indica a quantidade de mudança na saída que foi produzida por uma mudança adicional de um dado de entrada (HAMBY, 1994).

Análise de SDP baseia-se na existência de uma equação algébrica que descreve a relação entre a medida de interesse e os parâmetros de entrada. Esse método é obtido por meio do cálculo de derivadas parciais da métrica de interesse com relação a cada parâmetro de entrada. Por exemplo, encontrar a sensibilidade do parâmetro Y, que depende de um parâmetro λ, é conseguido na Equação 2.7 ou Equação 2.8 (FRANK, 1978), (HAMBY, 1994).

$$S_\lambda(Y) = \frac{\partial Y}{\partial \lambda} \tag{2.7}$$

$$S^*_\lambda(Y) = \frac{\partial Y}{\partial \lambda}\left(\frac{\lambda}{Y}\right) \tag{2.8}$$

A Equação 2.8 normalmente é utilizada quando é necessário normalizar os resultados. $S_\lambda(Y)$ e $S^*_\lambda(Y)$ são os coeficientes de sensibilidade de Y em relação a λ (HAMBY, 1994), cujos valores são ordenados a produzir um *ranking* que é utilizado para comparar o grau de influência entre todos os parâmetros.

As funções de sensibilidade com base em derivadas parciais são utilizadas em áreas como avaliação de desempenho de sistemas computacionais, processamento de sinal, economia, química e física (HAMBY, 1994),(SALTELLI et al., 2004). Nesses campos, a melhoria da precisão do modelo é muitas vezes o principal objetivo, por meio de medidas adicionais de parâmetros com maiores coeficientes de sensibilidade.

2.4.3 Diferença Percentual

Um outro método para a determinação do parâmetro de AS é para calcular a Diferença Percentual de saída quando varia um parâmetro de entrada a partir do seu valor mínimo para o seu valor máximo (HOFFMAN; GARDNER, 1983),(HAMBY, 1994). Hoffman e Gardner (1993), defendem a utilização de toda a gama de cada parâmetro de valores possíveis, a fim de avaliar a sensibilidade de parâmetros (HOFFMAN; GARDNER, 1983).

O índice de sensibilidade é calculado usando a Equação 2.9. Esta equação mostra a expressão para essa abordagem, em que $max\{Y(\theta)\}$ e $min\{Y(\theta)\}$ são os valores máximo e mínimo de saída, respectivamente, calculados ao variar o parâmetro θ ao longo de uma gama de n possíveis valores de interesse. Se $Y(\theta)$ é conhecido para variar monotonicamente, de modo que apenas os valores extremos de θ (i.e.,θ_1 e θn) podem ser usados para calcular $max\{Y(\theta)\}$ e $min\{Y(\theta)\}$ e consequentemente $S_\theta\{Y\}$ (MATOS et al., 2015).

$$S_\theta\{Y\} = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}} \tag{2.9}$$

Em que,

$$max\{Y(\theta)\} = max\{Y(\theta_1), Y(\theta_2), \cdots, Y(\theta_n)\} \tag{2.10}$$

e

$$min\{Y(\theta)\} = min\{Y(\theta_1), Y(\theta_2), \cdots, Y(\theta_n)\} \tag{2.11}$$

2.4.4 *Design of Experiments* - DoE

Design of Experiments (DoE) (BOX; WILSON, 1951) é um método usado para executar AS (MATOS et al., 2015) (MELO et al., 2015). Mediante este método, é possível avaliar a importância de cada um dos parâmetros do sistema e, além disso, ele pode ser usado para determinar simultaneamente os efeitos individuais e interativos de fatores que podem afetar as medidas de saída (MATHEWS, 2005), (MATOS et al., 2015). Nesse método, os parâmetros são chamados fatores e o valor que é atribuído a cada um dos fatores é chamado de nível (MATHEWS, 2005), (MATOS et al., 2015). Exemplos de análises realizadas para avaliar a importância dos parâmetros de um sistemas podem ser encontrados nas pesquisas realizadas pelos autores em (FRANCESCHINI; MACCHIETTO, 2008), (TIWARI et al., 2016).

DoE pode obter o máximo de informação com um pequeno número de experimentos (MATHEWS, 2005). Isso reduz o trabalho que teria sido gasto na coleta de dados (MATHEWS, 2005). Uma análise adequada de experimentos ajuda a separar os efeitos de vários fatores que podem afetar o desempenho (JAIN, 1991).

Além disso, o DoE permite determinar se um fator tem um efeito significativo ou se a diferença observada é simplesmente devido a variações aleatórias causados por erros de medição e parâmetros que não foram controlados (JAIN, 1991). Existem alguns tipos de DoE e entre os mais comumente usados estão: planejamento fatorial completo ou *design* fatorial completo, *design* fatorial fracionário e *design* simples (JAIN, 1991).

Em um *design* fatorial completo, todas as combinações possíveis de configuração e carga de trabalho são examinadas. Num determinado sistema, que tem o seu desempenho afetado por k fatores (parâmetros), e cada fator tem N níveis possíveis de valores, o número de experimentos seria N^K (MATHEWS, 2005). Este método permite encontrar o efeito de cada fator, incluindo as interações entre eles. O número de níveis de cada um dos fatores pode ser reduzido como uma estratégia para lidar com uma grande quantidade de fatores. JAIN (1991) considera o *design* fatorial 2^k como uma abordagem popular, em que apenas dois níveis são avaliados para cada fator (MATHEWS, 2005).

Às vezes, o número de experimentos necessários para um *design* fatorial completo é extenso, isso pode acontecer se tanto o número de fatores quanto o de seus níveis são numerosos (MATHEWS, 2005). Desse modo pode não ser possível a utilização de um *design* fatorial completo, devido ao custo ou ao tempo necessário para execução do experimento (MATHEWS, 2005). Por isso, existem três maneiras de reduzir o número de experimentos:

- reduzir o número de níveis para cada fator;
- reduzir o número de fatores;
- usar *design* fatorial fracionário (JAIN, 1991),(MATHEWS, 2005);

Fatoriais fracionários são comumente usados para reduzir o número de execuções necessárias para construir um experimento (MATHEWS, 2005). É recomendado para os casos em que o experimento possui 2^k modelos com cinco ou mais variáveis (MATHEWS, 2005).

2.4.5 Importância para a Confiabilidade - IC

Ao medir a importância relativa dos componentes, podemos determinar quais componentes merecem uma pesquisa adicional para o desenvolvimento de melhorias para a confiabilidade do sistema (BIRNBAUM, 1969). A medida de importância, conhecida como a medida de importância de Birnbaum do componente i no tempo t, foi introduzida por Birnbaum (HUSEBY, 2004):

$$I^B_{(i/t)} = \frac{\partial R(t)}{\partial p_i(t)} \tag{2.12}$$

Em que $I^{B}_{(i/t)}$ é o índice de confiabilidade de importância do componente i; p_i é a confiabilidade do componente i, e R é a confiabilidade de todo o sistema. Baseado nesta definição e observando que $0 < p_i < 1$, a importância da confiabilidade do componente i pode ser escrito como:

$$I^{R}_{i}(t) = R(1_i, p_i) - R(0_i, p_i) \tag{2.13}$$

Em que p_i representa o vetor de confiabilidade dos componentes com o iésimo componente removido; 0_i representa a condição quando o componente i está com falha e 1_i a condição quando o componente está sempre no modo operacional (FIGUEIRÊDO; MACIEL, 2011).

Em outras palavras, a importância para a confiabilidade de cada componente é obtida da seguinte forma: É calculado o valor para a confiabilidade do sistema quando o componente está operacional. Em seguida é calculado esse valor quando o componente está não operacional; o valor da importância é dado pela diferença entre o valor quando o componente está operacional e quando ele não está operacional.

A avaliação de Importância para Disponibilidade (ID) pode ser obtida a partir de cálculos de importância para a confiabilidade. No entanto, ao invés de utilizar a confiabilidade, a métrica utilizada é a disponibilidade, incluindo, além do tempo médio para falha, o tempo de recuperação dos componentes (FIGUEIRÊDO; MACIEL, 2011). A equação utilizada para o cálculo do índice de Importância para a disponibilidade é apresentada pela Equação 2.14:

$$I^{A}_{s} = A(1_i, p_i) - A(0_i, p_i) \tag{2.14}$$

Os resultados podem ser obtidos de forma normalizada pelo maior valor de acordo com a Equação 2.15:

$$I_{ni} = \frac{I_i}{I_x} \tag{2.15}$$

Em que I_{ni} é o índice normalizado para o componente i; I_i é o valor do índice não normalizado para o componente i e, I_x é valor do maior índice não normalizado entre os componentes. No cálculo desse índice não existe o parâmetro tempo como dado de entrada, uma vez que está sendo calculada a disponibilidade em estado estacionário e não a confiabilidade do sistema num determinado tempo.

Na Tabela 2.1 apresentamos um resumo das características de cada método que impulsionou a utilização dessas estratégias nessa pesquisa.

Tabela 2.1: Características das Estratégias de Análise de Sensibilidade

Estratégias	**Características**	**Referência**
Variação um por um	1. Permite visualização gráfica; 2. Fácil de implementar; 3. Análise dos parâmetros individualmente; 4. Baixo custo computacional; 5. Não analisa a interação entre parâmetros; 6. Pode ser aplicado em todos os tipos de modelos e experimentos, desde de seja possível medir a influencia da variação dos parâmetros de entrada na saída.	(SALTELLI, 1999), (PIANOSI et al., 2016), (MONTGOMERY, 2012)
Sensibilidade Paramétrica	1. Fornece uma equação matemática; 2. O efeito do parâmetro X em análise sobre o parâmetro Y, a métrica de interesse, é capturado totalmente pela derivada $\partial Y/\partial X$; 3. Baixo custo computacional; 4. Aplicado em CTMC, RBD e em modelos que a métrica de interesse possua uma equação com variavéis que a representa.	(SALTELLI, 1999), (PIANOSI et al., 2016), (NORTON, 2015).
Diferença Percentual	1. Fornece valores minimos e máximos para o cálculo dos índices; 2. Fácil de implementar; 3. Baixo custo computacional. 4. Aplicado em CTMC, RBD e redes de Petri.	(MATOS et al., 2015), (SALTELLI, 1999),
Design of Experiments	1. Interações potenciais entre dois ou mais fatores; 2. Utilizada em experimento e modelagem analítica; 3. Auxilia no planejamento, análise e organização do experimento; 4. Elabora o *design* de experimento construindo a combinação entre as variáveis envolvidas (fatores e níveis) que serão utilizadas para coletar as métricas de interesse; 5. Aplicado em todos os modelos, desde que a métrica de interesse possua uma equação com variavéis que a represente.	(WULFF, 2012), (BARRENTINE, 1999). (BOX; WILSON, 1951)
Importância para Disponibilidade	1. Identifica o componente individualmente, considerando-o como um bloco; 2. Baixo custo computacional. 3. É aplicado em modelos RBD.	(KUO; ZHU, 2012) (BIRNBAUM, 1969)

As **estratégias de análise de sensibilidade** surgem como peça-chave que podem auxiliar a manter esses ambientes funcionando na maioria do tempo durante o ano. Esse subsídio, acontece à medida em que, essas estratégias indicam os componentes críticos com segurança e confiabilidade, de forma que os administradores dos sistemas podem implementar ações acertivas de melhorias nesses componentes visando melhorar a disponibilidade do sistema.

As estratégias de AS foram abordadas nessa pesquisa, tais como: Variação de um parâmetro por vez ou variação um por um, análise de sensibilidade diferencial paramétrica, diferença percentual, fatorial experimental ou *design of experiments* (DoE) e importância crítica para disponibilidade.

A estratégia **variação um por um** foi utilizada como um método de avaliação inicial, antes de uma análise detalhada. Além disso, essa estratégia tem objetivo de representar as dependências complexas entre a entrada e a saída (FREY; MOKHTARI; DANISH, 2003). Ela demanda tempo para análise, mas é simples de implementar, computacionalmente possui custo baixo e fornece uma visão parcial do comportamento do modelo (SALTELLI, 1999).

Além disso, na estratégia variação um por um a sua representação gráfica poder ser usada para completar os resultados de outras estratégias que utilizam apenas de métodos matemáticos estatísticos proporcionando uma melhor interpretação (FREY; MOKHTARI; DANISH, 2003). No entanto, devido ao fato de que os parâmetros apresentam grandezas e variações diferentes não é possível representar todos os parâmetros no mesmo gráfico, sendo assim a representação do comportamento da arquitetura acontece individualmente, sendo um gráfico para cada parâmetro.

Por outro lado, a estratégia **análise diferencial paramétrica**, foi abordada por possuir um baixo custo computacional, além de que a Derivada $\partial X/\partial Y$ captura totalmente o efeito do parâmetro X, parâmetro em análise, sobre o parâmetro Y, a métrica de interesse, (SALTELLI, 1999), (PIANOSI et al., 2016),(NORTON, 2015). Essa estratégia, assim como a **variação um por um** traz uma AS pontual dos efeitos da entrada sob a saída da arquitetura (PIANOSI et al., 2016). Ademais, essa estratégia foi utilizada como base para a criação da estratégia DASI, Índice Discreto Médio de Sensibilidade.

Em contrapartida, a **diferença percentual** possui características de não fornecer uma medida única de importância para cada entrada, ela responde a variações mínimas e máximas submetidas na entrada. Os valores mínimos e máximos dessa estratégia são utilizados para medir o efeito da entrada sobre a saída da arquitetura em análise.

A estratégia **DoE**, foi utilizada pela versatilidade que possui podendo ser aplicada em modelo analítico e em experimentos. Além disso, adotamos o *design* fatorial fracionário, porque as arquiteturas das Figuras 2.1 e 2.2 (vide páginas 24 e 25), têm mais de cinco variáveis que precisam de análise (MATHEWS, 2005). Essa estratégia, possibilita a organização do experimento, na medida em que os parâmetros, chamados aqui de fatores e níveis, são identificados. Esses dados são combinados, determinam a sequência e a quantidade de repetições para cada cenário. A partir desses dados organizados, inicia-se a coleta dos dados que serão medidos durante o experimento. A informação relevante nessa estratégia para a AS são os **efeitos** (MATOS et al., 2015).

A ferramenta *Minitab* [2] foi usada para realizar o *design of experiments*. Nessa ferramenta é possível obter várias outras métricas, no entanto a métrica que interessa para a análise de sensibilidade é o Efeito. Um *ranking* de efeitos é produzido, apresentando a interação entre os parâmetros e indicando a importância de cada parâmetro, a partir do efeito obtido por meio da estratégia DoE. Essa estratégia foi implementada na ferramenta *Mercury*[3] (MERCURY, 2014).

[2] *Minitab* é uma ferramenta para cálculo estatistico que permite inserir dados rapidamente e executar vários tipos de análises.

[3] *Mercury* é uma ferramenta de modelagem e avaliação de sistemas desenvolvida pelo grupo de pesquisa MoDCS da UFPE. Disponivel em: $http://www.modcs.org/?page_id = 1397$. Acesso em 10 de dezembro de 2016.

As estratégias índice de **Importância para a Confiabilidade - IC** e índice de **Importância para a Disponibilidade - ID** finalizam este capítulo apresentando os seus principais conceitos e método de análise da confiabilidade e disponibilidade. A ID teve como objetivo identificar os componentes mais importantes com relação à disponibilidade do sistema, essa métrica foi escolhida para ser analisada nessa pesquisa. Além disso, é importante destacar que a **Tabela 2.1** (vide página 35) foi elaborada consolidando as principais características das estratégias de análise de sensibilidade adotadas nessa pesquisa.

2.5 Considerações Finais

Os tópicos abordados nesse capítulo tais como: a computação em nuvem, dependabilidade, modelos de confiabilidade e disponibilidade e análise de sensibilidade nortearam o desenvolvimento desse trabalho, e possibilitaram estabelecimento de conexões desses, conceitos com os aplicados nessa pesquisa.

No próximo capítulo, são apresentados e descritos os principais trabalhos relacionados a essa pesquisa, enfatizando as pesquisas desenvolvidas nas áreas de dependabilidade e de análise de sensibilidade.

3

Trabalhos Relacionados

Neste capítulo são apresentados a descrição dos principais trabalhos correlatos ao desenvolvimento dessa pesquisa. Eles são divididos em duas seções destacando as pesquisas recentes relacionadas à Dependabilidade e Análise de Sensibilidade (AS) identificadas na literatura e que possui proximidade com o objetivo de pesquisa proposta.

3.1 Pesquisas sobre Dependabilidade

Conforme apresentado no Capítulo 2, a dependabilidade possui atributos que estão presentes na computação em nuvem, e trabalhos têm sido desenvolvidos considerando este aspecto para esses ambientes. Desse modo, há várias pesquisas que preconizam a análise e avaliação de dependabilidade de sistemas em computação nas nuvens por intermédio de modelos analíticos. Contudo, um pequeno número de pesquisas tem empregado a modelagem hierárquica com o objetivo de reduzir a complexidade do sistema e facilitar a análise de disponibilidade e confiabilidade do sistema.

É possível constatar trabalhos aplicados à melhoria da disponibilidade da computação em nuvem. No entanto, essa pesquisa pretende identificar os pontos que requerem melhoria na infraestrutura de nuvem, combinando a proposição de estratégias de análise de sensibilidade com modelagem hierárquica aliada, ainda, aos mecanismos de redundância para atuar na melhoria da disponibilidade desses ambientes. Em vista disso, nesta seção são apresentados os trabalhos relacionados à Dependabilidade, mediante a adoção de técnicas e ferramentas de modelagem. Os trabalhos a serem apresentados abordam aspectos específicos, como a disponibilidade e desempenho de arquiteturas e serviços, e outros abordam a eficiência operacional.

Os trabalhos publicados têm utilizado modelagem hierárquica para representar arquiteturas de computação em nuvem, realizando comparações de várias soluções e avaliações de métricas de dependabilidade nessas arquiteturas (CHUOB; POKHAREL; PARK, 2011), (DANTAS et al., 2012). Como exemplo, podemos citar o trabalho realizado por DANTAS et al. (2012).

DANTAS et al. (2012), investigaram os benefícios de um mecanismo de redundância *warm standby* em um ambiente de computação em nuvem na plataforma *Eucalyptus*. Uma abordagem de modelagem hierárquica e heterogênea baseada em diagramas de bloco de confiabilidade e cadeias de *Markov* para representar as arquiteturas básica e redundante. Registra-se ainda, que os autores comparam a disponibilidade das duas arquiteturas. Os resultados mostram uma maior confiabilidade para o sistema redundante proposto, bem como uma diminuição no tempo de inatividade anual.

Igualmente a DANTAS et al. (2012), o trabalho de CHUOB; POKHAREL; PARK (2011), também apresentou uma solução de nuvem privada *Eucalyptus*, para um centro de dados. Os resultados indicaram que a plataforma *Eucalyptus* atendeu aos requisitos de qualidade, segurança e privacidade solicitados pelo ambiente. A cadeia de *Markov* foi utilizada para calcular a disponibilidade de cada componente da nuvem.

Por outro lado, os autores LENK; PALLAS (2013), descreveram uma abordagem utilizando mecanismo de redundância *cold standby* para plataforma de nuvem baseada em modelagem de custos e disponibilidade com base em cadeias de *Markov*. Mostra-se que, usando uma configuração de nuvem *cold standby* a disponibilidade pode ser aumentada. Nesse trabalho, diferente de DANTAS et al. (2012), os pesquisadores não especificaram em qual tipo de nuvem a estratégia foi aplicada. Além disso, esse trabalho quando comparado ao de (CHUOB; POKHAREL; PARK, 2011), não utilizou a modelagem hierárquica para representação da arquitetura.

GHOSH et al. (2014) realizaram uma análise de disponibilidade da plataforma de nuvem como serviço (Iaas), onde máquinas físicas foram agrupadas em três tipos de conjuntos de provisionamento de recursos: *hot* (em execução), *warm* (ligado, mas não operante) e *cold* (desligado). Com a divisão realizada com base no consumo de energia e nas características de espera do provisionamento de recursos, foi realizada uma abordagem por modelagem estocástica para análise de dependabilidade nos três modos por meio de cadeia de *Markov* e redes de Petri.

Na contramão aos demais trabalhos, os autores em GUIMARAES et al. (2011) investigaram a disponibilidade de redes de computadores em quatro cenários diferentes, dois dos cenários apresentam redundância *cold standby*; Dos outros dois, um não apresenta redundância e o outro possui redundância física no *link*. Para a investigação da disponibilidade nesses cenários, utilizaram a rede de Petri estocástica como uma abordagem de modelagem permitindo uma avaliação analítica de cenários complexos. Além disso, utilizaram a Importância para a Confiabilidade com o objetivo de encontrar os componentes mais importantes do sistema, de acordo com a métrica de interesse. Uma limitação deste trabalho, consiste em usar um único tipo de mecanismo de redundância, além de não usar mecanismos de redundância nos componentes críticos identificados por cada cenário.

SILVA et al. (2013) apresentaram uma modelagem hierárquica de dependabilidade baseados em redes de Petri estocástica e RBD considerando a ocorrência de desastres. Um estudo de caso avaliou a disponibilidade dos cenários levando em conta as migrações dos *data centers* para diferentes locais e com diferentes tempos de migração de máquinas virtuais. Uma

restrição deste trabalho está na ausência do uso das estratégias de análise de sensibilidade para conduzir o processo de migração das máquinas virtuais.

SOUSA et al. (2014) propuseram uma estratégia de modelagem baseada numa 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 *Eucaliptus* é utilizado para demonstrar a viabilidade da estratégia da modelagem proposta. As estratégias de AS não foram aplicadas no processo de escolha das infraestruturas de nuvem a serem utilizadas.

Os pesquisadores BRILHANTE et al. (2014) avaliaram uma infraestrutura em nuvem híbrida através de modelos. Para isso consideraram redes de Petri estocásticas. O estudo de caso utilizou um gerador de injeção de falhas para verificar o modelo e os resultados de disponibilidade. Igualmente ao trabalho realizado por SOUSA et al. (2014), este trabalho não usou as estratégias de AS com a finalidade de escolher o componente que causaria maior ou menor impacto na disponibilidade do sistema, à medida em que o injetor de falha fosse atuando na infraestrutura.

Os autores COSTA et al. (2015) propuseram um modelo hierárquico para avaliar a disponibilidade da plataforma *MBaaS OpenMobster* enfatizando dois cenários: a arquitetura básica e o processo de recuperação automática. Os modelos construídos foram validados através de um *testbed* de injeção de falhas e reparos em um ambiente real. Tendo em conta as três camadas: *hardware*, sistema operacional, e o *MBaaS OpenMobster*, observamos *OpenMobster* sendo o componente do serviço mais crítico. Para este componente foi aplicada a estratégia de *failover* na máquina virtual Java, obteve-se 10% de redução no tempo de inatividade anual. Os autores ainda abordaram as questões de dependabilidade, e realizaram a validação de modelos, no entanto este trabalho ressaltou o processo de injeção e recuperação de falha realizado de forma automática.

SOUSA et al. (2014), ao contrário da nossa pesquisa, não aplicaram as estratégias de AS para conduzir o processo de escolha das infraestruturas de nuvem a serem utilizadas. Comparando o trabalho realizado pelos autores em (DANTAS et al., 2012) com o nosso, ele não apresenta mecanismos de redundância como alternativa para melhorar a disponibilidade do sistema. No entanto, enquanto LENK; PALLAS (2013) concentraram as pesquisas no mecanismo *cold standby* e não utilizaram a modelagem hierárquica, em contrapartida, o nosso trabalho apresenta os mecanismos de redundância *warm standby*, *cold standby* e ativo-ativo associados às estratégias de AS como instrumento para melhoria da disponibilidade da infraestrutura de nuvem.

Análogo a essa pesquisa, os autores GHOSH et al. (2014), utilizaram modelos para avaliar os aspectos de dependabilidade e de mecanismo de redundância para a plataforma de nuvem como serviço. No entanto, o mecanismo de redundância ativo-ativo não foi abordado, além disso, apresentaram um solução para uma plataforma de nuvem geral. Vale lembrar que os autores não utilizaram as estratégias de análise de sensibilidade como critério de aplicação e

chaveamento para os mecanismos de redundância apresentados.

Quando relacionamos os trabalhos realizados por GHOSH et al. (2014) e GUIMARAES et al. (2011), identificamos que os autores possuem uma limitação que consiste da não aplicação do mecanismo de redundância ativo-ativo nas suas arquiteturas. O autor em BRILHANTE et al. (2014) não utilizou mecanismos de redundância. Ao contrário de GUIMARAES et al. (2011), essa pesquisa propõe o uso de modelos de disponibilidade; propõe e adapta estratégias de análise de sensibilidade para identificar os pontos de melhoria do sistema e faz uso de mecanismos de redundância nos principais componentes apontados pelas estratégias.

Já em COSTA et al. (2015), os autores desenvolveram um trabalho que possui semelhanças com a nossa pesquisa; uma vez que utilizaram as estratégias de AS para apontar os componentes mais expressivos, no entanto não se procuparam em propor alternativas de melhoria de disponibilidade por meio de mecanismos de redundância para a arquitetura analisada.

Para essa pesquisa, os modelos em Diagrama de bloco de confiabilidade, rede de *Petri* e cadeia de *Markov* foram combinados para avaliação da disponibilidade do sistema. Inclusive, realizamos a combinação de estratégias de AS propostas e adaptadas para auxiliar no processo de identificação do pontos que requerem melhoria em uma infraestrutura de nuvem. Ademais, associamos os mecanismos de redundância que norteam os gestores no processo de decisão a escolher os mecanismos de redundância, adequados a serem atribuídos a infraestrutura de nuvem com ênfase na redução do *downtime* e aumento na disponibilidade.

3.2 Pesquisas sobre Análise de Sensibilidade

OAKLEY; O'HAGAN (2004), afirmam que em muitas áreas da Ciência e da Tecnologia, os modelos matemáticos são construídos para simular complexos fenômenos do mundo real. Os autores ainda afirmam que tais modelos são tipicamente implementados em grandes programas para computadores e também são muito complexos, de modo que, a maneira como o modelo responde às mudanças em suas entradas não é transparente.

Nessa dimensão, a Análise de Sensibilidade, busca compreender como as mudanças nas entradas do modelo influenciam as saídas. Inclusive, isso pode ser motivado simplesmente por um desejo de entender as implicações de um modelo complexo, contudo, muitas vezes surgem questionamentos sobre as incertezas dos reais valores de entrada que devem ser usados para uma determinada aplicação (OAKLEY; O'HAGAN, 2004).

AS é cada vez mais utilizada na modelagem para uma variedade de propósitos, incluindo avaliação de incerteza, calibração de modelos, avaliação diagnóstica, análise de controle e tomada de decisão (PIANOSI et al., 2016). Além disso, a AS é utilizada para determinar os parâmetros que causam maior impacto na disponibilidade, identificando quais componentes requerem atenção ao tentar alcançar maior disponibilidade em um sistema e orientar um processo de otimização (DANTAS et al., 2016).

Outro benefício importante a destacar é a identificação de parâmetros que podem ser

removidos sem efeitos significativos para os resultados (DANTAS et al., 2016). Modelos abrangentes, com dezenas de parâmetros, podem ser drasticamente reduzidos usando essa abordagem (DANTAS et al., 2016). Os resultados oriundos de uma AS podem ser resumidos numa lista de parâmetros de entrada classificados pela quantidade de contribuição que cada um tem no modelo de saída. Essa lista é denominada de *ranking* de sensibilidade (DANTAS et al., 2016).

Há vários estudos que propõem o uso da análise de sensibilidade aplicada em diversas áreas do conhecimento e alguns poucos aplicados a avaliação em ambiente de infraestrutura de nuvem por meio de modelos analíticos. Esta seção apresenta trabalhos que empregam AS para identificação dos parâmetros críticos dos sitemas computacionais e alguns outros trabalhos que aplicam a AS associada à avaliação de atributos de dependabilidade.

CAMPOLONGO; SALTELLI (1997), realizaram uma análise comparativa entre EAS [1] por meio do método proposto por Morris, SRC[2] e Sobol buscando identificar os fatores que influenciam os parâmetros de saída e além disso, descobrir os elementos comuns entre eles. No entanto, os autores OU; DUGAN (2000), utilizaram árvore de falha dinâmica e cadeias de *Markov* para realizar modelagem de sistema de assistência cardíaca, com o foco na métrica confiabilidade, os autores utilizaram a estratégia de análise de sensibilidade, conhecida como análise de importância, para identificar os principais componentes do sistema.

Ao contrário disso, SUN; XIAO; LANG (2011), avaliaram duas estratégias de análise de sensibilidade sendo elas: análise de correlação simples e regressão múltipla e perceberam que os resultados de cada estratégia individualmente não eram satisfatórios, então decidiram combiná-las. A partir dessa combinação surge uma estratégia que produziu resultados dentro do esperado.

No entanto, em (ZHOU; FU; CHIU, 2011), analisaram e propuseram um algoritmo de replicação genérico de balanceamento aleatório de carga para serviço de VoD P2P. Esse algoritmo, sugere o balaceamento de carga durante o *upload* dos vídeos. Para validar os resultados, foram utilizadas simulação e estratégia de AS, "variação um por um", foi empregada para avaliação do comportamento dos parâmetros de entrada do sistema e o seu reflexo na saída do sistema. Uma limitação identificada está relacionada à não utilização da AS na descoberta de quais componentes implementar o balanceamento de carga utilizando mais de uma estratégia de AS de modo a obter resultados com segurança.

Um estudo feito por MILLS; FILLIBEN; DABROWSKI (2011), aplicou um método de AS, que combina análise de correlação e *clustering* para demonstrar o quão relevante são as combinações e comportamentos dos parâmetros que podem ser identificados por uma infraestrutura de simulador de nuvem que destina-se a comparar os algoritmos de alocação de recursos.

Por outro lado, MATOS et al. (2012), propuseram um método baseado na AS paramétrica aplicada em cadeia de *Markov* para determinar os possíveis congestionamentos na disponibilidade

[1] *EAS*: Estratégia de análise de sensibilidade
[2] SRC: Coeficiente de regressão padrão

de um sistema virtualizado. Esse método foi aplicado para analisar as medidas de disponibilidade das máquinas virtuais associadas à falha, recuperação e migração de suas aplicações.

Em outro estudo, BEZERRA et al. (2014) investigaram técnicas de modelagem hierárquica para avaliar um serviço de *VoD* numa arquitetura básica, além disso usaram análise de sensibilidade diferencial paramétrica para identificar os pontos críticos dessa arquitetura. MATOS et al. (2015) utilizaram modelagem hierárquica e quatro EAS diferentes para determinar os parâmetros que causam o maior impacto sobre a disponibilidade de uma nuvem computacional móvel. Os resultados mostraram que as abordagens distintas forneceram resultados semelhantes quanto ao *ranking* de sensibilidade com exceções específicas. Uma avaliação combinada indica que a disponibilidade do sistema pode ser melhorada, eficazmente, concentrando-se em um conjunto reduzido de fatores que produzem grande variação na medida do seu interesse.

Os autores em DANTAS et al. (2016) investigaram um serviço hospedado em um ambiente de computação em nuvem privada. Modelos de disponibilidade foram apresentados, considerando o servidor de *Streaming* componentes que são necessários para o acessar serviço. Ademais, técnicas de modelagem hierárquica foram usadas para lidar com a representação do sistema, e duas estratégias de AS foram utilizadas para identificar os parâmetros que causam maior impacto na disponibilidade.

Os pesquisadores PREECE; MILANOVIC (2015) utilizaram EAS, para avaliar a sensibilidade de um sistema de potência. A análise teve por objetivo identificar quais os componentes críticos desse sistema, usando métodos probabilísticos. HE et al. (2016), abordaram um modelo em cadeia de *Markov* para avaliar e otimizar a confiabilidade e o desempenho de sistemas degradados baseados em nuvem, sujeitos à estratégia de redundância ativa e *cold standby*. Além disso, utilizaram a AS para verificação do comportamento do sistema e não como estrátegia de seleção dos mecanismos de redundância.

OU; DUGAN (2000) mostraram a aplicação de EAS em um contexto diferente da computação em nuvem. No entanto, por iguais razões que o nosso trabalho, os autores usam a AS para a identificação dos componentes relevantes, que orientaram o *designer* a identificar em qual ponto do sistema pode-se atuar para melhorar o desempenho de todo o sistema. Todavia, nossa pesquisa propõe a utilização de mais de uma EAS e, após essa análise, oferece para o *designer* opções de adotar os mecanismos de redundância com foco no incremento da disponibilidade do sistema.

É possível perceber uma pequena semelhança entre o trabalho de CAMPOLONGO; SALTELLI (1997), com essa pesquisa. A semelhança concentra-se na ideia da comparação das EAS. No entanto, as estratégias utilizadas pelos autores, são diferentes das utilizadas nesse trabalho, assim como os autores não utilizaram técnicas de modelagem para representação e avaliação do sistema, bem como não adotaram mecanismos de redundância, como alternativa de melhoria na disponibilidade da infraestrutura analisada.

Ainda que a pesquisa desenvolvida por SUN; XIAO; LANG (2011), siga a mesma dimensão de proximidade com esta pesquisa, sob a perspectiva da utilização de mais de uma

estratégia de AS e da criação de uma estratégia, no entanto, na pesquisa proposta foram definidas estratégias diferentes das apresentadas pelos autores.

Igualmente aos autores CAMPOLONGO; SALTELLI (1997), o trabalho realizado por SUN; XIAO; LANG (2011) não utilizou modelos para representação do sistema, nem a aplicação de mecanismos de redundância nas infraestruturas.

Uma limitação encontrada em MATOS et al. (2012) está relacionada a ações que podem ser tomadas após a identificação dos componentes mais importantes do sistema. Em contrapartida, a nossa pesquisa sugere alternativas de redundância após identificação dos congestionamentos do sistema.

Diferente desta pesquisa, BEZERRA et al. (2014) não utilizou mais de uma estratégia de AS para identificar os pontos críticos da arquitetura, bem como não avaliou o comportamento do sistema após a implementação de redundância nos componentes sinalizados pela AS. Em comum ao trabalho de BEZERRA et al. (2014), a pesquisa dos autores MATOS et al. (2015), possui uma limitação relacionada à não aplicação de mecanismo de redundância como opção para melhorar a disponibilidade do sistema, à medida em que os componentes selecionados apresentassem falha.

Os trabalhos realizados pelos autores ZHOU; FU; CHIU (2011) e MILLS; FILLIBEN; DABROWSKI (2011) são diferentes do trabalho proposto nesta pesquisa, apesar de ambos abordarem a temática da análise de sensibilidade e suas estratégias.

Por outro lado, ainda no trabalho de MATOS et al. (2015), identificou-se que este possui alguns pontos comuns com a nossa pesquisa, em razão de abordar várias estratégias de análise de sensibilidade e de utilizar modelos para representar e avaliar o sistema. Entretanto, esta tese apresenta a combinação de estratégias de análise de sensibilidade propostas e adaptadas com modelos hierárquicos para identificar pontos que precisam de melhorias, além da aplicação de mecanismos de redundância do tipo ativo-ativo, *cold standby* e *warm standby* com o objetivo de melhorar a disponibilidade do sistema.

Diferente do realizado nesta pesquisa, os trabalhos desenvolvidos pelos autores DANTAS et al. (2016) e PREECE; MILANOVIC (2015) não utilizaram mecanismos de redundância nos componentes que são comuns às EAS aplicadas. Em (PREECE; MILANOVIC, 2015) os autores realizaram um levantamento dos componentes que causam impacto no sistema de energia apontado pelas estratégias de AS, no entanto, nesta análise realizada pelos autores não utilizaram modelos analíticos para a representação do sistema.

Os autores HE et al. (2016) não utilizaram a modelagem hierárquica na representação da sua arquitetura. Diferente deles, utilizamos mais de uma EAS, para encontrar com exatidão, em qual componente da infraestrutura de nuvem, deve-se aplicar os mecanismos de redundância. Ademais apresentamos três possibilidades de mecanismos que podem ser implementadas, sendo eles: ativo-ativo, *warm* e *cold standby*, assim como utilizamos a modelagem hierárquica na representação desses ambientes.

Diferente dos demais trabalhos apresentados, esta pesquisa propõe a aplicação de modelos hierárquicos que representam as características desses sistemas, bem como a utilização

de estratégias propostas e consolidadas de análise de sensibilidade que foram combinadas e comparadas para serem utilizadas em uma infraestrutura de nuvem. Além disso, índices de sensibilidade foram criados com o objetivo de identificar os pontos relevantes e captar particularidades do sistema para aplicação dos mecanismos de redundância, com o intuito de promover ações de melhorias com relação à disponibilidade do sistema por meio desses mecanismos.

3.3 Comparação com os Trabalhos Relacionados

Nesta seção é introduzida a comparação dos trabalhos citados anteriormente em relação ao trabalho proposto nesta tese. Na Tabela 3.1 apresentamos os trabalhos correlatos em relação ao trabalho proposto nesta tese, considerando os formalismos matemáticos adotados para modelagem, os aspectos de dependabilidade avaliados e as estratégias de análise de sensibilidade utilizadas para infraestrutura de computação em nuvem.

Tabela 3.1: Trabalhos Relacionados

Referências	Modelos	Métrica	Análise de Sensibilidade
DANTAS et al. (2012)	CTMC, RBD	Disponibilidade	Não
CHUOB; POKHAREL; PARK (2011)	CTMC	Disponibilidade	Não
SILVA et al. (2013)	RBD, SPN	Disponibilidade	Não
LENK; PALLAS (2013)	CTMC	Disponibilidade	Não
GHOSH et al. (2014)	CTMC	Disponibilidade	Não
SOUSA et al. (2014)	SPN	Disponibilidade	Não
BRILHANTE et al. (2014)	SPN	Disponibilidade	Não
CAMPOLONGO; SALTELLI (1997)	Não	Não	Sim/3
ZHOU; FU; CHIU (2011)	Não	Não	Sim/1
MILLS; FILLIBEN; DABROWSKI (2011)	Não	Não	Sim/1
PREECE; MILANOVIC (2015)	Não	Não	Sim/4
SUN; XIAO; LANG (2011)	Não	Não	Sim/2
OU; DUGAN (2000)	CTMC	Confiabilidade	Análise de Importância
GUIMARAES et al. (2011)	SPN	Disponibilidade	Importância para Confiabilidade
HE et al. (2016)	CTMC	Disponibilidade	Variação um por um
MATOS et al. (2012)	CTMC	Disponibilidade	Diferencial Paramétrica (DP)
BEZERRA et al. (2014)	CTMC, RBD	Disponibilidade	DP
COSTA et al. (2015)	CTMC, RBD	Disponibilidade	DP e Variação um por um
DANTAS et al. (2016)	CTMC, RBD	Disponibilidade	DP e Variação um por um
MATOS et al. (2015)	CTMC, RBD	Disponibilidade	DP, DoE,Variação um por um e DP
Este Trabalho	CTMC, RBD, SPN	Disponibilidade	DoE,DP, Variação um por um,ICD e DASI

Na Tabela 3.1 são apresentadas as características dos 20 trabalhos relacionados a essa tese. As características de cada trabalho foram agrupadas quanto ao uso do modelo, à avaliação de disponibilidade e à aplicação ou não de estratégias para a avaliação da análise de sensibilidade.

Os trabalhos relacionados em DANTAS et al. (2012), CHUOB; POKHAREL; PARK (2011), LENK; PALLAS (2013), SILVA et al. (2013), SOUSA et al. (2014), BRILHANTE et al. (2014), GHOSH et al. (2014), CAMPOLONGO; SALTELLI (1997) ZHOU; FU; CHIU (2011), MILLS; FILLIBEN; DABROWSKI (2011) SUN; XIAO; LANG (2011), PREECE; MILANOVIC (2015), OU; DUGAN (2000), GUIMARAES et al. (2011), HE et al. (2016), MATOS et al. (2012) BEZERRA et al. (2014), COSTA et al. (2015), DANTAS et al. (2016) e MATOS et al. (2015), foram analisados em relação ao trabalho proposto, considerando as

características de modelagem, disponibilidade e estratégias de análise de sensibilidade e à quantidade de EAS que foram utilizadas.

Os trabalhos analisados em DANTAS et al. (2012), CHUOB; POKHAREL; PARK (2011), LENK; PALLAS (2013), SILVA et al. (2013), SOUSA et al. (2014), BRILHANTE et al. (2014) e GHOSH et al. (2014), adotaram diferentes tipos de modelo baseado em RBD, CTMC e rede de Petri, além disso avaliaram a disponibilidade do sistema, mas não utilizaram estratégias de análise de sensibilidade. Por outro lado, no trabalho relacionado em CAMPOLONGO; SALTELLI (1997), os pesquisadores abordaram três diferentes estratégias de AS para avaliação das suas arquiteturas, as questões relacionadas à modelagem e disponibilidade não foram abordadas pelos autores.

Em ZHOU; FU; CHIU (2011) e MILLS; FILLIBEN; DABROWSKI (2011), os pesquisadores abordaram apenas uma estratégia de AS para avaliação das suas arquiteturas, as questões relacionadas à modelagem e disponibilidade não foram abordadas. Em contrapartida, os autores SUN; XIAO; LANG (2011) e PREECE; MILANOVIC (2015), abordaram duas e quatro estratégias de AS para avaliação das suas arquiteturas, respectivamente. As questões relacionadas â modelagem e disponibilidade não foram abordadas.

O trabalho de OU; DUGAN (2000) empregou o modelo CTMC para representação do sistema, a disponilidade da infraestrutura não foi avaliada, no entanto, foi utilizada uma estratégia de AS para identificação dos componentes críticos do sistema. Em GUIMARAES et al. (2011), HE et al. (2016), MATOS et al. (2012), BEZERRA et al. (2014) e COSTA et al. (2015), houve o uso da estratégia de modelagem e estratégia de análise de sensibilidade para avaliação das suas arquiteturas.

Em DANTAS et al. (2016) e MATOS et al. (2015), houve a apresentação da modelagem hierárquica, avaliação da disponibilidade e apresentaram mais de duas estratégias de AS para avaliação do seus sistemas. No entanto, esses trabalhos possuem uma limitação com relação a não proposição de mecanismos de redundância para as suas arquiteturas, com o objetivo de melhorar a disponibilidade das mesmas.

Neste trabalho é abordado a proposição e adaptação de estratégias de análise de sensibilidade, combinada com a modelagem hierárquica para identificar os pontos que requerem melhoria na disponibilidade em uma infraestrutura de nuvem. Além disso, para obter uma alta confiabilidade no resultado da análise foram utilizadas cinco estratégias de AS e a combinação dos resultados gerados por elas. Para finalizar, foi proposto a implementação de modelos para representação de mecanismos de redundância nos pontos identificados.

3.4 Considerações Finais

Este capítulo apresentou os principais trabalhos correlatos ao estudo proposto. Embora existam trabalhos na literatura que proporcionam a avaliação da disponibilidade, por meio de modelos analíticos que enfatizam a aplicação das EAS para a infraestrutura de nuvem, e que abordem as questões de mecanismos de redundância, é bem verdade que não identificamos trabalhos que reúnam todos esses aspectos.

Um número reduzido de trabalhos realiza a avaliação de aspectos dependabilidade por meio de modelos analíticos e expressões matemáticas associados às estratégias de análise de sensibilidade. Todavia, não foram identificados trabalhos que propuseram a alteração de EAS já existentes.

Um número reduzido de trabalhos apresentou uma estratégia de modelagem hierárquica para avaliação de dependabilidade de infraestruras de nuvem, mas poucos trabalhos avaliam o impacto da atribuição de mecanismos de redundância aos componentes e serviços hospedados na nuvem.

Embora existam trabalhos que apresentem várias estratégias de análise de sensibilidade para identificação dos pontos de criticidade dos sistemas, por meio de modelagem hierárquica, não foram identificados trabalhos que proporcionem uma metodologia que aplique cinco EAS e que realize uma comparação entre essas EAS, dando origem a um *ranking*, com o propósito de indicar em qual componente deve-se aplicar os mecanismos de redundância para atuar na melhoria da disponibilidade do sistema.

O próximo capítulo apresenta e descreve a proposição e adaptação de estratégias de análise de sensibilidade que foram utilizados nessa pesquisa, sendo elas o índice de discreto médio de sensibilidade, o qual denominamos de DASI e importância crítica para a disponibilidade, denominado de ICD.

4

Estratégias para Análise de Sensibilidade: DASI e ICD

Neste capítulo, abordamos a proposição e adaptação de estratégias de análise de sensibilidade denominadas nessa pesquisa de: i) Índice Discreto Médio de Sensibilidade (DASI) e ii) Índice de Importância Crítica para a Disponibilidade (ICD). Esses índices foram obtidos a partir de modificações realizadas na estratégia de sensibilidade diferencial paramétrica e no índice de importância crítica para a confiabilidade. A implementação conjunta dessas com as estratégias variação um por um, com a diferença percentual e com o *design of experiments* – DoE, tem sinalizado resultados relevantes na identificação dos pontos que requerem melhoria na disponibilidade em uma infraestrutura de nuvem, servindo como uma estratégia adicional ou complementar às frequentementes utilizadas para a análise de disponibilidade de sistemas em nuvens.

4.1 Introdução

As razões que motivaram as alterações dessas estratégias foram baseadas em análise e pesquisa realizadas na literatura em HAMBY (1994), CAMPOLONGO; SALTELLI (1997), OU; DUGAN (2000), MATOS et al. (2012), BEZERRA et al. (2014) , MATOS et al. (2015) e DANTAS et al. (2016), como visto no Capítulo 3. Para a estratégia de sensibilidade diferencial paramétrica, um dos motivos que intensificaram a necessidade de modificação esteve relacionado ao fato de que o cálculo do índice de sensibilidade dessa estratégia é realizado para um parâmetro fixo; consequentemente isso faz com que permaneçamos na dependência do parâmetro. No entanto, ao propor a alteração nessa estratégia eliminamos a submissão de um único parâmetro, uma vez que propomos calcular a estratégia para mais de um parâmetro. Um outro motivo, que alavancou essa modificação está relacionado à utilização da estratégia de sensibilidade diferencial paramétrica em sistemas que visam avaliar as questões de dependabilidade dos sistemas.

Por outro lado, há três motivos que impulsionaram realizar alterações na estratégia índice de importância crítica para confiabilidade são: i) O primeiro motivo está relacionado

ao fato de que essa estratégia trabalha na identificação dos componentes críticos propondo a avaliação de dois caminhos da arquitetura. Um deles é um caminho operacional, em que a métrica de interesse é avaliada quando os componentes estão funcionando e um outro chamado de não operacional, em que é verificada a métrica de interesse considerando que os componentes falham; ii) O segundo motivo está relacionado ao fato de que essa estratégia, de acordo com MARVIN RAUSAND (2003), prioriza ações de manutenção em sistemas complexos, como os ambientes de computação em nuvem, sugerindo a atuação inicial nos componentes identificados pelo caminho falho; iii) Finalizando, o terceiro motivo está relacionado ao fato de que essa estratégia refere-se ao componente, e não aos seus parâmetros individuais, isso possibilita uma visão ampla do sistema, tornando-a relevante num processo de análise de um sistema.

Essa estratégia de acordo com MARVIN RAUSAND (2003) prioriza ações de manutenção em sistemas complexos, como os ambientes de infraestrutura de nuvem. Desse modo, iniciaremos a condução deste capítulo a partir da apresentação das mudanças realizadas nas estratégias índice de sensibilidade diferencial parámetrica e índice de importância crítica para a confiabilidade.

4.2 *Discrete Averaged Sensitivity Index* - DASI

Inicialmente, apresentaremos estratégias de análise de sensibilidade (EAS) diferencial paramétrica, em virtude dessa estratégia ter proporcionado a origem da estratégia *Discrete Averaged Sensitivity Index* (Índice Discreto Médio de Sensibilidade), denominada de DASI. A EAS diferencial paramétrica usa a derivada parcial para obter o índice de sensibilidade do parâmentro Y (métrica de interesse) que depende de um parâmetro θ (θ corresponde a taxa de falha e de reparo). Estamos apresentando esta equação de derivada parcial com a normalização, na medida em que multiplicamos o resultado da derivada parcial pelo parâmetro θ e dividimos pela métrica de interesse Y($\frac{\theta}{Y}$).

$$S^*_\theta(Y) = \frac{\partial Y}{\partial \theta}\left(\frac{\theta}{Y}\right) \tag{4.1}$$

A sensibilidade diferencial paramétrica calcula a taxa de variação em um parâmetro específico. Ela é dependente da configuração do parâmetro e traz um resultado pontual. A EAS chamada de Índice Discreto Médio de Sensibilidade (DASI) foi idealizada a partir da análise de sensibilidade diferencial paramétrica, no entanto, com a proposição de uma alteração na Equação da derivada parcial. A DASI baseia-se na existência de uma equação algébrica que descreve a relação entre a medida de interesse e os parâmetros de entrada. No entanto, a DASI propõe reduzir a questão da dependência da configuração associada a um único parâmetro que temos na sensibilidade diferencial paramétrica.

A DASI é obtida calculando as derivadas parciais a partir da medida de interesse para um conjunto de parâmetros de entrada. No entanto, esses não possuem mais um valor fixo, esses

valores são variáveis obdecendo a um critério previamente definido, sendo ele: a utilização de uma faixa de valores entre -50% a +50% do valor nominal do parâmetro. Para esse cálculo posicionamos o valor nominal no centro e calculamos o percentual sugerido.

Após o cálculo da derivada parcial para todos os parâmetros do *range*, é realizada a média desses valores, sendo obtido o valor médio das derivadas parciais. Esse valor médio obtido compõe o índice de sensibilidade da estratégia DASI para a métrica de interesse. Por exemplo, para encontrar a DASI do parâmetro A (métrica de interesse), que depende do parâmetro θ, o qual corresponde à taxa de falha e de reparo de cada componente, MTTF e MTTR respectivamente. O MTTF e MTTR são definidos como: MTTF=$1/\lambda$ e MTTR=$1/\mu$. O índice de sensibilidade da estratégia DASI pode ser obtido através da Equação 4.2:

$$DASI^{*}_{\theta}(A) = \frac{\sum_{i=1}^{n} \frac{\partial A}{\partial \theta} \left(\frac{\theta}{A}\right)|_{\theta=\theta i}}{n} \tag{4.2}$$

Em que o parâmetro n, representa a quantidade de parâmetros para realizar a média. O valor θ irá variar de acordo com uma variação do valor nominal do parâmetro entre uma faixa de $-$ 50% e $+$ 50%. Na Figura 4.1 apresentamos um modelo em RBD série que utilizaremos para calcular a disponibilidade desse sistema e na sequência aplicaremos a estratégia DASI para obtermos o *ranking* de sensibilidade para o modelo. A disponibilidade (A) desse modelo é calculada por meio da Equação 4.3, de acordo com os parâmetros de tempo de falha e de reparo de cada bloco.

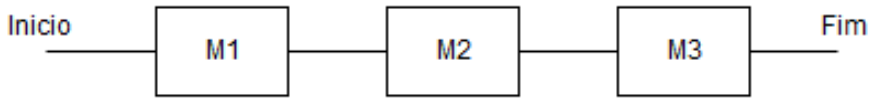


Figura 4.1: Modelo RBD série

$$A = A(M1) \times A(M2) \times A(M3) \tag{4.3}$$

Lembrando que a disponibilidade de cada bloco do modelo pode ser apresentada por meio da Equação:

$$A = (\frac{\mu}{\lambda + \mu}) \tag{4.4}$$

Desse modo, a disponibilidade para o modelo da Figura 4.1 pode ser apresentada por meio da Equação:

$$A_1 = (\frac{\mu_1}{\lambda_1 + \mu_1}) \times (\frac{\mu_2}{\lambda_2 + \mu_2}) \times (\frac{\mu_3}{\lambda_3 + \mu_3}) \tag{4.5}$$

Em que, λ e μ representam a taxa de falha e de reparo do modelo respectivamente. Para

calcular o *ranking* de sensibilidade por meio da estratégia DASI para o modelo representado na Figura 4.1, seguiremos os passos:

a) Definir a métrica de interesse que pretende-se obter a partir do modelo, nesse caso a disponibilidade. Apresentar a equação geral que representa o modelo. Aplicar a derivada parcial na Equação 4.14 de disponibilidade e obter como resultado a Equação 4.6. Essa é a equação que usaremos para calcular o índice de sensibilidade DASI;

b) Obter as equações de derivada parcial para cada parâmetro do modelo. Como cada bloco do modelo possui 2 (dois) parâmetros λ e μ, dessa forma, obteremos 6 (seis) equações;

c) Realizar o cálculo de cada equação substituindo os parâmetros na equação. Para cada equação, os parâmetros de entrada foram variados de -50% a +50% do valor nominal, o valor nominal também é considerado, desse modo adotamos 11 (onze) valores para cada parâmetro. Após isso, realizamos a média desses resultados. Esses valores resultantes foram utilizados para compor o índice de análise de sensibilidade DASI;

d) Classificar em ordem decrescente os resultados obtidos no passo anterior. Desse modo, obtemos o *ranking* de sensibilidade DASI.

$$
\begin{aligned}
DASI^{*}_{1\,\theta_i}(A_1) = & \frac{\sum_{i=1}^{n} \frac{\partial A_1(M1)}{\partial \theta} \left(\frac{\theta}{A_1}\right) |_{\theta=\theta i}}{n} \times A_1(M2) \times A_1(M3) + \\
& A_1(M1) \times \frac{\sum_{i=1}^{n} \frac{\partial A_1(M2)}{\partial \theta} \left(\frac{\theta}{A_1}\right) |_{\theta=\theta i}}{n} \times A_1(M3) + \\
& A_1(M1) \times A_1(M2) \times \frac{\sum_{i=1}^{n} \frac{\partial A_1(M3)}{\partial \theta} \left(\frac{\theta}{A_1}\right) |_{\theta=\theta i}}{n}
\end{aligned}
\tag{4.6}
$$

A Equação da disponibilidade 4.14 é a base para o cálculo das expressões da derivada parcial, correspondente para cada parâmetro. Desse modo, as derivadas parciais para a métrica disponibilidade em relação a cada parâmetro são apresentadas por meio das seguintes Equações: 4.7, 4.8, 4.9, 4.10, 4.11 e 4.12.

A Equação 4.7 representa a derivada parcial para a métrica disponibilidade do bloco M1 em relação ao parâmetro λ_{M1} e a Equação 4.8 representa a derivada parcial para a métrica disponibilidade do bloco M1 em relação ao parâmetro μ_{M1}. As expressões derivativas para M2 e M3 são apresentadas por meio das Equações 4.9, 4.10, 4.11 e 4.12.

$$\frac{\partial A_1^*(M1)}{\partial \lambda_{M1}} \frac{\lambda_{M1}}{A_1} = -\frac{\lambda_{M2}\lambda_{M3}\mu_{M1}}{(\mu_{M1}+\lambda_{M1})^2(\lambda_{M2}+\mu_{M2})(\lambda_{M3}+\mu_{M3})} \frac{\lambda_{M1}}{A_1} \tag{4.7}$$

$$\frac{\partial A_1^*(M1)}{\partial \mu_{M1}} \frac{\mu_{M1}}{A_1} = -\frac{\lambda_{M2}\lambda_{M3}\mu_{M1}}{(\lambda_{M1}+\mu_{M1})^2(\lambda_{M2}+\mu_{M2})(\lambda_{M2}+\mu_{M2})} \frac{\mu_{M1}}{A_1} + \frac{\lambda_{M2}\lambda_{M3}}{(\lambda_{M1}+\mu_{M1})(\lambda_{M2}+\mu_{M2})(\lambda_{M3}+\mu_{M3})} \frac{\mu_{M1}}{A_1} \tag{4.8}$$

$$\frac{\partial A_1^*(M2)}{\partial \lambda_{M2}} \frac{\lambda_{M2}}{A_1} = -\frac{\mu_{M1}\lambda_{M2}\lambda_{M3}}{(\lambda_{M2}+\mu_{M2})^2(\lambda_{M1}+\mu_{M1})(\lambda_{M3}+\mu_{M3})} \frac{\lambda_{M2}}{A_1} + \frac{\mu_{M1}\lambda_{M3}}{(\lambda_{M1}+\mu_{M1})(\lambda_{M2}+\mu_{M2})(\lambda_{M3}+\mu_{M3})} \frac{\lambda_{M2}}{A_1} \tag{4.9}$$

$$\frac{\partial A_1^*(M2)}{\partial \mu_{M2}} \frac{\mu_{M2}}{A_1} = -\frac{\mu_{M1}\lambda_{M2}\lambda_{M3}}{(\lambda_{M2}+\mu_{M2})^2(\lambda_{M1}+\mu_{M1})(\lambda_{M3}+\mu_{M3})} \frac{\mu_{M2}}{A_1} \tag{4.10}$$

$$\frac{\partial A_1^*(M3)}{\partial \lambda_{M3}} \frac{\lambda_{M3}}{A_1} = -\frac{\mu_{M1}\lambda_{M2}\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})^2(\lambda_{M1}+\mu_{M1})(\lambda_{M2}+\mu_{M2})} \frac{\lambda_{M3}}{A_1} + \frac{\mu_{M1}\lambda_{M2}}{(\lambda_{M1}+\mu_{M1})(\lambda_{M2}+\mu_{M2})(\lambda_{M3}+\mu_{M3})^2} \frac{\lambda_{M3}}{A_1} \tag{4.11}$$

$$\frac{\partial A_1^*(M3)}{\partial \mu_{M3}} \frac{\mu_{M3}}{A_1} = -\frac{\mu_{M1}\lambda_{M2}\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})^2(\lambda_{M1}+\mu_{M1})(\lambda_{M2}+\mu_{M2})} \frac{\mu_{M3}}{A_1} \tag{4.12}$$

Essa estratégia utiliza uma faixa de valores para λ e para μ, os quais foram obtidos a partir dos valores nominais dos parâmetros fornecidos pelos fabricantes. Se eles não forem fornecidos, podem ser obtidos por meio de experimentos. Na Tabela 4.1, apresentamos os parâmetros de entrada utilizados no cálculo do índice de sensibilidade DASI para o modelo da Figura 4.1.

Na Tabela 4.2, apresentamos o *ranking* de sensibilidade obtido por meio da estratégia DASI, a partir dos parâmetros de entrada fornecidos pela Tabela 4.1. Além disso, na Tabela 4.2 apresentamos o resultado do *ranking* de sensibilidade utilizando a estratégia sensibilidade diferencial paramétrica (SDP). O objetivo de apresentar os dois resultados das estratégias na mesma tabela é para realizar uma comparação entre a estratégia proposta com a SDP. As duas

Tabela 4.1: Parâmetros de entrada para o cálculo do índice DASI

	Valores (Horas)										
Parâmetros	**1**	**2**	**3**	**4**	**5**	**Nominal**	**7**	**8**	**9**	**10**	**11**
$1/\mu$M1	1/0,9	1/0,8	1/0,7	1/0,6	1/0,5	1	1/1,1	1/1,2	1/1,3	1/1,4	1/1,5
$1/\lambda$M1	1/162	1/144	1/126	1/108	1/90	1/180	1/198	1/216	1/234	1/252	1/270
$1/\mu$M2	1/1,8	1/1,6	1/1,4	1/1,2	1	1/2	1/2,2	1/2,4	1/2,6	1/,8	1/3
$1/\lambda$M2	1/302,4	1/268,8	1/235,2	1/201,6	1/168	1/336	1/369,6	1/403,2	1/436,8	1/470	1/504
$1/\mu$M3	1/2,7	1/ 2,4	1/2,1	1/1,8	1/1,5	1/3	1/3,3	1/3,6	1/3,9	1/4,2	1/ 4,5
$1/\lambda$M3	1/360	1/320	1/280	1/240	1/200	1/400	1/440	1/480	1/520	1/560	1/600

estratétegias possuem resultados semelhantes e satisfatórios, no entanto, nesse exemplo, observamos diferenças entre os dois *rankings*, a estratégia DASI elencou como os principais elementos os parâmetros referente a taxa de reparo de cada bloco. Esse resultado deve ser destacado, em virtude de que os valores atribuídos à taxa de reparo serem pequenos e normalmente as empresas disporem de uma equipe de manutenção para toda a empresa.

Tabela 4.2: *Ranking* de sensibilidade obtido através das estratégias DASI e SDP para o RBD série

Parâmetro	$DASI^*_{\theta i}(\lvert A\rvert)$	**Parâmetro**	$SDP^*_{\theta i}(\lvert A\rvert)$
μ_{M3}	0,008166462	μ_{M3}	0,00744416
μ_{M2}	0,006494838	λ_{M3}	0,00744416
μ_{M1}	0,006071946	λ_{M2}	0,00591716
λ_{M3}	0,007430132	μ_{M2}	0,00591716
λ_{M2}	0,005908682	λ_{M1}	0,00552486
λ_{M1}	0,005517466	μ_{M1}	0,00552486

Na Figura 4.2 apresentamos um outro exemplo de configuração utilizando um modelo RBD série e paralelo. A disponibilidade foi calculada e na sequência aplicamos a estratégia DASI para obtermos o *ranking* de sensibilidade para os componentes desse modelo.

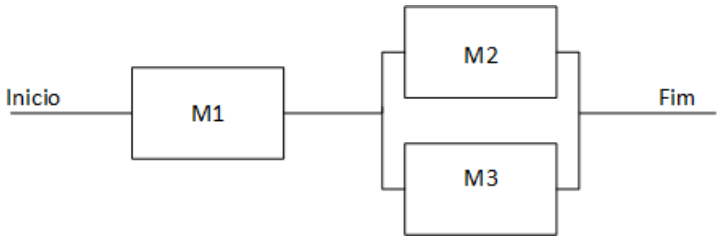


Figura 4.2: Modelo RBD série e paralelo

A disponibilidade para o modelo da Figura 4.2 é obtida por meio da Equação 4.13:

$$A_2 = (M1) \times (1 - (1 - M2) \times (1 - M3) \tag{4.13}$$

Reescrevendo a Equação 4.13 da disponibilidade do modelo em termos de taxa de falha e reparo, obtivemos a Equação 4.14:

$$A_2 = (\frac{\mu_1}{\lambda_1+\mu_1}) \times (1-(1-(\frac{\mu_2}{\lambda_2+\mu_2}))) + (1-(\frac{\mu_3}{\lambda_3+\mu_3})) \tag{4.14}$$

A expressão que traz o índice de sensibilidade DASI para esse modelo por meio da derivada parcial pode ser obtido pela Equação 4.15:

$$\begin{aligned} DASI^*_{2\,\theta_i}(A_2) = & \frac{\sum_{i=1}^{n} \frac{\partial A_2(M1)}{\partial \theta}(\frac{\theta}{A_2})|_{\theta=\theta i}}{n} \times (1-(1-A_2(\mathrm{M2}))) \times A_2(M3) + \\ & A_2(M1) \times \frac{\sum_{i=1}^{n} \frac{\partial A_2(M2)}{\partial \theta}(\frac{\theta}{A_2})|_{\theta=\theta i}}{n} + \times (1-D_2(M3)) + \\ & A_2(M1) \times (1-(1-A-2(\mathrm{M2}))) \times \frac{\sum_{i=1}^{n} \frac{\partial D_2(M3)}{\partial \theta}(\frac{\theta}{A_2})|_{\theta=\theta i}}{n} \end{aligned} \tag{4.15}$$

A Equação 4.16 representa a derivada parcial para a métrica disponibilidade do bloco M1 em relação ao parâmetro λ_{M1} e a Equação 4.17 representa a derivada parcial para a métrica disponibilidade do bloco M1 em relação ao parâmetro μ_{M1}. As expressões derivativas parciais para os blocos M2 e M3 são apresentadas por meio das Equações 4.18, 4.19, 4.20 e 4.21.

$$\frac{\partial A_2^*(M1)}{\partial \lambda_{M1}}(\frac{\lambda_{M1}}{A_2}) = -\frac{\mu_{M2}(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})^2}(\frac{\lambda_{M1}}{A_2}) \tag{4.16}$$

$$\begin{aligned} \frac{\partial A_2^*(M1)}{\partial \mu_{M1}}(\frac{\mu_{M1}}{A_2}) = & -\frac{\mu\lambda_{M2}(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})^2}(\frac{\mu_{M1}}{A_2}) + \\ & \frac{\lambda_{M2}(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})}(\frac{\mu_{M1}}{A_2}) \end{aligned} \tag{4.17}$$

$$\begin{aligned} \frac{\partial A_2^*(M2)}{\partial \lambda_{M2}}(\frac{\lambda_{M2}}{A_2}) = & -\frac{\mu\lambda_{M2}(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M1}+\mu_{M1})+(\lambda_{M2}+\mu_{M2})^2}(\frac{\lambda_{M2}}{A_2}) + \\ & \frac{\mu(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})}(\frac{\lambda_{M2}}{A_2}) \end{aligned} \tag{4.18}$$

$$\frac{\partial A_2^*(M2)}{\partial \mu_{M2}}(\frac{\mu_{M2}}{A_2}) = \frac{\mu(1-\frac{\lambda_{M3}}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M1}+\mu_{M1})+(\lambda_{M2}+\mu_{M2})^2}(\frac{\mu_{M2}}{A_2}) \tag{4.19}$$

$$\frac{\partial A_2^*(M3)}{\partial \lambda_{M3}}(\frac{\lambda_{M3}}{A_2}) = \frac{\mu\lambda_{M2}(\frac{\lambda_3}{(\lambda_{M3}+\mu_{M3})}\frac{1}{(\lambda_{M3}+\mu_{M3})})}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})^2}(\frac{\lambda_{M3}}{A_2}) \tag{4.20}$$

$$\frac{\partial A_2^*(M3)}{\partial \mu_{M3}}(\frac{\mu_{M3}}{A_2}) = \frac{\mu\lambda_{M2}\lambda_{M3}}{(\lambda_{M2}+\mu_{M2})+(\lambda_{M1}+\mu_{M1})+(\lambda_{M3}+\mu_{M3})^2}(\frac{\mu_{M3}}{A_2}) \tag{4.21}$$

Tabela 4.3: *Ranking* de sensibilidade obtidos através das estratégias DASI e SDP para o RBD série e paralelo

Parâmetro	$DASI^*_{\theta i}(\|A\|)$	**Parâmetro**	$SDP^*_{\theta i}(\|A\|)$
μ_{M1}	0,00569202647	μ_{M1}	0,00552486187
λ_{M1}	0,00506543231	λ_{M1}	0,00552486187
μ_{M2}	0,00004806553	μ_{M2}	0,00004378962
μ_{M3}	0,00004796529	λ_{M2}	0,00004378962
λ_{M2}	0,00004372694	λ_{M3}	0,00004372235
λ_{M3}	0,00003569433	μ_{M3}	0,00004372235

Na Tabela 4.3, apresentamos o resultado para a análise de sensibilidade por meio do método DASI e SDP. O propósito dessa comparação é mais uma vez mostrar que elas apresentam valores próximos e que a alteração da estratégia proposta em relação à SDP traz um resultado semelhante, além disso identificamos que a taxa de reparo, representada pelos parâmetros μ_{M1}, μ_{M2} e μ_{M3}, foi destacada em relação à taxa de falha, como elemento importante entre os demais parâmetros dos três blocos. Essa observação é relevante, uma vez que esse parâmetro possui, na maioria dos casos, tempos pequenos de recuperação e atuarmos com maior rapidez no mesmo ou aplicarmos estratégias para que eles não ocorram, obteremos uma indisponibilidade mínima para o sistema.

A partir dos exemplos apresentados, o uso da estratégia DASI possibilita verificar o comportamento do sistema sob o ponto de vista da AS para uma faixa de parâmetros. Desse modo, o sistema não está dependente de um único parâmetro para encontrar o índice de sensibilidade para o sistema. Apresenta ainda, em alguns casos, que no *ranking* de sensibilidade, a taxa de reparo do componente tem destaque em relação à taxa de falha do mesmo.

4.3 Importância Crítica para a Disponibilidade - ICD

Antes de iniciarmos, é importante, mencionar que a estratégia Importância Crítica para a Confiabilidade será apresentada, por razões de ser essa estratégia que foi modificada para dar origem à estratégia de Importância Crítica para a Disponibilidade - ICD.

Os autores KUO; ZUO (2003b), propuseram duas medidas de importância crítica para a confiabilidade, que analisa a confiabilidade no caminho operacional (sistema funcionando) e não operacional (sistema com falha). Seguindo essa premissa, adaptamos esse conceito para a disponibilidade, realizando adequações necessárias na equação que formaliza essa estratégia para a confiabilidade e ajustamos para a disponibilidade.

Essa estratégia tem como calcular a disponibilidade do sistema por meio do caminho operacional e do caminho não operacional. Desse modo, a disponibilidade é calculada inicialmente, observando os componentes operacionais num determinado caminho e na sequência a disponibilidade é calculada para um mesmo caminho quando os componentes não estão operacionais.

A formalização desse método é apresentada a partir das seguintes considerações: A Importância Crítica para a Disponibilidade (ICD) do componente i quando o sistema está funcionamento, é indicado pelo $I_{cs}(i;p)$, sendo portanto, definida como a probabilidade de que o componente i funciona e, é crítica para o funcionamento do sistema, dado que o sistema está funcionando. Esse índice pode ser expresso matematicamente por meio da Equação 4.22:

$$I_{cs}(i;p) = \frac{pi(A(1_i,p) - A(0_i,p))}{A(p)} \tag{4.22}$$

A ICD do componente i, quando o sistema está em falha, é denotado pelo $I_{cf}(i;p)$, sendo definido como a probabilidade de que o componente i falha, dado que o sistema apresenta uma falha. Matematicamente pode ser expressa pela Equação 4.23:

$$I_{cf}(i;p) = \frac{qi(A(1_i,p) - A(0_i,p))}{1 - A(p)} \tag{4.23}$$

O caminho definido como caminho de falha (onde existem componentes não operacionais), pode ser utilizado no diagnóstico de falhas. Quando um sistema falha, o componente com a maior $I_{cf}(i;p)$ é o mais provável de ter causado a falha e, portanto, deve ser verificado primeiramente.

O vetor de disponibilidade do componente é representado por pi quando está funcionado; 0_i representa a condição de falha do componente i; 1_i representa o componente funcionando no modo estacionário; q_i representa um vetor de indisponibilidade do componente quando não está funcionando. $A(1_i,p)$ representa a disponibilidade individual do componente quando está funcionando e $A(0_i,p)$ representa a disponibilidade individual do componente quando não está funcionando. $A(p)$ representa a disponibilidade total do sistema quando todos os componentes estão funcionando. Na Figura 4.3, apresentamos um exemplo no qual aplicaremos a estratégia ICD.

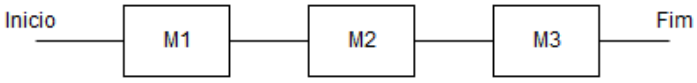


Figura 4.3: Modelo RBD série para aplicação da estratégia ICD

As etapas que são necessárias para o obter a estratégia ICD na Tabela 4.4.

Tabela 4.4: Etapas da estratégia ICD para o caminho operacional

Etapas	Descrição
1	Calcular a disponibilidade individualmente de cada bloco (p_i), por meio da equação $A_{Mx} = \mu_{Mx}/(\lambda_{Mx}+\mu_{Mx})$, em que Mx representa os blocos M1, M2 e M3. O resultado desse cálculo encontra-se na Tabela 4.5, (vide página 57);
2	Calcular a disponibilidade total do sistema (AS) por meio da expressão $AS = AM_1 \times AM_2 \times AM_3$. O resultado desse cálculo encontra-se na Tabela 4.5, (vide página 57);
3	Calcular a disponibilidade do sistema quando cada bloco estiver funcionando. Nesse caso, por exemplo, quando o bloco 1 está funcionando a disponibilidade é dada por meio da expressão $A = 1 \times AM_2 \times AM_3$. o resultado desse cálculo encontra-se na Tabela 4.6, (vide página 58);
4	Calcular a disponibilidade do sistema quando cada bloco não estiver funcionando. Nesse caso, por exemplo, a disponibilidade é dada pela equação $AS = AM_1 \times AM_2 \times AM_3$, como a disponibilidade do bloco é 0 (zero), quando ele não está operacional, desse modo a disponibilidade do sistema também o é, em virtude de ter na Equação uma das parcelas o valor 0 (zero). O resultado desse cálculo encontra-se na Tabela 4.6), (vide página 58);
5	Calcular a diferença entre a disponibilidade individual do bloco funcionando subtraindo a disponibilidade individual do bloco não funcionando, isso é obtido através da Equação $(A(1_i,p) - A(0_i,p))$. O resultado desse cálculo encontra-se na Tabela 4.6 (vide página 58);
6	Calcular o índice de importância crítica para disponibilidade quando o sistema está funcionando $I_{cs}(i;p)$ é obtido através do resultado dessa **etapa 6** . O resultado da **etapa 5**, é o resultado da Equação $(A(1_i,p) - A(0_i,p))$. Esse resultado dividimos pela disponibilidade do sistema (AS) **etapa 2** e na sequência multiplicamos esse resultado pela disponibilidade individual do bloco em funcionamento **etapa 3**. O resultado desse cálculo encontra-se na Tabela 4.6, (vide página 58);

Após a execução da **etapa (6)** determinamos o ICD para a disponibilidade no caminho em funcionamento. Enfatizamos que a Equação 4.24 foi utilizada para o cálculo da disponibilidade no caminho operacional.

$$I_{cs}(i;p) = \frac{pi(A(1_i,p) - A(0_i,p))}{A(p)} \tag{4.24}$$

Na Tabela 4.6, o resultado apresentado para a coluna índice $I_{cs}(i;p)$ não está em ordem decrescente, mas observamos que o bloco M2 é o componente de maior relevância e para esse bloco, ações de contigência devem ser providenciadas com o objetivo de otimizar a disponibilidade do sistema visando diminuir a probabilidade de risco do sistema.

Tabela 4.5: Parâmetros e resultados para da disponibilidade de cada bloco do sistema

Parâmetro	MTTF	MTTR	Disponibilidade individual de cada bloco (p_i) (etapa 1)	Disponibilidade do sistema (etapa 2)
$M1$	100	0,5	0,98083734	0,975957557
$M2$	180	2	0,98680153	
$M3$	360	3	0,98409053	

Tabela 4.6: Resultados para a disponibilidade no caminho operacional

Parâmetro	**Disponibilidade do bloco funcionando (etapa 3)**	**Disponibilidade do bloco não funcionando (etapa 4)**	$I_{cs}(i;p)$ **(etapa 6)**
$M1$	$0,980837344$	0	$0,98574153$
$M2$	$0,98680153$	0	$0,99776599$
$M3$	$0,984090536$	0	$0,99229129$

Após o detalhamento das etapas que ocorreram para geração do ICD para o caminho operacional, apresentaremos nesse momento, como o processo ocorre para encontrar o ICD para o caminho não operacional (com falha) por meio das etapas descritas na Tabela 4.7.

As etapas que estão descritas na Tabela 4.7 para a identificação da estratégia ICD do caminho **não operacional**, são praticamente as mesmas do caminho operacional, no entanto, com algumas pequenas alterações em relação ao caminho operacional.

Tabela 4.7: Etapas do ICD para o caminho não operacional

Etapas	**Descrição**
1	Calcular a indisponibilidade individualmente de cada bloco (q_i) através da fórmula: $UA_{Mx} = 1 - A_{Mx}$, em que Mx representa os blocos M1, M2 e M3 e A_{Mx} representa a disponibilidade do bloco. O resultado desse cálculo encontra-se na Tabela 4.8, (vide página 58);
2	Calcular a indisponibilidade total do sistema (IDS) através da expressão $UA_s = 1 - AS$. O resultado desse cálculo encontra-se na Tabela 4.8, (vide página 58)
3	Calcular a disponibilidade do sistema quando cada bloco estiver funcionando, nesse caso, por exemplo, quando o bloco 1 está funcionando a disponibilidade é dada por meio da expressão $A = 1 \times AM_2 \times AM_3$. o resultado desse cálculo encontra-se na Tabela 4.6, (vide página 58);
4	Calcular a disponibilidade do sistema quando cada bloco não estiver funcionando. Nesse caso, por exemplo, a disponibilidade é dada pela equação $AS = AM_1 \times AM_2 \times AM_3$, como a disponibilidade individual do bloco é 0 (zero), quando ele não está operacional, desse modo a disponibilidade do sistema também é 0 (zero), em virtude de ter na equação uma das parcelas o valor 0 (zero). O resultado desse cálculo encontra-se na Tabela 4.6), (vide página 58);
5	Calcular a diferença entre a disponibilidade individual do bloco funcionando subtraindo a disponibilidade do bloco não funcionando, é obtido através da equação $(A(1_i,p) - A(0_i,p))$. O resultado desse cálculo encontra-se na Tabela 4.6 (vide página 58);
6	Calcular o índice de importância crítica para disponibilidade quando o sistema não está funcionando $I_{cf}(i;p)$ é obtido por meio dessa **etapa 6**. O resultado dessa equação $(A(1_i,p) - A(0_i,p))$ obtida na **etapa 5** divide pela indisponibilidade do sistema **etapa 2** multiplicado pelo indisponibilidade individual do bloco em falha **etapa 3**. O resultado desse cálculo encontra-se na Tabela 4.8, (vide página 58).

Tabela 4.8: Resultados para a estratégia ICD no caminho não operacional

Parâmetro	**Indisponibilidade individual de cada bloco (q_i) (etapa 1)**	$I_{cf}(i;p)$	**Indisponibilidade do sistema (etapa 2)**
$M1$	$0,01916265$	$0,01925846$	$0,02404244$
$M2$	$0,01319847$	$0,01334512$	
$M3$	$0,01590946$	$0,01604204$	

Após a execução da etapa (6) alcançamos o ICD no caminho não operacional. Na Tabela 4.8, o resultado apresentado para a coluna índice $I_{cf}(i;p)$ não está em ordem decrescente. Entretanto, corresponde ao índice de ICD. Nessa coluna, observamos que os blocos M1 e M3 são

os componentes de maior relevância para o sistema. Desse modo, é importante deligenciar ações de contingências com a finalidade de melhorar a disponibilidade e diminuir a probabilidade de risco do sistema. A Equação 4.25 que aplicamos para obtenção dos resultados para o caminho não operacional.

$$I_{cf}(i;p) = \frac{pi(A(1_i,p) - A(0_i,p))}{1 - A(p)} \tag{4.25}$$

A ICD é uma estratégia utilizada de acordo com os autores KUO; ZUO (2003b) e MARVIN RAUSAND (2003) para a métrica confiabilidade. Em virtude dessa estratégia poder ser utilizada para priorizar ações de manutenção em sistemas complexos, como os ambientes de computação em nuvem (MARVIN RAUSAND, 2003), e além de produzir como resultado o componente, e não os parâmetros individuais dele, ela também potencializa a análise do sistema por meio de dois caminhos: operacional e com falha. Essas foram as razões pelas quais decidimos realizar pequenas modificações na sua estrutura, com o propósito de conseguir avaliar a métrica disponibilidade do sistema, nessa pesquisa.

Os dois caminhos dessa estratégia foram implementados na ferramenta *Mercury*[1] (SILVA et al., 2015), de modo a automatizar o processo de construção do índice ICD. Ela retorna como resultado o componente mais relevante com as métricas de taxa de falha e reparo integrada ao mesmo. Isso é relevante em razão de proporcionar ao administrador do sistema uma visão de alto nível sobre o quão importante e quão insignificante os componentes na arquitetura do sistema podem representar.

4.4 Considerações Finais

Neste capítulo abordamos as estratégias de análise de sensibilidade *Discrete Averaged Sensitivity Index* (Índice Discreto Médio de Sensibilidade) (DASI) e Importância Crítica para a Disponibilidade (ICD), apresentando com detalhes, as fórmulas que as definem, a descrição, bem como exemplos de aplicação, além de particularidades apresentadas por cada uma.

No próximo capítulo são apresentados a descrição do método e o desenvolvimento da proposição para a aplicação de estratégias de análise de sensibilidade, em conjunto com a modelagem hierárquica para identificar os pontos que requerem melhoria na disponibilidade de uma infraestrutura de nuvem.

[1]Mercury que é uma ferramenta de modelagem e avaliação de sistemas desenvolvida pelo grupo de pesquisa MoDCS da UFPE. Disponivel em: http://www.modcs.org/?page$_id = 1397$

5

Metodologia

O objetivo deste capítulo é apresentar uma metodologia que baseia-se na proposição e adaptação de estratégias de análise de sensibilidade em conjunto com a modelagem hierárquica para identificar os pontos críticos de uma infraestrutura de nuvem que requerem melhoria na disponibilidade. As estratégias propostas são:Índice Discreto Médio de Sensibilidade, o qual denominamos de DASI e o Índice de Importância Crítica para a Disponibilidade, denominado de ICD.

5.1 Método

Nesta seção são apresentados a metodologia adotada para identificar os pontos que requerem melhoria na disponibilidade do sistema, a partir da aplicação de um conjunto das estratégias de análise de sensibilidade (EAS) propostas e existentes, integrada com a implementação de modelos hierárquicos e modelos para representação de mecanismos de redundância, num ambiente de infraestrutura de nuvem. Na Figura 5.1 apresentamos o diagrama de atividades da metodologia adotada. Esta é formada por 8 (oito) etapas: descrição do sistema, definição da métrica de interesse, modelagem em alto nível, modelagem com detalhes, avaliação dos modelos, aplicação das estratégias de análise de sensibilidade (EAS), cálculo do *ranking* de sensibilidade para cada técnica, identificação dos componentes relevantes e apresentação do *ranking* de sensibilidade.

Todas essas etapas se reúnem para que os serviços hospedados na nuvem possam ser desenvolvidos com a disponibilidade mais próxima possível dos 100% de funcionamento, significa um sistema sem interrupção, que é a meta a ser atingida. A seleção dos serviços candidatos para análise dessa metodologia foram por exemplo: os serviços de *Streaming* de Vídeo e *Mobile Backend as a Service* - MBaaS da plataforma *OpenMobster*, em razão de apresentarem um crescimento expressivo nos últimos tempos, e pela facilidade de implementação desses serviços num ambiente de teste desenvolvido em laboratório.

1º Etapa - Descrição do sistema: Esta etapa compreende a identificação das características do sistema de nuvem, bem como os componentes e interações. Essa descrição é importante

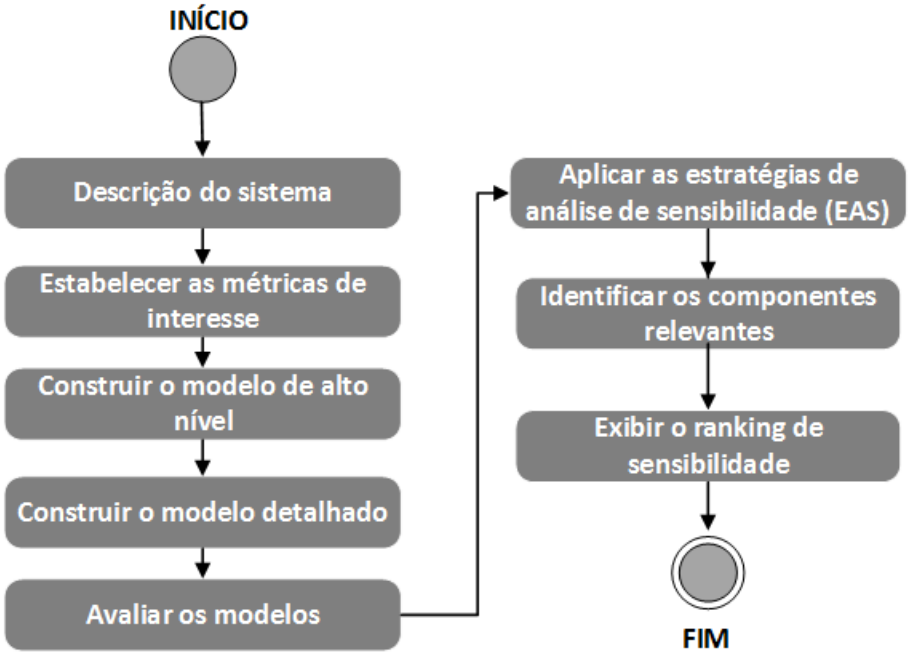


Figura 5.1: Metodologia de apoio à identificação de pontos críticos em infraestrutura de nuvem

para identificar o tipo de nuvem, a quantidade de máquinas virtuais (VM) e Nós, o *Storage* e o *Frontend* existentes na arquitetura que será avaliada, assim como verificar os detalhes de funcionamento e funcionalidades, identificando os componentes independentes e dependentes. Nesta etapa foi verificado o tipo de serviço que está implementado e as suas características de funcionamento do sistema. Nesse caso, dois serviços foram implementados para análise. sendo eles: o serviço de *Streaming* de Vídeo e o serviço baseado em MBaaS da plataforma *OpenMobster*, ambos implementados numa infraestrutura de nuvem.

2º Etapa - Estabelecimento das métricas de interesse: As medidas de interesse devem ser identificadas antes da concepção do modelo. Essa métrica é a informação essencial que será obtida por meio do modelo para diagnosticar o comportamento da dependabilidade ou do desempenho do sistema.

Em muitos casos, uma vez que os usuários de sistemas baseados em nuvem podem estar em qualquer lugar e o sistema é centralizado, métricas como *throughput* e utilização da CPU podem não corresponder exctamente à qualidade de serviço percebidos pelos usuários finais. Por outro lado, ao lidar diretamente com o IaaS, algumas métricas centradas no sistema podem ser muito importantes porque os usuários finais são de fato administradores de sistemas que podem querer saber o tempo de resposta, vazão, *downtime*[1], disponibilidade, uso da memória e CPU, a utilização das VMs. Neste trabalho enfatizamos a métrica disponibilidade e *downtime*.

3º Etapa - Construção do modelo de alto nível: A partir de uma infraestrutura de computação em nuvem, precisamos obter uma visão geral do desempenho do sistema ou confi-

[1] Período de tempo em que o sistema está indisponível

abilidade que nos permite criar um modelo de alto nível. Este é o principal modelo que pode descrever a interconexão de subsistemas (grupos de componentes relacionados) no ambiente IaaS, as atividades globais de processamento de solicitações de usuários ou a relação de dependência global. Considerando que é uma infraestrutura de nuvem, precisamos obter uma visão ampla de funcionamento de modo que seja factível a elaboração do modelo analítico que a represente. Esse modelo deve retratar a relação entre a arquitetura de nuvem e o serviço provido, além das atividades gerais para processamento de solicitações dos clientes, bem como a relação entre os componentes. Modelos como RBD por exemplo podem ser utilizados para a representação dessa infraestrutura.

Nesta etapa verificamos algumas condições para construção do modelo de alto nível que são: i) conhecimento prévio sobre o sistema em nuvem a ser modelado e os possíveis formalismos a serem utilizados, ii) verificação do tipo pretendido de análise (por exemplo, disponibilidade, confiabilidade, desempenho); lista de principais componentes ou subsistemas, valores de parâmetros e descrição da dependência ou interligação entre eles, iii) a escolha do formalismo de modelagem, criação do modelo de alto nível, esse é parametrizado e pronto para ser refinado em submodelos, também chamados de modelos detalhados.

4º Etapa - Construção dos modelos detalhados: Os modelos detalhados são aqueles que apresentam um nível de hierarquia diferente, podemos chamá-los de submodelos. Este descreve o comportamento detalhado de cada subsistema, além de poder lidar com os mecanismos de *hardware* e *software* especializados que não estão presentes no modelo principal, ou ainda fornecer avaliação refinada de componentes como: VM, *Node*, Serviço, Aplicativo ou até mesmo reunir o comportamento de mais de um elemento. Vale ressaltar que esta proposta metodológica não restringe a quantidade de níveis nos modelos hierárquicos.

Para os modelos detalhados existerem, é condição que exista um modelo principal e que algum subsistema do modelo principal precise ser detalhado. Um submodelo pode compreender outros modelos de nível inferior de modo a reduzir a complexidade da modelagem, por exemplo. O analista do sistema deve ser cauteloso quanto a possíveis perdas de precisão devido a níveis excessivos no modelo. Podem ser escolhidos formalismos distintos para cada submodelo, dependendo do conhecimento do modelador e do formalismo que melhor descreva esse subsistema específico. Desse modo, os parâmetros do modelo detalhado, como MTTF e MTTR, servem de dados de entrada para o modelo principal, o qual ele representa.

Nesta etapa consideramos algumas condições para construção do modelo de detalhado: i) a existência de um modelo principal (de nível superior) com alguns subsistemas a serem refinados, ii) lista de subsistemas que podem ser refinados; Descrição dos componentes, parâmetros, valores e funcionamento interno de cada subsistema, iii) escolha de formalismos de modelagem; Criação de submodelos e definição de como eles estão conectados ao modelo principal, iv) os submodelos são parametrizados e conectados ao modelo principal.

5º Etapa - A avaliação de modelos: A avaliação dos modelos hierárquicos é a próxima etapa na presente metodologia. Se o modelo principal possuir um modelo detalhado associado a

ele, nesse caso, inicialmente, resolve-se o modelo detalhado e os parâmetros de saída obtidos são utilizados como parâmetros de entrada no modelo principal. Se o sistema possui apenas modelo principal, sem modelos detalhados associados, nesse caso são atribuídos os parâmetros de entrada desse modelo e obtido o resultado para a métrica pretendida.

6º Etapa - Aplicação das estrátégias de análise de sensibilidade - EAS: Nessa etapa, aplicamos as EAS nos modelos hierárquicos, para encontrar os índices de sensibilidade. A variação um por um, *design of experiments* - DoE, diferença percentual, e por último as estratégias propostas chamadas de *Discrete Averaged Sensitivity Index* (Índice Discreto Médio de Sensibilidade) (DASI) e Importância Crítica para a Disponibilidade (ICD) são métodos possíveis para essa tarefa. Cada estratégia terá a sua particularidade de funcionalidade e fórmula a ser aplicada nos modelos.

A estratégia **DASI** propõe aplicar a derivada parcial em mais de um parâmetro de entrada, isso representa uma confiabilidade na informação a ser obtida, além de sairmos da dependência de um único parâmetro. Por outro lado, a estratégia **ICD** estabelece um *ranking* a partir da métrica disponibilidade, quando os componentes presentes na arquitetura fazem parte de um caminho operacional e do caminho não operacional, com falha. Essa última estratégia captura os componentes com falha, podendo ser usada para priorizar ações de manutenção em sistemas complexos (MARVIN RAUSAND, 2003).

É importante ressaltar que as estratégias DoE, diferença percentual e ICD capturam comportamento estático do sistema, com exceção da estratégia DASI e variação um por um que consegue capturar o comportamento dinâmico do sistema, em virtude da possibilidade de variação dos seus parâmetros de entrada.

À medida em que aplicamos o procedimento para o cálculo do índice de sensibilidade em cada estratégia, obtemos como resultado para as mesmas um conjunto de valores. Esses valores podemos ordenar em ordem decrescente, e quando necessário podemos normalizá-los. Após esse procedimento, teremos naturalmente os dados na forma de um *ranking*. Registramos que a estratégia de **variação um por um** não apresenta um *ranking* como resultado da análise de sensibilidade, e que a estratégia ICD apresenta como resultado da sua análise um *ranking* indicando os componentes mais importantes, não apresentando os seus respectivos parâmetros.

7º Etapa - Identificar os componentes relevantes para as EAS: Nessa etapa, identificamos quais são os componentes que se destacam nas estratégias, a partir da comparação da posição que cada parâmetro ocupa no *ranking* das estratégias. No *ranking*, os componentes estão presentes de forma desencontrada, desse modo aplicamos um método para verificar a contribuição de cada parâmetro no *ranking*. Na Equação 5.1 apresentamos como esse método foi formalizado.

$$F(i) = \sum_{i}^{n} \prod_{j}^{M} (C_j \times W) \tag{5.1}$$

Onde W, é o peso expresso por:

$$W_{(i,j)} = \frac{1}{P_{(i,j)}}$$

C_j representa os critérios que correspondem à estratégia utilizada. De acordo com cada posição (P) do componente no *ranking* do critério (C_j), é atribuído um peso (W) a cada componente, de acordo com o sua posição no *ranking*. Na Equação 5.2 apresentamos como o peso é calculado.

Na Figura 5.2 apresentamos um exemplo contendo três *rankings* de estratégias de sensibilidade diferentes, denominadas de C1, C2 , C3. Para essas estratégias (C), atribuímos pesos de acordo com a sua posição no *ranking* de cada critério.

Componente	C1	C2	C3	P(i,Cj)	W	
P1	0.9	0.7	O.75	P(1,C1)=2	P(1,C2)=3	P(1,C3)=2
P2	0.8	0.75	0.85	P(2,C1)=4	P(2,C2)=1	P(2,C3)=1
P3	0.95	0.6	0.7	P(3,C1)=1	P(3,C2)=4	P(3,C3)=3
P4	0.85	0.75	0.7	P(4,C1)=3	P(4,C1)=1	P(4,C3)=3

Cálculo do Ranking

$F(1) = 0.9 \times \frac{1}{2} + 0.7 \times \frac{1}{3} + 0.75 \times \frac{1}{2} = 1.058$

$F(2) = 0.8 \times \frac{1}{4} + 0.75 \times \frac{1}{1} + 0.85 \times \frac{1}{1} = 1.8$

$F(3) = 0.95 \times \frac{1}{2} + 0.6 \times \frac{1}{4} + 0.7 \times \frac{1}{3} = 1.33$

$F(4) = 0.85 \times \frac{1}{3} + 0.75 \times \frac{1}{1} + 0.7 \times \frac{1}{3} = 1.26$

Ranking dos componentes

P2
P3
P4
P1

Figura 5.2: Exemplo do cálculo de *ranking* de sensibilidade

Na Figura 5.2, por exemplo, o parâmetro do componente P1, tem o valor de 0,9 na estratégia ou critério C1, esse valor em relação aos demais valores do *ranking* ocupa a segunda posição nesse critério. O componente P2, tem o valor de 0,8 no critério C1, esse valor em relação aos demais valores do *ranking* ocupa a quarta posição nesse critério.

Aplicando a Equação 5.1, em que para cada parâmetro será atribuído um peso em função da posição que o mesmo ocupa no *ranking* de sensibilidade para cada critério ou estratégia. O resultado dessa avaliação produzirá um *ranking* em que os resultados expressam os componentes relevantes às estratégias, considerando a posição que eles ocupam em cada uma. O resultado desse exemplo indica, na ordem de importância, dos componentes P2, P3, P4 e P1. Esse resultado identifica com alta confiabilidade os componentes significativos do sistema, uma vez que o seu resultado é um índice produto da avaliação realizada por mais de uma estratégia.

É importante destacar que as estratégias de **variação um por um** e **ICD** não fazem parte dessa comparação em que a Equação 5.1 foi aplicada. Isso ocorre, em razão da estratégia variação um por um, não apresentar como resultado um *ranking* e a estratégia ICD apresentar um *ranking* em função dos componentes, e não em função dos parâmetros.

8º Etapa - Exibir *ranking* de sensibilidade Esta etapa apresenta o resultado da etapa

anterior, que consiste em um *ranking*, exibindo a ordem de importância de cada parâmetro. Esse resultado está disponível para ser analisado e interpretado sobre que ações que podem ser tomadas em relação a eles, com o objetivo de melhorar a disponibilidade dos sistemas em que estão inseridos. Uma das ações de melhorias pode ser a aplicação dos modelos para representação dos mecanismos de redundância, sendo eles: *warm standby*, *cold standby* e ativo-ativo.

As etapas metodológicas compõem a proposição de um *Guideline*, para identificar os pontos de melhoria da disponibiliddae de uma infraestrutura de nuvem, com um serviço hospedado na mesma. Essas etapas foram aplicadas e validadas no capítulo 7 páginas 82 a 126 por meio de diferentes estudos de casos.

5.2 Informações Adicionais

Nesta seção registraremos algumas informações adicionais a respeito da metodologia utilizada nesta Tese, bem como a delimitação do uso dos modelos, das EAS e a capacidade do sistema modelado.

- **Usuários**: Esta metodologia foi elaborada para os usuários que são administradores de ambientes de infraestrutura de nuvem que trabalham no planejamento, gerenciamento e monitoramento desses. Além disso, para usuários que buscam alternativas para identificar os pontos críticos, ou seja, que requerem melhoria em relação a disponibilidade dessas infraestruturas, e que ainda buscam a proposição de soluções, como por exemplo, as relacionadas aos mecanismos de redundância, para garantir o nível de acordo de serviço estabelecido entre o provedor da infraestrura e do serviço com o usuário final.

- **Modelos e EAS**: Os modelos foram elaborados para capturar o comportamento estático do sistema, considerando uma configuração básica e uma configuração com implementação de redundância. Com relação as EAS é importante ressaltar que as estratégias DoE, diferença percentual e ICD capturam comportamento estático do sistema, com exceção da estratégia DASI e variação um por um que consegue capturar o comportamento dinâmico do sistema, em virtude da possibilidade de variação dos seus parâmetros de entrada.

- **Capacidade do sistema**: O serviço de vídeo *streaming* foi considerado em quatro tipos diferentes de *VMs* do *Eucalyptus*, que correspondem as instâncias padrões do *Amazon EC2* EC2 (2014). As especificações dessas *VMs*, bem como a quantidade de usuários que elas suportam, estão descritas na Tabela 5.1.

Tabela 5.1: Especificações das Máquinas Virtuais.

Tipo	CPU	Memória (MB)	HD (GB)	Nº de usuários suportados
m1.Small	1	256	5	12
m1.Medium	1	512	10	32
m1.Large	2	512	10	32
m1.Xlarge	2	1024	10	74

5.3 Considerações Finais

Neste Capítulo apresentamos a metodologia adotada para identificar os pontos de melhoria na disponibilidade de uma infraestrutura de nuvem, detalhando suas etapas e expondo os principais resultados das mesmas. Além disso, apresentamos algumas informações adicionais sobre este método tais como: os possíveis usuários desta metodologia; os modelos e as EAS utilizadas nesta pesquisa, bem como a capacidade do sistema considerando as máquinas virtuais instanciadas e a capacidade de usuário que cada uma suporta.

No próximo capítulo, apresentaremos os modelos para representação dos mecanismos de redundância sendo eles: *warm standby*, *cold standby* e ativo-ativo, bem como o processo de validação desses modelos.

6

Modelos para Representação dos Mecanismos de Redundância

Nesta seção, são apresentados os modelos para a representação dos mecanismos de redundância sendo eles: *cold standby*, ativo-ativo e *warm standby*, bem como o processo de validação desses modelos. Esses modelos foram elaborados para serem aplicados à infraestrutura de nuvem com o propósito de prover melhorias na disponibilidade do sistema.

6.1 Introdução

Os mecanismos de redundância podem ser classificados como ativo-ativo e ativo-*standby*. Os mecanismos do tipo ativo-ativo são usados quando os componentes primários e secundários compartilham a carga de trabalho do sistema. Quando quaisquer um desses componentes falham, outros componentes terão a responsabilidade de atender às demandas dos usuários do sistema. Esses mecanismos de redundância podem ser classificados como N + K, em que K componentes secundários idênticos aos N componentes primários são necessários para compartilhar o sistema de carga de trabalho. Na configuração N + 1, um componente secundário idêntico ao N componente primário é necessário para o compartilhamento da carga de trabalho do sistema (BAUER; ADAMS; EUSTACE, 2011).

Por outro lado, os mecanismos de redundância do tipo ativo-*standby* são empregados quando os componentes primários atendem às solicitações dos usuários do sistema e componentes secundários estão em espera. Quando o componente primário falhar, os secundários serão responsáveis por servir aos pedidos dos usuários do sistema. Os mecanismos de redundância ativo-*standby* podem ser categorizados como: *cold standby*, *warm standby* e *hot standby*, (BAUER; ADAMS; EUSTACE, 2011).

Os modelos ativo-ativo, *cold* e *warm standby* que são apresentados, nesta seção, foram inicialmente idealizados para o serviço de *streaming* de vídeo, sendo executado numa plataforma de nuvem *Eucalyptus*. Em consequência das observações realizadas na avaliação inicial dessa pesquisa, identificamos que esses modelos podem ser aplicados em um outro serviço, mediante

algumas alterações e adequações nos mesmos. Desse modo, além de utilizar o serviço de *Streaming* de Vídeo, o serviço *Mobile Backend as a Service - MBaaS* na plataforma *OpenMobster*, hospedado igualmente no ambiente de nuvem *Eucalyptus* foi utilizado. As escolhas desses dois serviços deram-se ao fato de ambos serem amplamente utilizados e do crescimento de tráfego de dados referente a eles.

Na próxima seção, apresentamos os modelos elaborados para a representação dos mecanismos de redundância *cold standby, warm standby* e ativo-ativo.

6.2 Modelo *Cold Standby*

Inicialmente utilizaremos um modelo RBD de alto nível (ver Figura 6.1) para representar a disponibilidade do serviço de *Streaming* de Vídeo hospedado numa infraestrutura de nuvem. A partir dessa perspectiva, é implementada a redundância *cold standby*. O bloco RBD para o módulo SSMARKOV representado na Figura 6.1 retrata uma arquitetura que possui uma máquina para o *Frontend* e duas máquinas para os *Nodes*, funcionando no modo *cold standby*. Além disso, a máquina, que é o *Frontend*, é a responsável pelo controle dos *Nodes*. Essa arquitetura é representada pelo bloco RBD para o módulo SSMARKOV. O bloco RBD para o módulo Volume, em série com o SSMARKOV, representa o dispositivo de armazenamento dos vídeos.

Figura 6.1: Modelo RBD para sistema *streaming* de vídeo

A disponibilidade do sistema representado pelo modelo da Figura 6.1 é obtido através da Equação 6.1.

$$A_s = A_{SSMARKOV} \times A_{VOLUME} \tag{6.1}$$

Em virtude de algumas particularidades, que apenas o modelo de redundância possui, o modelo RBD não consegue representar esses detalhes, por isso é necessário refinar o bloco RBD para o módulo SSMARKOV para uma cadeia de *Markov*. As particularidades associadas ao modelo *cold standby* estão relacionadas ao tempo de ativação da máquina que está *offline*, ou seja, desligada aguardando uma falha da máquina que está funcionando, a fim de que possa entrar em operação. Além disso, a necessidade de representar o *status* dos componentes, como por exemplo a condição *UP* representa o componente em funcionamento, a condição D (desligado) representa o componente desligado ou falha dos componentes *Frontend*, *Node* principal e secundário, *offiline* representa *Node* secundário *offiline*. Desse modo, o bloco RBD SSMARKOV foi representado por meio da cadeia de *Markov* na Figura 6.2. Esse modelo é composto por 8 lugares sendo eles: UOU, UOD, DOU, DOD, DUU, DUD, DDU e DDD. Os estados em cinza

são os que correspondem ao sistema no modulo operacional enquanto os estados transparentes correspondem ao sistema no modo não operacional.

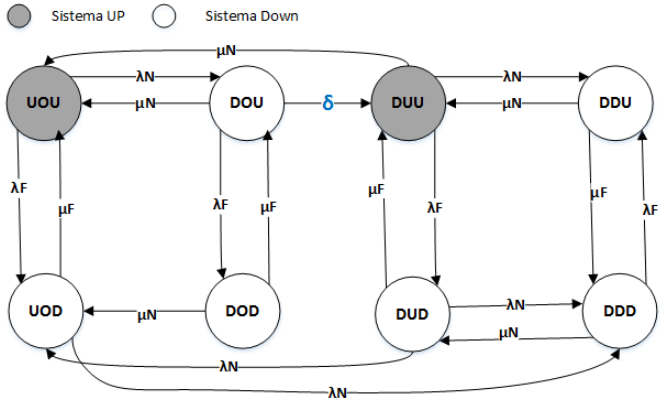


Figura 6.2: Modelo CTMC *cold standby*

A notação para os estados é baseada na condição atual de cada máquina. As três letras representam *Node* principal na situação *UP* ou Desligado, *Node* secundário na condição *Offline*, *UP* ou Desligado é o estado inicial do *Frontend* na situação Desligado ou *UP* respectivamente. Na plataforma *Eucalyptus* a máquina correspondente ao *Frontend* detém o controle das máquinas onde estão alocados os *Nodes*, no momento que essa máquina falha não temos o controle do sistema levando-o para o estado de indisponibilidade do serviço.

No estado UOU o sistema está com o *Node* principal e o *Frontend* na condição *Up* representando o sistema funcionando, o *Node* secundário está na condição de *offline* (desligado). A partir desse estado, é possível alcançar os estados DOU e o UOD. Atingindo o estado UOD encontramos o *Node* principal !*UP*, o *Node* secundário *offline* e o *Frontend* no estado de falha por meio da taxa λ_f, sendo possível realizar o seu reparo por meio da taxa μ_f. Quando o estado UOU atinge o estado DOU, percebe-se que o *Node* principal está *down* por meio da taxa de falha λ_N, sendo possível repará-lo por meio da taxa de reparo μ_N, o *node* secundário está desligado e o *Frontend* está *Up*. Através do estado UOD podemos atingir o estado DDD a partir da falha do *Node* principal.

A partir do estado DOU, três caminhos são possíveis UOU, DOD, DUU. Ao atingir o estado UOU ocorre o reparo do *Node* principal através da taxa de reparo μ_N. Chegando no estado DOD, ocorre a falha no *Frontend*, deste estado posso atingir o estado UOD através da recuperação do *Node* principal. Do estado DOU para o estado DUU ocorre a ativação do *Node* secundário. Nesse estado o *Node* secundário está no modo *cold standby* (desligado) e para ligá-lo levará um certo tempo que chamamos de 'δ' para que a máquina entre em operação e assuma a carga de trabalho.

No estado DUU o sistema está operacional com o *Node* secundário *Up* e o *Node* principal *down* e o *Frontend* no estado operacional, a partir desse estado é possível atingir os estados DUD e DDU e também UOU. Indo para o estado DDU ocorre uma falha no *Node* secundário, nesse caso, teremos os dois Nodes indisponívies e o *Frontend* operacional, no entanto o serviço não

estará operacional. Atingindo o estado DUD *Node* principal está *down* e o *Node* secundário *UP*, porém o *Frontend down* levando o serviço para a indisponibilidade. Ao atingir o estado DDD temos os três componentes *down*, os dois *Nodes* (principal e redundante) e o *Frontend*. A Equação da disponibilidade para este modelo da Figura 6.2 é obtido através da Equação 6.2:

$$A = P(UOU) + P(DUU) \tag{6.2}$$

O submodelo que representa o componente *Frontend* é representado pelo modelo RBD ilustrado na Figura 6.3. Este subsistema consiste de *Hardware* (HW), Sistema Operacional (SO), e os seguintes componentes do *Eucalyptus*: CLC (Controlador da Nuvem), CC (Controlador do *Cluster*), SC (Controlador de Armazenamento) e *Walrus* (DANTAS et al., 2012).

Figura 6.3: Modelo RBD para o *Frontend*

Esse modelo foi utilizado para calcular o tempo médio de falha (MTTF) e o tempo médio de reparo (MTTR) do componente *Frontend*. O modo operacional desse modelo é apresentado por meio da Equação 6.3.

$$MO_{Frontend} = HW \wedge SO \wedge CLC \wedge CC \wedge SC \wedge Walrus \tag{6.3}$$

Na Figura 6.4 representa o modelo RBD do componente *Node*. Além do *Hardware* (HW) e do Sistema Operacional (SO) também presentes no *Frontend*, cada nó requer um *hypervisor* (KVM) e o Controlador do *Node* (NC) do *Eucalyptus* (DANTAS et al., 2012), a VM (Máquina Virtual) e as aplicações VLC e *Apache*, estes três últimos componentes representam o serviço.

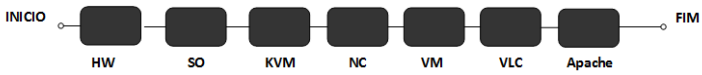


Figura 6.4: Modelo RBD para o *Node*

Esse modelo foi utilizado para calcular o tempo médio de falha (MTTF) e o tempo médio de reparo (MTTR) do componente *Node*. O modo operacional é apresentado por meio da Equação 6.4.

$$MO_{Node} = HW \wedge SO \wedge KVM \wedge NC \wedge VM \wedge VLC \wedge Apache \tag{6.4}$$

Para facilitar a compreensão, a Figura 6.5 apresenta a consolidação de todos os modelos que compõem o modelo *cold standby*. Inicialmente, o modelo de alto nível em RBD formado pelos blocos SSMARKOV e Volume, na sequência temos a cadeia de *Markov* apresentada anteriormente, com os tempos de falha e de reparo dos dois submodelos em RBD, sendo eles o *Frontend* e o *Node*.

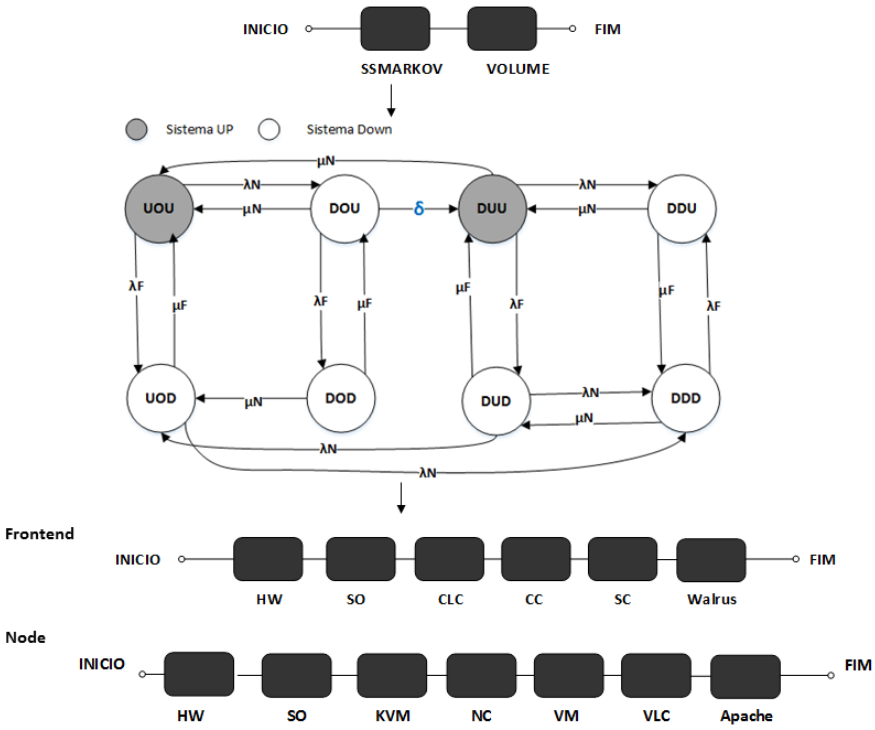


Figura 6.5: Modelo *cold standby* com os submodelos

A Tabela 6.1 contém os parâmetros utilizados no modelo *cold standby* com os submodelos e as respectivas descrições. É importante destacar que as taxas de falha e de reparo do bloco RBD para o módulo SSMARKOV é o resultado do submodelo que corresponde aos módulos *Frontend* e o *Node*.

Esse modelo utiliza cadeia de *Markov* para representar o mecanismo de redundância *cold standby* abordando as dependências existentes entre o *Frontend* e os dois *Nodes*, no entanto os autores (GUIMARÃES; MACIEL; MATIAS, 2013), (CALLOU et al., 2014), (SOUSA et al., 2015), utilizam SPN para representar esse tipo de mecanismo. Por outro lado, os autores em (HU; XIE, 2008), (LEVITIN; XING; DAI, 2015) não utilizam modelos para representação de mecanismos *cold standby*.

6.3 Modelo Ativo-Ativo

O modelo de redundância ativo-ativo proposto representa o serviço de *Streaming* de Vídeo na plataforma *Eucalyptus,* por meio de redes de Petri estocásticas (SPN). Esse modelo foi baseado em uma cadeia de *Markov* apresentada em (BAUER; ADAMS; EUSTACE, 2011).

Na Figura 6.6 apresenta o modelo SPN adotado para estimar a disponibilidade de um sistema com redundância ativo-ativo. Nessa configuração, temos duas máquinas (node 1 e node 2) funcionando com o serviço de *Streaming* de Vídeo ativo em ambas simultaneamente, trabalhando em paralelo e de forma independente uma da outra. Quando o serviço de vídeo

Tabela 6.1: Parâmetros de entrada para modelo e submodelos do *cold standby*

Módulo	Componente	Descrição
Frontend	λ_{HW}	Tempo médio de falha do HW
	μ_{HW}	Tempo médio de reparo do HW
	λ_{SO}	Tempo médio de falha do SO
	μ_{SO}	Tempo médio de reparo do SO
	λ_{CLC}	Tempo médio de falha do CLC
	μ_{CLC}	Tempo médio de reparo do CLC
	λ_{CC}	Tempo médio de falha do CC
	μ_{CC}	Tempo médio de reparo do CC
	λ_{SC}	Tempo médio de falha do SC
	μ_{SC}	Tempo médio de reparo do SC
	λ_{Walrus}	Tempo médio de falha do *Walrus*
	μ_{Walrus}	Tempo médio de reparo do *Walrus*
Node	λ_{HW}	Tempo médio de falha do HW
	μ_{HW}	Tempo médio de reparo do HW
	λ_{KVM}	Tempo médio de falha do KVM
	μ_{KVM}	Tempo médio de reparo do KVM
	λ_{NC}	Tempo médio de falha do NC
	μ_{NC}	Tempo médio de reparo do NC
	λ_{VM}	Tempo médio de falha do VM
	μ_{VM}	Tempo médio de reparo do VM
	λ_{VLC}	Tempo médio de falha do VLC
	μ_{VLC}	Tempo médio de reparo do VLC
	λ_{Apache}	Tempo médio de falha do *Apache*
	μ_{Apache}	Tempo médio de reparo do *Apache*
SSMARKOV	λ_N	Tempo médio de falha do módulo *Node*
	μ_N	Tempo médio de reparo do módulo *Node*
	λ_f	Tempo médio de falha do módulo *Frontend*
	μ_f	Tempo médio de reparo do módulo *Frontend*
	δ	Tempo médio de ativação do *Node* Secundário
Volume	λ_{Volume}	Tempo médio de falha do Volume
	μ_{Volume}	Tempo médio de reparo do Volume

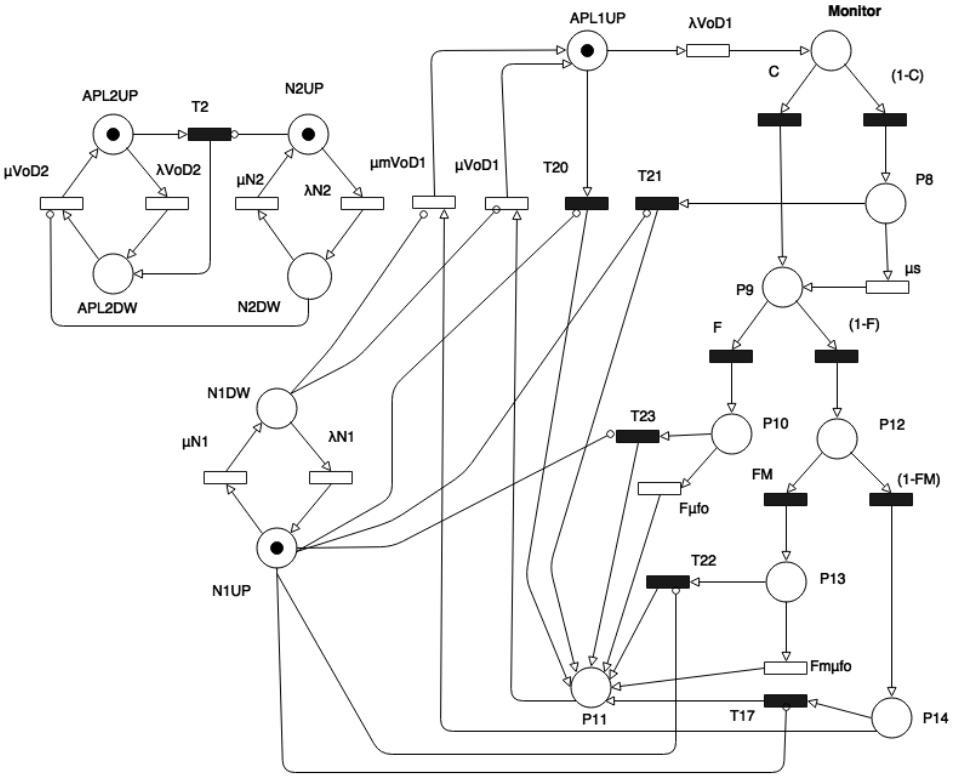


Figura 6.6: Modelo SPN Ativo-Ativo

falhar em uma das máquinas, no caso a máquina 1; a máquina 2 assume a carga de trabalho da máquina não operacional.

As marcações dos lugares APL2UP, APL1UP denotam os estados operacionais. O lugar APL2DW corresponde a um estado em que a aplicação do serviço de *Streaming* de Video está com falha. Esses *Nodes* funcionam como um *backup* ativo do sistema para que, caso ocorra uma falha em um dos *Nodes*,o *Node* que estiver operacional possa assumir as atividades do *node* que apresentou problema.

O lugar APL2UP, correponde à aplicação 2, quando essa aplicação falha, ocorre um evento relacionado à transição λ_{VoD2}, que alcança o lugar APL2DW, que significa aplicação não operacional. Após o evento de falha da aplicação 2 podemos iniciar o processo de recuperação da mesma através da transição μ_{VoD2}. No entanto, para existir a condição de recuperação da aplicação, é preciso que o *Node* 2 esteja operacional, ou seja, o lugar N2UP está com uma marcação. Caso ocorra falha no *Node* 2, através do evento relacionado à transição λ_{N2}, a aplicação para de funcionar automaticamente sem tempo associado à sua paralisação, levando ao lugar N2DW. A recuperação do *node* 2 ocorre através do evento relacionado à transição μ_{N2}, levando-o novamente ao estado operacional N2UP. A transição T2 controla o evento de falha da aplicação para que não ocasione falha no *Node*.

No *Node* N1, funciona a aplicação 1, quando a aplicação 1 para de funcionar, a carga de trabalho gerada por essa aplicação é absorvida pelo *Node* 2. A falha da aplicação é representada pelo evento da transição λ_{VoD1} e pode tomar dois caminhos, sendo eles um caminho coberto e não coberto. O caminho coberto significa que o evento ocorreu com sucesso e que o monitoramento do serviço é feito de forma automática, esse evento é representado pela transição C. O caminho não coberto significa que não houve sucesso na cobertura e que o monitoramento aconteceu de forma manual, sendo representado pela transição $1-C$.

A análise do modelo seguindo o caminho do monitoramento automátco, inicia-se quando a aplicação APL1UP falha, é representada pelo evento λ_{VoD1} que alcança o lugar **Monitor**. A transição C representa que o evento ocorre com 90% de sucesso e nesse momento, isso quer dizer o processo de *failover* automático é iniciado com ativação da transição F, o percentual de sucesso do *failover* é de 99% de sucesso. Esse processo assume que 50% do tráfego antes da falha já foi processado pela máquina onde ocorreu a falha e que os outros 50% serão direcionados para a máquina que está operacional.

Isso ocorre no primeiro momento da falha e o lugar P10, representa esse evento transitório. No entanto, após algum tempo, 100% do tráfego da máquina com falha, no caso a aplicação 1, será direcionado para a máquina que está operacional. Este evento será representado através da ativação da transição $F\mu_{fo}$ que é o tempo de detecção do *failover* automático, atigindo portanto, o lugar P11 com uma das máquinas funcionando com 100% da carga até que recuperemos a máquina com falha na aplicação. A recuperação da aplicação com falha é realizada através da ativação da transição μ_{VoD1} chegando ao lugar APL1UP restabelecendo a aplicação 1.

Existe a possibilidade do *failover* automático não funcionar em 1% (BAUER; ADAMS;

EUSTACE, 2011) dos casos. Nesse caso, o chaveamento da carga de trabalho ocorrerá de forma manual e a transição $1-F$ é ativada chegando no lugar P12. Nesse momento atingimos a transição FM indicando que o processo iniciará de forma manual.

Ao atingir o lugar P13, teremos um estado momentâneo cuja carga de trabalho está sendo migrada de uma máquina para outra, ao acionarmos a transição $Fm\mu_{fo}$ que representa o tempo de ativação manual do processo de *failover*. Ao chegar no lugar P11 todo o tráfego já terá sido absorvido pela outra máquina e permanecerá nesse lugar até que a aplicação seja restabelecida. O restabelecimento da aplicação acontecerá com ativação da transição μ_{VoD1}.

Assim como o processo de *failover* automático, o processo de *failover* manual pode falhar em 1% (BAUER; ADAMS; EUSTACE, 2011). No modelo quando esse evento ocorre temos a ativação da transição $1-FM$. Quando isso ocorre, o lugar P14 é atingido e na sequência a transição μ_{mVoD1} é acionada e a aplicação será recuperada de forma manual levando um tempo maior do que no processo automático. As transições T17,T21,T22 e T23 são ativadas para garantir o funcionamento do *Node* 1. Em estado operacional, quando estão acontecendo os processos de monitoramento cobertos ou não, *failover* automático ou manual, manutenção automática ou manual.

A Equação da disponibilidade para esse o modelo da Figura 6.6 é obtido através da Equação 6.5:

$$A = P(APL2UP = 1)OR(APL1UP = 1) \tag{6.5}$$

Para facilitar a compreensão separamos as transição imediatas e as transições temporizadas, nas Tabelas 6.2 e 6.3 respectivamente.

Tabela 6.2: Atributos das Transições imediatas

Transição	Tipo	Descrição
$T2$	-	é responsável pelo funcionamento do node 2
C	-	probabilidade de sucesso de cobertura
$1-C$	-	probabilidade de insucesso de cobertura
F	-	probabilidade de sucesso do *failover* automático
$1-F$	-	probabilidade de insucesso do *failover* automático
FM	-	probabilidade de sucesso do *failover* manual
$1-FM$	-	probabilidade de insucesso do *failover* manual
$T17$	-	é responsável pelo funcionamento do node 1
$T20$	-	é responsável pelo funcionamento do *Node* 1
$T21$	-	é responsável pelo funcionamento do *Node* 1
$T22$	-	é responsável pelo funcionamento do *Node* 1
$T23$	-	é responsável pelo funcionamento do *Node* 1
$T2$	-	é responsável pelo funcionamento do *Node* 2

Esse modelo configurou uma rede de Petri estocástica para mostrar o funcionamento do mecanismo de redundância ativo-ativo, monitorando o sucesso do *failover* automático e o sucesso do processo manual. Além disso, apresenta a recuperação do sistema de forma manual, caso o sistema automático não funcione corretamente.

Os mecanismos presentes nas redes de Petri permitem a especificação, análise e verifica-

Tabela 6.3: Atributos das Transições temporizadas

Transição	Tipo	Descrição
μ_{VoD2}	exponencial	tempo de reparo do serviço
λ_{VoD2}	exponencial	tempo de falha do *Node*
μ_{N2}	exponencial	tempo de reparo do*Node*
λ_{N2}	exponencial	tempo de falha do *Node*
μ_{N1}	exponencial	tempo de reparo do *Node*
λ_{N1}	exponencial	tempo de falha do *Node*
μ_{mVoD1}	exponencial	tempo de reparo manual do serviço
μ_{VoD1}	exponencial	tempo de reparo do serviço
λ_{VoD1}	exponencial	tempo de falha do serviço
μs	exponencial	tempo de falha na cobertura
$F\mu fo$	exponencial	tempo de ativação do *failover*
$Fm\mu fo$	exponencial	tempo de ativação do *failover* manual

ção de propriedades e a corretude do sistemas modelados (ALBUQUERQUE JÚNIOR, 2013). A análise das propiedades para esse modelo ativo-ativo em rede de Petri foi realizada por meio da ferramenta INA (STARKE; ROCH, 1992). Desse modo, foi possível identificar que essa rede é pura, é limitada e estruturalmente limitada, é subconservativa, ordinária e homogênea, além de está livre de pontos de conflitos (ver Figura 6.3). Em virtude do modelo concebido possuir a propriedade estruturalmente limitada (MURATA, 1989), isso significa que os seus espaços de estados são finitos, desse modo, uma cadeia de *MarKov* pode ser gerada.

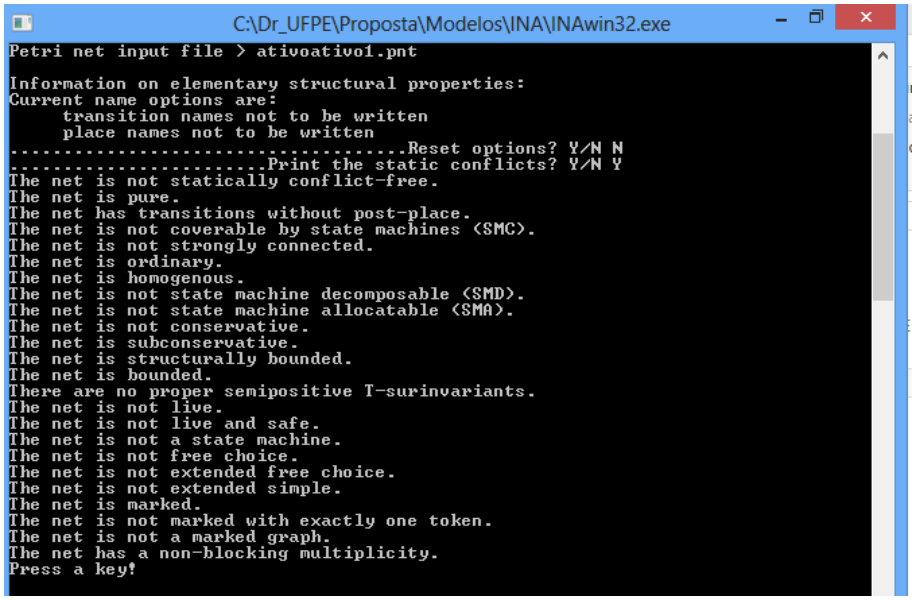


Figura 6.7: Propriedades da rede de Petri

Por outro lado, essa SPN permite realizar um estudo de disponibilidade orientada à capacidade, a qual denominamos de COA. A disponibilidade orientada à capacidade é uma das técnicas utilizadas para alcançar a disponibilidade desejada, sendo possível calcular métricas para obtenção de resultados e tomadas de decisões adequadas (DANTAS et al., 2012). A COA é uma métrica que indica não apenas se o sistema está operacional ou não, mas também a capacidade de atendimento do sistema. É possível calcular a disponibilidade orientada à capacidade para esse modelo, por meio da Equação 6.6:

$$COA = \frac{P(\#APL2UP = 1 \quad AND \quad \#APL1UP = 1) \times 2 + \alpha + \theta}{2} \tag{6.6}$$

Em que:
$\alpha = P(\#APL2UP = 1 \quad AND \quad \#APLL1UP = 0)$;
$\theta = P(\#APL2UP = 0 \quad AND \quad \#APL1UP = 1)$.

A Equação 6.6 destina-se a capturar o poder computacional, quando os dois servidores estão trabalhando simultaneamente e quando ocorre uma falha em um deles.

Em consonância com essa pesquisa, autores usam rede de Petri aplicado em mecanismo de redundância, como o trabalho realizado por SOUSA et al. (2015), que utiliza rede de Petri para o mecanismo de redundância ativo-ativo, no entanto, não monitora recuperação do processo manual. Por outro lado, os pesquisadores em COSTA et al. (2015), apresentaram modelos que utilizam mecanismos de redundância ativo-ativo com *failover* automático e manual, no entanto, para isso eles utilizaram cadeias de *Markov* na sua implentação, ao invés de SPN. Além disso não abordaram todas as situações de sucesso e insucesso do processo manual e automático onde estes mecanismos podem ocorrer.

6.4 Modelo *Warm Standby*

O modelo em Cadeia de *Markov* de Tempo Contínuo (CTMC) da Figura 6.8 representa um modelo com o serviço de *Streaming* de Vídeo funcionando com redundância *warm standby*.

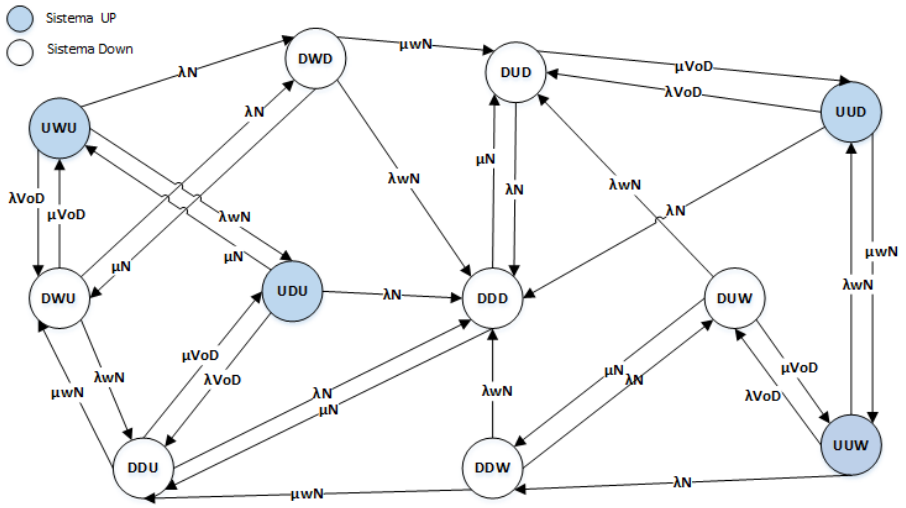


Figura 6.8: Modelo CTMC *warm standby*.

O modelo compreende os estados em azul UUW, UDU, UUD, e UWU (serviço disponível), e os estados transparentes, tais como: DDW, DUW, DDD, DWU, DDU, DWD, e DUD (serviço indisponível). A notação para os estados é baseada no estado atual de cada

componente. As três letras representam o estado inicial do funcionamento dos três componentes, respectivamente, o serviço, o primeiro *Node* (NC), e o segundo *Node*.

O serviço pode estar *Up* (U) ou *Down* (D). Os *Nodes* operam de forma alternada no modo *warm standby*, e apenas um deles deverá estar *Up* (U) por vez, não havendo hierarquia entre eles. Os *Nodes*, ainda, podem assumir a condição de aguardo (W) ou Down (D). Nesse modelo, o serviço inicial é representad0 por UWU, onde o serviço está disponível, o primeiro *Node* está em *warm standby*, e o segundo *node* está em execução. A partir desse estado, é possível mover-se para DWU (falha do serviço), DWD (falha segundo node), ou UDU (falha primeiro node). A partir do estado DWU (serviço está *Down*, primeiro *Node* em *warm standby* e segundo *Node UP*), pode ser alcançado UWU (representando reparação de serviço), DWD (falha no segundo *Node*), ou DDU (falha primeiro *Node*).

A partir do estado DWD, três outros caminhos são possíveis: falha do *Node* em *standby*, alcançando o estado DDD; ativação do *Node* em *standby*, alcançando o estado DUD; reparo do *Node* em falha, alcançando o estado DWU. A partir do estado DDU, surgem outras três possibilidades: falha do NC ativo, levando o sistema ao estado DDD, em destaque na Figura 6.8; reparo do serviço, alcançando o estado UDU; reparo do NC em *standby*, alcançando o estado DWU. Diante do estado DWU, é possível que haja o reparo do serviço, levando o sistema ao estado UWU; que haja a falha do *Node* ativo, levando o sistema ao estado DWD; que haja a falha do *Node* em *standby*, alcançando o estado DDU.

No estado DDD todos os componentes do sistema estão indisponíveis (D). A partir desse estado, dois estados podem ser alcançados: reparo do primeiro NC, levando o sistema ao estado DUD, ou o reparo do segundo NC, levando o sistema ao estado DDU. Partindo do estado DUD, outros três caminhos são possíveis: falha do NC ativo, alcançando o estado DDD; reparo e inicialização do serviço, levando ao estado UUD; reparo do segundo NC, levando-o ao modo de *standby* no estado DUW. No estado UUD, é possível que o serviço falhe, levando o sistema ao estado DUD; que o NC falho seja reparado no modo *standby*, atingindo o estado UUW; ou que haja a falha do primeiro NC, fazendo com que todos os componentes falhem, como indica o estado DDD.

A partir do estado UDU, é possível que o segundo NC falhe, fazendo com que todos os componentes estejam falhos, como expressa o estado DDD; que o serviço falhe, alcançando o estado DDU; que o primeiro NC seja reparado e opere no modo *standby*, através do estado UWU. No estado UDU, é possível que haja a falha de todos os componentes, alcançando assim o estado DDD; ou o reparo do primeiro NC, assumindo a condição de aguardo, como mostra o estado DWU; ou ainda que uma nova VM seja instanciada, tornando o serviço disponível novamente, alcançando o estado UDU.

A partir do estado DUW, é possível alcançar outros três caminhos: falha do NC em *standby*, alcançando o estado DUD; falha do NC ativo, atingindo o estado DDW, ou ainda o reparo do serviço, como indica o estado UUW.

No estado UUW, pode ocorrer a falha do segundo NC levando ao estado UUD, ou ocorrer

a falha do serviço alcançando, assim, o estado DUW ou ainda pode ser que ocorra a falha do primeiro NC, tornando o serviço indisponível no estado DDW. O estado DDW representa o sistema indisponível e, a partir desse estado, é possível que todos os componentes falhem através do estado DDD; que o segundo NC seja inicializado, alcançando o estado DDU; ou ainda que haja o reparo no primeiro NC, caracterizando o estado DUW.

A falha é um evento que ocorre quando o serviço provido se desvia do serviço pretendido (SILVA et al., 2015). As taxas de falha para os dois nós são representadas por λ_N, enquanto que μ_N representa a taxa de reparo do *Node*. O *node* em *warm standby* tem taxa de falha igual a λ_{WN}. Para que um NC em espera assuma a carga de trabalho do serviço, a taxa utilizada é μ_{WN}. A taxa de falha do serviço é λ_{VoD}, enquanto o reparo do serviço é μ_{VoD}. A ação-reparo do serviço é instanciação de uma nova VM, que inclui as aplicações necessárias ao funcionamento do serviço de VoD *streaming* que são o *Apache* e o VLC.

Por meio dos modelos hierárquicos e analíticos é possível iniciar a obtenção dos valores de dependabilidade da arquitetura com redundância nos *Nodes*. Primeiro, a partir do modelo CTMC da Figura 6.8, podemos obter a Equação 6.7 que representa a fórmula fechada para alcance da disponibilidade do bloco do Serviço com NCs redundantes.

$$A_{VoD} = \frac{\alpha(2\beta^2\alpha_1 + \beta\alpha_2\beta_1 + \alpha_3)\beta_1^2 + \alpha_4\beta_1^3}{\alpha_5(\beta^2\alpha_1(\lambda N + 2\mu N) + \beta\varphi\beta_1 + \varphi_1\beta_1^2 + \beta_7\beta^3)}, \tag{6.7}$$

Em que:
$\beta = \lambda_{WN},; \beta_1 = \mu_{WN}; \alpha = \mu_N\mu_{VoD}; \alpha_1 = (\lambda_N + \beta + \mu_N)2(\lambda_N + \beta + \mu_N),$
$\alpha_2 = 4\lambda_{N^2} + 17\lambda_N\beta + 13\beta + 5\lambda\mu_N + 12\beta\mu_N + 2\mu_{N^2}; \alpha_3 = (8\beta(2\beta + \mu_N) + \lambda_N(11\beta + \mu_N)),$
$\alpha_4 = (7\beta + 2\mu_N)(\mu_{WN^3}); \alpha_5 = \lambda_N + \lambda_{VoD} + \mu_{VoD},$
$\alpha_6 = 2\lambda_N(\lambda_N + \beta)(\lambda_N + 3\beta); \beta_2 = (8\lambda_{N^2} + 24\lambda_N\lambda_{WN} + 13\beta^2)\mu_N,$
$\beta_3 = (7\lambda_N + 12\beta)\mu_{N^2} + 2\mu_{N^3}; \beta_4 = 2\lambda_N\beta(2\lambda_N + 3\beta),$
$\beta_5 = (\lambda_N + \beta)(\lambda_N + 16\beta)\mu_N; \beta_6 = (\lambda_N + 8\beta)\mu_{N^2},$
$\beta_7 = (\lambda_N(\beta + 2\mu_N) + \mu_N(7\beta + 2\mu_N)); \varphi = \alpha_6 + \beta_2 + \beta_3, and \varphi_1 = \beta_4 + \beta_5 + \beta_6.$

A Tabela 6.4 contém todos os parâmetros utilizado no modelo *warm standby* com a sua respectiva descrição.

Tabela 6.4: Parâmetros de entrada para o CTMC do serviço com redundância.

Parâmetro	Descrição
λ_{N1}=λ_{N2}=λ_N	Tempo médio de falha do *node*
λ_{wN1}=λ_{wN2}=λ_{wN}	Tempo médio de falha do *node* em *standby*
μ_{N1}=μ_{N2}=μ_N	Tempo médio de reparo do *node*
μ_{wN1}=μ_{wN2}=μ_{wN}	Tempo médio de reparo do *node* em *standby*
λ_{VoD}	Tempo médio de falha do serviço
μ_{VoD}	Tempo médio de instanciar o serviço

Esse modelo utiliza cadeia de *Markov* para representar o mecanismo de redundância *warm standby* abordando as particularidades (como por exemplo, tempo de ativação do *node* em *standby*, tempo para instanciar o serviço) que envolvem os dois *Nodes* e o Serviço, explorando o

processo de ativação do NC em *warm standby* e a ativação do serviço de vídeo.

No entanto, os autores (DANTAS et al., 2012), (LAL; KUMAR; JOSHI, 1980) e (AMARI; PHAM; MISRA, 2012), utilizaram cadeia de *Markov* dentro de uma perspectiva diferente da proposta neste trabalho. Os autores (HUANG; LOMAN; SONG, 2014) e (TANNOUS et al., 2011) apresentaram uma abordagem de mecanismo *warm standby* através de modelos RBD.

6.5 Validação de Modelos

Os modelos *warm e cold standby* presentes nas Figuras 6.2 e 6.8, respectivamente, são compostos por um modelo CTMC do serviço e por modelos validados apresentados em (DANTAS et al., 2012), (BEZERRA et al., 2014),(XU et al., 2014). Por outro lado, o modelo ativo-ativo da Figura 6.6 foi baseado em uma cadeia de *Markov* apresentada em (BAUER; ADAMS; EUSTACE, 2011) e ilustrado por meio da Figura 6.9. O processo de validação do modelo CTMC do serviço e do modelo ativo-ativo são apresentados nessa seção.

A validação do CTMC do serviço foi realizada por meio de um experimento, através de injeção de falhas, em um ambiente de teste. Um cenário básico foi considerado para verificar se o comportamento do modelo desenvolvido é similar ao comportamento do ambiente de teste real. Esse experimento consistiu na injeção de falha nesse ambiente de testes, a partir da combinação do uso de valores do modelo e do ambiente real. No entanto, para injetar falhas no sistema, foi necessário criar um *script* através de *Shell Script*, que foi executado no ambiente por cinco dias. Um fator de redução de 1000 é usado para o MTTF, no entanto, no MTTR não foi aplicado nenhum fator de redução. Na Tabela 6.5 mostra os valores com o fator de redução.

Tabela 6.5: Fator de redução do MTTF.

Componente	Falha Real (h)	Fator de Redução (h)
Apache	788,4	0,788
VLC	336	0,336
VM	2880	2,880

O método de distribuição-F foi aplicado, e desse modo obtivemos o número de reparos e falhas usados no sistema real. Desse modo, o grau de liberdade é calculado a partir da soma do número de ocorrências de reparação de falhas (KEESEE, 1965b). Nesse caso, o grau de liberdade de 524 foi atingido, indicando os valores máximos e mínimos da distribuição F com 95% de confiança. O intervalo de confiança para a disponibilidade do sistema real pode ser calculado com o valor de ρ, utilizando a Equação 6.8.

$$\rho = \frac{Y_n}{S_n}, \tag{6.8}$$

Em que, Y_n é o número de componentes de serviço reparados, e S_n é o número de componentes de serviço que falharam. Consequentemente, são necessários os valores de ρ_L

mínimo e ρ_U máximo, e isso é conseguido dividindo-se esses valores pelos valores mínimo e máximo, respectivamente, encontrados na F-distribuição (KEESEE, 1965b).

Assim sendo, para calcular o intervalo de confiança para a disponibilidade do serviço real, A_U e A_L, esses respectivos valores são obtidos a partir da inversão dos valores de ρ_U e ρ_L, respectivamente, para encontrar A_U e A_L, que representam respectivamente, os valores da disponibilidade mínima e máxima encontrada no experimento, podem ser vistos na Tabela 6.6.

Tabela 6.6: Intervalo de confiança para A e ρ.

Intervalo de Confiança (95%)		
ρ	ρ_L	2,3792
	ρ_U	3,3625
A	A_L	0,2292
	A_U	0,2959

Após a definição do intervalo de confiança, o modelo proposto pode ser validado de acordo com os valores definidos na Tabela 6.6. As disponibilidades mínima e máxima é de 0,2292 e de 0,2959, respectivamente. A disponibilidade do sistema real atingida foi de 0,2921, portanto este valor se encontra dentro do intervalo definido, onde podemos afirmar que a validação do modelo ocorreu satisfatoriamente, uma vez que a disponibilidade do sistema real encontra-se dentro dos limites das disponibilidades identificadas no processo de validação.

Os Modelos de redundância ativo-ativo em SPN e cadeia de *Markov*, apresentados por meios das Figuras 6.6 e 6.9 respectivamente, possuem o mesmo princípio de funcionamento com a diferença que o modelo em CTMC não especifica a aplicação utilizada. O modelo CTMC (ver Figura 6.9) possui seis estados, sendo eles UC, DUP, DC, SIMP, FF, e DF, os detalhes de funcionamento desse modelo encontram-se em (BAUER; ADAMS; EUSTACE, 2011), página 47.

Para validar o modelo ativo-ativo utilizamos os mesmos parâmetros da cadeia de *Markov* no modelo ativo-ativo em SPN. Os resultados da disponibilidade do modelo de redundância ativo-ativo proposto e do modelo de redundância ativo-ativo apresentado em (BAUER; ADAMS; EUSTACE, 2011) (ver Figura 6.9) foram comparados. Para tal, foram adotados os mesmos parâmetros de entrada de dependabilidade do modelo CTMC da Figura 6.9, no modelo em SPN.

Na Tabela 6.7, encontramos a disponibilidade de 99,99871% para o modelo ativo-ativo em SPN, e para o modelo em cadeia de *Markov* a disponibilidade é 99,99724%. Essa pequena diferença provavelmente ocorreu em razão do modelo ativo-ativo em SPN considerar a aplicação. Desconsiderando a aplicação utilizada, o resultado da disponibilidade dos dois modelos foi de 99,99%. Esse resultado indica que o modelo de Redundância ativo-ativo apresentado está validado.

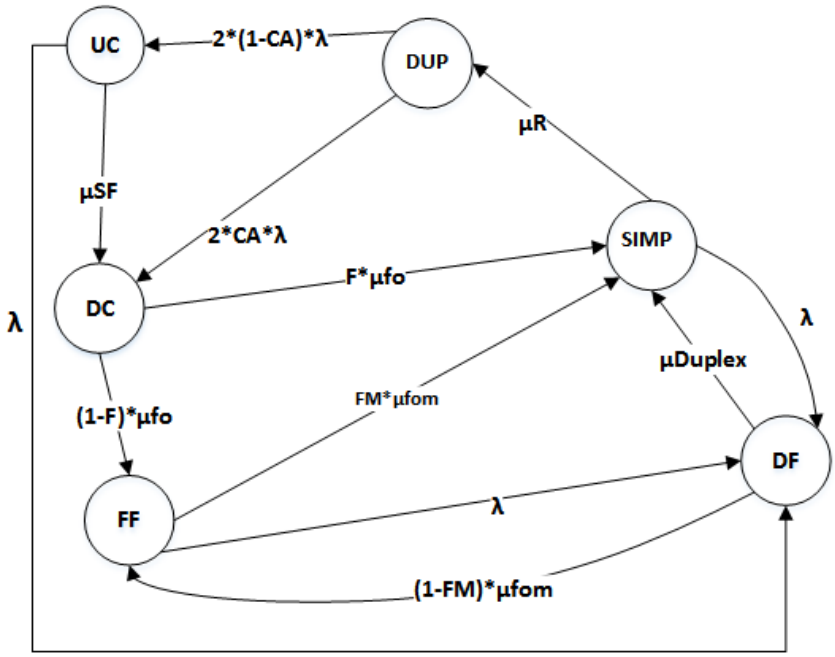


Figura 6.9: Modelo ativo-ativo em cadeia de *Markov*

Tabela 6.7: Resultados da disponibilidade dos modelos SPN e cadeia de *Markov*

Modelo ativo-ativo	Disponibilidade (A)
Cadeia de *Markov*	99,99724
Rede de Petri	99,99835

No próximo capítulo, apresentaremos três estudos de casos utilizados para validação dessa metodologia. Os primeiro e segundo estudos de caso estão relacionados ao serviço de *Streaming* de Vídeo e o terceiro está relacionado ao serviço MBaaS da plataforma *OpenMobester*, ambos hospedados numa plataforma de nuvem *Eucalyptus*.

7

Estudos de Casos

Este capítulo tem como objetivo apresentar três estudos de casos, onde demonstram a aplicação da metodologia proposta na avaliação de cenários em uma infraestrutura de nuvem. Eles têm como base a avaliação das métricas disponibilidade e *downtime*. Inicialmente, apresentaremos dois estudos de casos relacionados ao serviço de *Streaming* de Vídeo e um relacionado ao serviço *Mobile Backend as a Service* - MBaaS *OpenMobster* na plataforma *Eucalyptus*.

O primeiro estudo de caso tem por objetivo aplicar as estratégias de análise de sensibilidade propostas, para identificação dos pontos que requerem melhoria na disponibilidade do sistema, numa infraestrutura de nuvem hospedando o serviço de *Streaming* de Vídeo.

O segundo tem por finalidade verificar se a análise de sensibilidade realizada no estudo de caso 1 melhorou efetivamente a disponibilidade do sistema. O terceiro tem por finalidade aplicar metodologia para provimento de um serviço diferente do *Streaming* de Vídeo. Nesse caso, avaliaremos a disponibilidade e o *downtime* do serviço *Mobile Backend as a Service* - MBaaS *OpenMobster* na plataforma de nuvem *Eucalyptus*.

7.1 Arquitetura básica do serviço de *Streaming* de Vídeo

Este estudo de caso visa aplicar a metodologia proposta na avaliação da disponibilidade e *downtime* da infraestrutura de nuvem configurada com a plataforma *Eucalyptus* para o serviço de *Streaming* de Vídeo. Além disso, identificaremos os pontos que requerem melhoria na disponibilidade do sistema, a partir da utilização de estratégias de análise de sensibilidade propostas e existentes. Nessa identificação, a modelagem hierárquica e o modelo para representação do mecanismo de redundância do tipo *cold standby*, foi utilizado na obtenção desse resultado.

7.1.1 Descrição do sistema

Inicialmente, apresentaremos a **plataforma de nuvem *Eucalyptus*** e os seus principais componentes e características de funcionamento. O *Eucalyptus* é uma arquitetura de *software* baseada em *Linux*, que facilita a implementação da infraestrutura como Serviço (IaaS) de nuvens privadas e híbridas. Os clientes podem utilizar os próprios recursos do sistema, juntamente

com serviços de nuvem por meio de uma interface de auto-atendimento, de acordo com as suas necessidades em qualquer momento.

A estrutura de *software Eucalyptus* é modular (EUCALYPTUS, 2014b) e consiste em cinco componentes de alto nível, cada um com o próprio serviço. Estes são: i) o Controlador da Nuvem (CLC); ii) o Controlador do Cluster (CC), iii) o Controlador de Node ou Nó (NC), iv) o Controlador de Storage (SC), e v) o *Walrus*.

O CLC é o *Frontend*, ponto necessário para acesso dos clientes à infraestrutura de nuvem. Ele emprega *interfaces* de serviços *Web* para receber os pedidos de ferramenta de cliente de um lado, e interagir com os restantes dos componentes do *Eucalyptus*, no outro lado (EUCALYPTUS, 2014b),(EUCALYPTUS, 2014c). O CC geralmente é executado em uma máquina *Frontend* ou em qualquer máquina que tem conectividade de rede para NCs e a máquina executando o CLC (EUCALYPTUS, 2014b).

O NC é executado em cada Nó e controla o ciclo de vida das instâncias em execução no nó (EUCALYPTUS, 2014b). O NC interage com o sistema operacional e com o *hypervisor* executado no Nó. O SC fornece armazenamento de bloco persistente para uso pelas instâncias de máquinas virtuais (VM). *Walrus* é um serviço de armazenamento de dados baseado em arquivo que é compatível com a *interface* do *Simple Storage Service da Amazon - (S3)* (EUCALYPTUS, 2014b).

A **arquitetura do sistema para *Streaming* de Vídeo**, considerada nesse estudo, é baseada na plataforma *Eucalyptus*. O *Eucalyptus* foi escolhido entre outras estruturas de nuvem privada, em razão de ter sido usado em alguns provedores de serviço de *Streaming* de Vídeo, além de ter uma documentação abrangente e possuir modos de instalação flexível e rápida, essa plataforma possui máquinas virtuais que são compatíveis com o EC2 da *Amazon*. Isso significa que podemos instanciar a mesma imagem da VM que utilizaremos nesse estudo, na *Amazon*. Na Figura 7.1 apresentamos uma visão geral dos principais componentes do sistema,

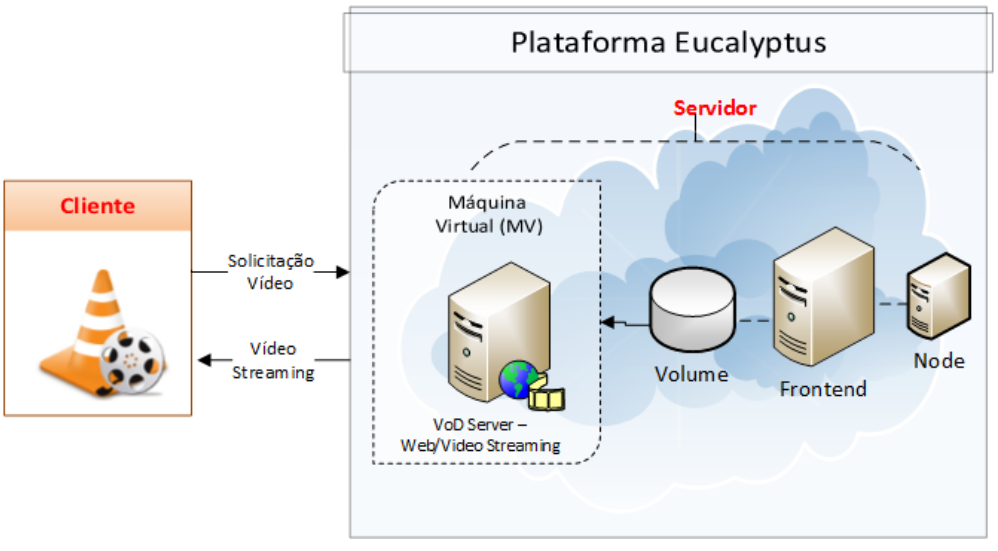


Figura 7.1: Arquitetutura do serviço de *Streaming* de Vídeo em nuvem privada

destacando os lados do Servidor e do Cliente. O lado servidor, conforme observamos na Figura 7.1, é formado por duas máquinas, sendo uma para o *Frontend* e outra para o *Node* ou Nó. No *Node* está a máquina virtual.

No entanto, no lado do Cliente, (ver Figura 7.1), observamos que o mesmo se conecta ao servidor de *Streaming* de Vídeo por meio da *internet*. O módulo Volume é o responsável pelo armazenamento. Ele está alocado no *Frontend* para armazenar a coleção de vídeos. A Máquina Virtual (VM) é a responsável por executar os aplicativos *Apache* e *VLC*, que são instanciados nos *Nodes*. A função do *player* da aplicação *VLC* é dar suporte para que o *streaming* aconteça e seja transmitido até o usuário. A aplicação *Apache* é responsável por hospedar o serviço em uma página *Web* dedicada. O usuário emite um pedido para a exibição de um vídeo hospedado em uma página *Web* específica. O *VLC*, por sua vez, obtém o vídeo solicitado a partir do volume de armazenamento remoto e transmite o *streaming* para o usuário. O aplicativo *VLC* foi escolhido para essa pesquisa em virtude de suportar uma variedade de formatos de vídeos.

As configurações de *hardware* das máquinas utilizadas nesse serviço de *streaming* de vídeo, estão apresentadas na Tabela 7.1. Com relação ao *software* utilizado, as máquinas do ambiente da nuvem utilizaram o *Eucalyptus* versão 3.2.2 e o sistema operacional *CentOS* CENTOS (2014) versão 6.4. A máquina utilizada como usuário faz uso do *Ubuntu Server Linux* versão 12.04.

Tabela 7.1: Configurações de *hardware* das máquinas utilizadas.

Recursos	**Frontend**	**Node**	**Usuário**
Processador	Intel Xeon E3	Xeon E5	Core i7
Memória (GB)	8	8	8
HD (GB)	500	1000	1000

Os modelos analíticos hierárquicos foram criados para descrever o comportamento de infraestruturas de nuvem privada, de acordo com a pilha geral de componentes para plataformas de nuvem *open-source*. Esses modelos propostos auxiliam no cálculo da disponibilidade do sistema.

7.1.2 Definir as métricas de interesse

Essa etapa exige uma definição das métricas de interesse a respeito do que se propõe a analisar. Inicialmente, analisaremos a disponibilidade e o *downtime* dessa arquitetura e para isso, é necessário identificar todas as taxas de falha (MTTF) e reparo (MTTR) de todos os componentes envolvidos na arquitetura.

7.1.3 Construir modelos de alto nível

Modelos analíticos hierárquicos foram utilizados para descrever o comportamento dessa arquitetura e ajudar a encontrar disponibilidade do sistema. A arquitetura do serviço de *Streaming*

de Vídeo pode ser representada por um modelo hierárquico RBD de alto nível baseado na nuvem Eucalyptus. O modelo pode ser dividido em quatro partes: *Frontend*, *Node* ou Nó, Volume e Serviço, que são representados pelo modelo RBD, conforme a Figura 7.2. A ferramenta *Mercury* (SILVA et al., 2015), foi utilizada para modelar e calcular a disponibilidade do sistema.

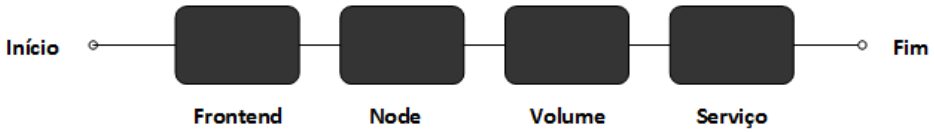


Figura 7.2: Modelo RBD do serviço de *streaming* de vídeo

Os modelos RBD permitiram o uso de expressões de forma fechada para calcular a disponibilidade do sistema no estado estacionário. Essa etapa facilita a aplicação das estratégias de análise de sensibilidade com respeito a cada um dos parâmetros do modelo.

A Equação 7.1, denota a expressão de fórmula fechada para a disponibilidade do sistema. Nessa Equação, A_f, A_n, A_v, e A_{VoD1} correspondem à disponibilidade do *Frontend, Node*, Volume, e Serviço, respectivamente e podem ser calculados a partir de cada um dos modelos RBD apresentado.

$$A_{S1} = A_{\mathrm{f}} \times A_{\mathrm{n}} \times A_{\mathrm{v}} \times A_{\mathrm{VoD1}} \tag{7.1}$$

7.1.4 Construir modelos detalhados

O subsistema do *Frontend* é composto por outros componentes que estão representados na Figura 7.3, por meio de um modelo RBD. Esse subsistema consiste de *Hardware*, Sistema Operacional, e os seguintes componentes de *Eucalyptus*: CLC, CC, SC e *Walrus*.

Figura 7.3: Modelo RBD do *Frontend*

Na Figura 7.4 apresentamos o modelo RBD que representa o *Node*. Além do sistema operacional e *hardware*, que também estão presentes no *Frontend*, o *Node* é formado pelos componentes *hypervisor*, Kernel-based Virtual Machine (KVM) (HU et al., 2010) e um controlador de *Node* do *Eucalyptus*. O *Node* está operacional, se e somente se, todos esses componentes estão ativos, sem falha.

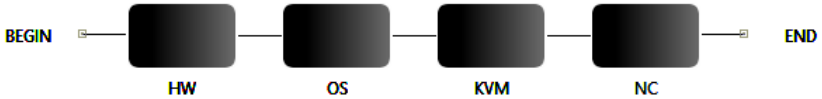


Figura 7.4: Modelo RBD do *Node*

O subsistema do Volume é um dispositivo NAS (*Network Attached Storage*) responsável por armazenar a coleção de vídeos, e não é representado por um submodelo. O subsistema serviço está refinado por uma CTMC na Figura 7.5, que permite calcular valores de disponibilidade para ser considerado no modelo de alto nível RBD (ver Figura 7.2).

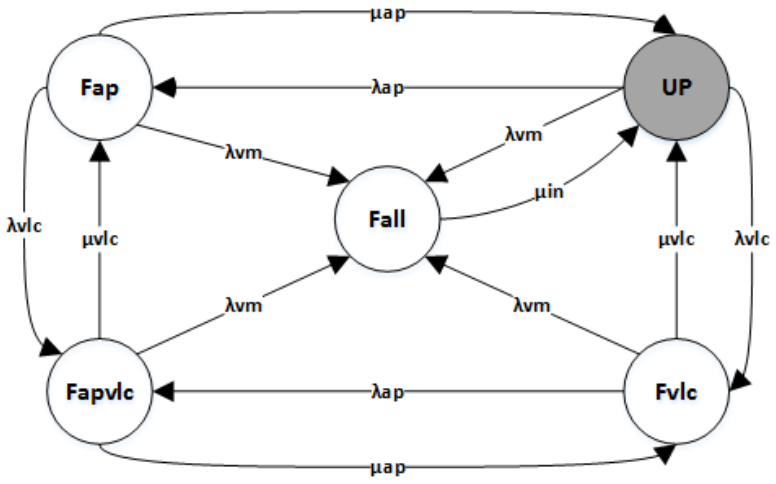


Figura 7.5: Modelo CTMC para o subsistema Serviço

A CTMC foi proposta devido à interdependência entre os componentes do sistema e pela necessidade de representar algumas particularidades que envolviam o funcionamento do subsistema serviço. Essas particularidades, são por exemplo, o tempo necessário para instanciar uma VM, uma vez que não recuperamos a VM, e sim instanciamos uma nova VM, em virtude do tempo de instanciação de uma nova VM ser menor do que o tempo de reparo da mesma. Além disso, essas particularidades também são os tempos de falha e reparo das aplicações *Apache* e VLC.

O modelo CTMC permite a concepção de uma Equação de fórmula fechada da disponibilidade no estado estacionário. A Equação é útil para fins de análise de sensibilidade, bem como, para calcular as métricas desejadas, sem incorrer numa solução numérica do modelo.

A CTMC representada na Figura 7.5 é um modelo de disponibilidade do sistema composto pelos seguintes componentes de *software*: *Apache*, VLC e VM. A CTMC tem cinco estados: UP, Fap, Fapvlc, Fvlc e Fall. Os círculos transparentes indicam os estados *down* - em que o serviço não está disponível devido a uma falha em algum componente e o círculo cinza indica o estado operacional.

Quando o *Apache* falhar, com taxa λ_{ap}, o modelo vai para o estado Fap e o serviço não é mais fornecido. A partir do estado *UP* o modelo também pode ir para o estado Fvlc, denotando a falha do aplicativo VLC com taxa de λ_{vlc}. O estado Fapvlc indica que ambos *Apache* e VLC falharam, esse estado pode ser alcançado a partir de estados: Fap ou Fvlc. O estado Fall representa a falha de todos os componentes, devido a uma falha na VM. Este estado é atingido sempre que a VM falhar, que ocorre com uma taxa de falha λ_{vm}. As taxas λ_{ap}, λ_{vlc} e λ_{vm} denotam a taxa de falha do *Apache*, VLC, e VM, respectivamente. As taxas de falha e

reparo são: $\lambda_i = 1/MTTF_i$, e $\mu_i = 1/MTTR_i$, $MTTF_i = \frac{1}{\lambda_i}$, e $MTTR_i = \frac{1}{\mu_i}$

A taxa de reparo para o *Apache* é μ_{ap}, e para o VLC é μ_{vlc} respectivamente. A taxa μ_{in} é a taxa para instanciar uma nova VM com o serviço, isto é, o inverso do tempo para iniciar uma VM após ser solicitado.

A Equação fechada 7.2 para calcular a disponibilidade do subsistema do Serviço A_{VoD1}, foi obtida da CTMC na Figura 7.5.

$$A_{VoD1} = \frac{\mu_{in}(\lambda_{ap}\lambda_{vm}(\beta) + \lambda_{ap}(\beta_1)\mu_{vlc} + (\beta_1)(\beta_2)(\beta + \mu_{vlc}))}{((\lambda_{ap} + \beta_1)(\lambda_{vm} + \mu_{in})(\beta)(\lambda_{ap} + \beta + \mu_{vlc}))}, \tag{7.2}$$

Em que: $\beta = \lambda_{vlc} + \lambda_{vm} + \mu_{ap}; \beta_1 = \lambda_{vm} + \mu_{ap}; \beta_2 = \lambda_{vm} + \mu_{vlc}$

A Equação de Fórmula fechada para calcular a disponibilidade da infraestrutura completa, A_{S1}, pode ser obtida como demonstrado na Equação 7.3. A_f, A_n, A_v podem ser calculadas a partir do RBD da Figura 7.2, enquanto A_{VoD1} é calculada da Equação 7.2. Os valores de MTTF e MTTR usados para o RBD de alto nível da Figura 7.2, são os valores de MTTF e MTTR correspondentes a cada submodelo do *Frontend*, *Node*, Volume e Serviço. A Equação 7.3 corresponde à disponibilidade total da infraestrutura, ao passo que A_f, A_n, A_v, e A_{VoD1} correspondem à disponibilidade do *Frontend*, *Node*, Volume e Serviço, respectivamente.

$$A_{S1} = A_f \times A_n \times A_v \times A_{VoD1} \tag{7.3}$$

7.1.5 Definição dos parâmetros de entrada

A Tabela 7.2 contém as taxas MTTF e MTTR para os elementos *Frontend*, *Node* e Volume desta arquitetura. Estes valores foram obtidos a partir de (DANTAS et al., 2012),(BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009). O cálculo de métricas de dependabilidade para o módulo *Frontend* produziu um MTTF de 180,72 horas e um MTTR de 0,96 horas.

Tabela 7.2: Parâmetros de entrada para o sistema RBD não-redundante

Módulo	Componente	MTTF	MTTR
Frontend	HW	8760 h	100 min
	SO	2895 h	1 h
	CLC	788,4 h	1 h
	CC	788,4 h	1 h
	SC	788,4 h	1 h
	Walrus	788,4 h	1 h
Node	KVM	2990 h	1 h
	NC	788,4 h	1 h
Volume	Volume	100000 h	1 h

Os mesmos valores de MTTF e MTTR (DANTAS et al., 2012) são assumidos para os elementos de HW e SO do subsistema *Nodes*, Tabela 7.2 inclui os valores de parâmetros para o KVM e NC (BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009). A Análise do modelo deste subsistema produz um MTTF de 481,82 horas e um MTTR de 0,91 horas.

Tabela 7.3: Entrada dos parâmetros para o RBD de alto nível

Componente	MTTF	MTTR
Frontend	180,72 h	0,96999 h
Node	481,83 h	0,91000 h
Volume	100000 h	1 h
Serviço	217,77 h	0,92633 h

Na Tabela 7.3 é possível identificar os parâmetros do modelo RBD para a estrutura apresentada na Figura 7.2. Calculamos o MTTF e MTTR de cada subsistema utilizando os correspondentes modelos RBD e CTMC. A análise forneceu os valores apresentados na Tabela 7.3. A disponibilidade do módulo de serviço é computada através do CTMC representado na Figura 7.5, na página 86.

Tabela 7.4: Valores dos parâmetros para o modelo CTMC

Parâmetro	Descrição	Valor (h)
$1/\lambda_{ap}$	tempo médio de falha do apache	788,4
$1/\lambda_{vlc}$	tempo médio de falha do vlc	336
$1/\lambda_{vm}$	tempo médio de falha da vm	2880
$1/\mu_{ap}$	tempo médio de reparo do apache	1
$1/\mu_{vlc}$	tempo médio de reparo do vlc	1
$1/\mu_{in}$	tempo médio para instanciar uma nova vm	0,019166

Na Tabela 7.4 são apresentados os valores de todos os parâmetros utilizados para calcular a disponibilidade do módulo de serviço. Os valores baseiam-se nas análises apresentadas em (DANTAS et al., 2012),(BEZERRA et al., 2014), exceto o μ_{in}, que é o inverso do MTTI, o tempo médio para instanciar um nova VM com serviço de *Streaming* de Vídeo, calculado através da Equação 7.4, em que MTVM é o tempo médio para iniciar uma nova VM e MTTS é o tempo médio para o início ao serviço de *Streaming* de Vídeo, com as aplicações *Apache* e VLC.

$$\mu_{in} = \frac{1}{MTTI} = \frac{1}{MTVM + MTSS} \tag{7.4}$$

Os valores de MTVM e MTSS foram estimados por meio de experimentos. Para esses experimentos, *scripts* de monitoramento foram criados usando os utilitários de *Linux*, tais como: *data* e *mpstat* (BLUM, 2008). Os *scripts* foram usados para monitorar a inicialização do serviço. Foram realizadas 100 medições para obtenção do MTSS e do MTVM.

7.1.6 Avaliação do modelo hierárquico

Calculamos as medidas de disponibilidade usando os parâmetros de entrada mencionados no modelo hierárquico. Ambos, RBD e CTMC foram resolvidos numericamente, e obtivemos a disponibilidade para cada submodelo. Assim, os correspondentes valores de disponibilidade de cada um dos subsistemas foram usadas no modelo RBD de alto nível.

Na Tabela 7.5 apresentamos a disponibilidade da arquitetura apresentada considerando apenas um *Node*, um *Frontend*, um Volume e um módulo que representa o Serviço. Além da disponibilidade no estado estacionário, na Tabela 7.5 mostramos o número de noves (MARWAH et al., 2010), o que constitui uma visão logarítmica da disponibilidade e do tempo de inatividade, *downtime*, o que melhor representa o impacto da indisponibilidade do serviço do usuário.

Tabela 7.5: Medidas de disponibilidade e *downtime* da arquitetura básica do serviço de *streaming* de vídeo

Medidas	**Arquitetura**
Disponibilidade	0,988571
Número de 9's	1,9420
Downtime Anual	100,12 horas

Esses resultados revelam que para a configuração de parâmetros apresentados, encontramos um valor de 0,9885713 para a disponibilidade do sistema de *Streaming* de Vídeo. Essa disponibilidade corresponde a cerca de 100 horas de tempo de inatividade em um ano. Portanto, este sistema requer que apliquemos as estratégias de análise de sensibilidade para identificar os componentes críticos do sistema com relação à disponibilidade nessa arquitetura, e assim, orientar as futuras melhorias na disponibilidade do sistema.

7.1.7 Estratégias de análise de sensibilidade nos modelos

Na Figura 7.6 mostramos a análise de sensibilidade realizada por meio da EAS **variação um por um**, ou seja, a variação de um parâmetro de cada vez, para cada um dos componentes mostrados na Figura 7.2, com os correspondentes efeitos sobre a disponibilidade do sistema. Mudamos o valor de cada parâmetro em passos médios de 10% em relação ao valor nominal do parâmetro (MATOS; MACIEL; SILVA, 2015).

A Equação 7.5 foi utilizada para o cálculo dessa disponibilidade a partir da variação dos parâmetros de entrada da arquitetura do sistema. A Equação 7.5 pode também ser apresentada por meio da Equação 7.6, uma vez que a disponibilidade do Serviço é representada por um bloco RBD, mas possui uma cadeia de *Markov* como submodelo que foi apresentado na Seção 7.1.4.

$$A_{S1} = A_f \times A_n \times A_v \times A_{VoD1} \tag{7.5}$$

$$A_{S1} = A_f \times A_n \times A_v \times \frac{\mu_{in}(\lambda_{ap}\lambda_{vm}(\beta) + \lambda_{ap}(\beta_1)\mu_{vlc} + (\beta_1)(\beta_2)(\beta + \mu_{vlc}))}{((\lambda_{ap} + \beta_1)(\lambda_{vm} + \mu_{in})(\beta)(\lambda_{ap} + \beta + \mu_{vlc}))} \quad (7.6)$$

Na Figura 7.6 apresentamos o comportamento de todos os componentes do sistema, a partir da variação individual de cada parâmetro da Equação 7.6, enfatizando que a variação dos parâmentos ocorre uma de cada vez.

Analisaremos cada componente da Figura 7.6, desse modo ma Figura 7.6 (a) apresentamos o parâmetro de MTTF do componente *Frontend*, em que o valor da disponibilidade é incrementado a passo de 10% do valor inicial, durante um intervalo de 180 a 378 horas. Enfatizando que esse parâmetro tem um valor de sensibilidade positiva, pois observa-se que o aumento do tempo de falha produz uma variação significativa na disponibilidade em comparação com a *baseline* que é a disponibilidade de referência, 0,988571. A disponibilidade é inicialmente de 0,988217205 com o tempo de falha mais curto, e termina no valor inicial de 0,990999557 com o tempo de falha mais longo, um aumento de $0,28\%$ na disponibilidade em relação à inicial.

Na Figura 7.6 (b), encontramos o parâmetro de MTTF do componente *Node*. Os valores do parâmetro MTTF do *Node* foram variados ao longo de um intervalo de 480 a 1080 horas. Os resultados de disponibilidade do parâmetro MTTF do *Node* começam em 0,988231206 e terminam em 0,98921169. Esse resultado produz uma redução de $8,33\%$ no tempo de inatividade em relação à disponibilidade inicial, o que pode ser considerado um valor significativo.

Na Figura 7.6 (c), apresentamos, o parâmetro de MTTF do componente Volume. Os valores do parâmetro MTTF do volume foram variados ao longo de um intervalo de 10000 a 210000 horas. Os resultados de disponibilidade do parâmetro MTTF do Volume começam em 0,988238309 e terminam em 0,988243485. Esse resultado produz um pequeno incremento na disponibilidade do sistema, em virtude desse componente possuir um MTTF significativo (KIM; MACHIDA; TRIVEDI, 2009).

Na Figura 7.6 (d) apresentamos o parâmetro de MTTR para componente Volume. Os valores do parâmetro MTTR do volume foram variados de 0,1 a 1 hora, redução de $0,96\%$ no tempo de inatividade em relação ao tempo de reparo inicial. Esse componente, não é crítico para a disponibilidade do sistema, em contraste com os subsistemas *Frontend*, *Node* e o Serviço.

Na Figura 7.6(e) e 7.6(f) apresentamos os parâmetros MTTR dos componentes *Frontend e Node* respectivamente. Os valores do parâmetro MTTR do *Frontend* foram variados ao longo de um intervalo de 0,097 a 0,97 horas e do *Node* em 0,09 a 0,91 horas. Observamos que, nos dois componentes, à medida em que reduzimos o tempo de reparo desses, temos um incremento na disponibilidade do sistema. Isso reforça a teoria de que quanto menor for o tempo de reparo de um sistema, maior será o tempo em que esse sistema ficará funcionando.

Nas Figuras 7.6 (g),(h) e (k), apresentamos os parâmetros MTTF da aplicação *Apache*, do componente VM, e da aplicação VLC respectivamente. Os valores do parâmetro MTTF foram variadas ao longo de um intervalo de 394,2 a 1182,6 horas para o *Apache*, 1440 a 4320 horas para

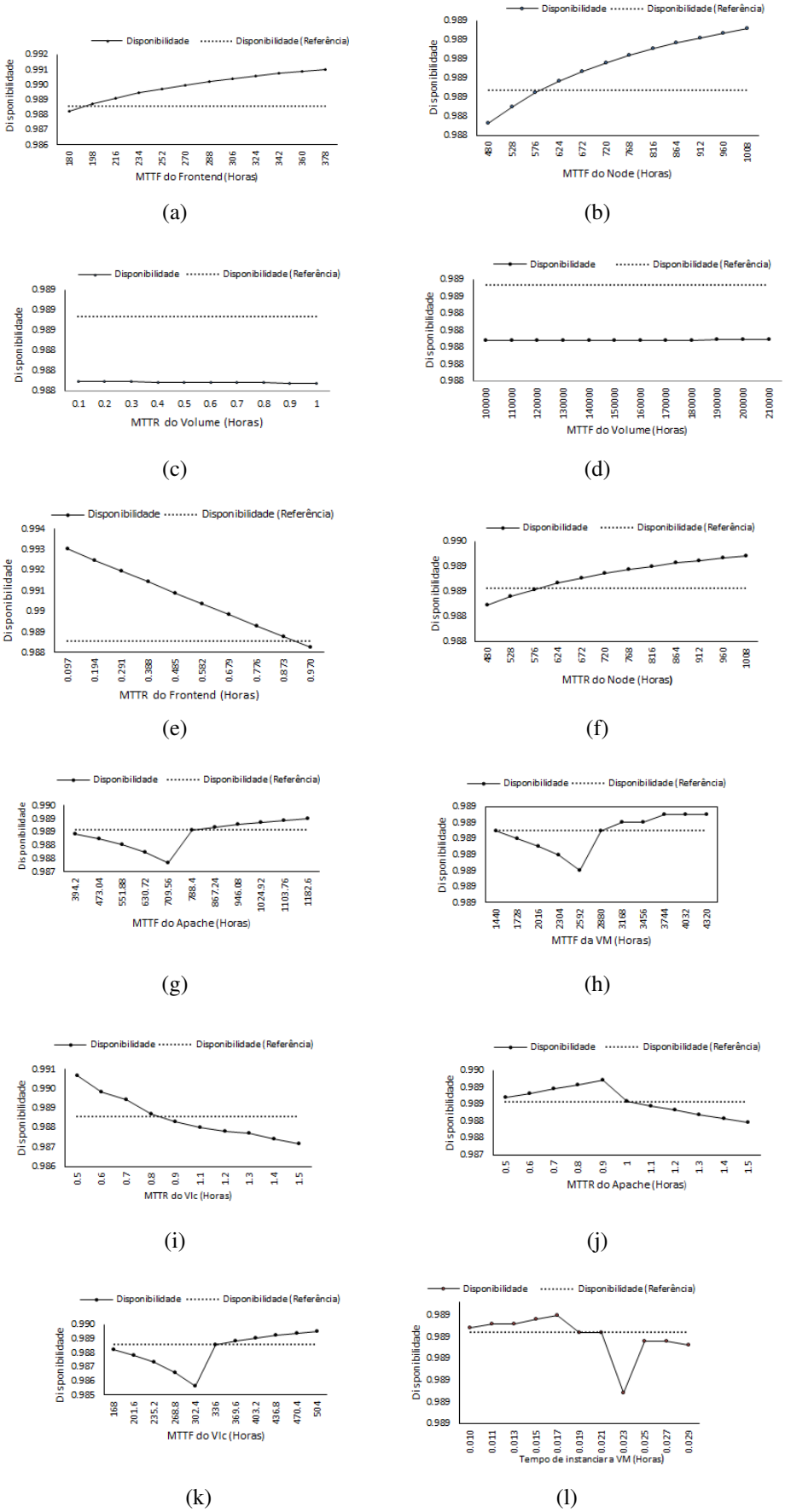


Figura 7.6: Análise de sensibilidade - Variação um por um

a VM e para o VLC de 168 a 504 horas. Nos três gráficos, observamos que a disponibilidade apresenta uma queda brusca, isso aconteceu porque os valores de MTTF que antecedem a 788,4 horas para o *Apache*, 2880 horas para VM, e 336 horas para o VLC não são os valores nominais de falha para esses componentes. Desse modo, observa-se que na medida em que eles atingem os patamares do valor nominal do MTTF, a disponibilidade volta ao seu comportamento normal.

Nas Figuras 7.6 (i) e (j) respectivamente, apresentamos os parâmetros MTTR das aplicações VLC e *Apache*. Os valores do parâmetros MTTR foram variados ao longo de um intervalo de 0,5 a 1,5 horas para as duas aplicações. Nos dois gráficos, observamos que a disponibilidade aumenta e depois apresenta uma queda brusca, isso aconteceu porque as aplicações estavam trabalhando com taxa de reparo abaixo do valor nominal esperado para cada aplicação desse modo, quanto menor o tempo de reparo, maior é a disponibilidade do sistema.

Na Figura 7.6(l), apresentamos o parâmetro que corresponde ao tempo para instanciar uma nova máquina virtual. Os valores desses parâmetros foram variados ao longo de um intervalo de 0,010 a 0,029 horas. Observamos que na medida em que demoramos para a instanciar uma VM, a disponibilidade do sistema decresce e no tempo 0,023 horas a disponibilidade é 0,988557 levando o sistema a um incremento no *downtime* de $0,13\%$. Esse aumento parece ser insignificante, mas não é em virtude do tempo de ativação ser pequeno.

A abordagem realizada por meio da EAS, variação um por um, proporcionou uma visão geral do sistema, apresentando o comportamento dos parâmetros quando a sua taxa de falha ou de reparo sofre variação ao longo do tempo. Identificamos que a disponibilidade do sistema sofre uma influência signiticativa das taxas de falha e reparo referente aos componentes *Frontend* e *Node*, na medida em que esses parâmetros são variados.

A taxa de falha relacionada ao componente Volume não apresentou uma variação significativa na disponibilidade mediante a variação dos seus parâmetros, provavelmente esse evento ocorreu em razão da métrica adotada para variação ser de 10% e como esse parâmetro possui uma taxa de falha expressiva, a variação desse parâmetro, nessas proporções, não causou um impacto expressivo. O comportamento do componente Serviço composto pelos parâmetros λ_{vlc}, μ_{vlc}, λ_{ap}, μ_{ap}, λ_{vm} e o μ_{in} similar ao dos componentes *Frontend* e *Node*.

Aplicando a estratégia de ***design of experiment*** (DoE) para o modelo hierárquico da Figura 7.2, vide página 85, esse modelo tem a disponibilidade expressa por meio da Equação 7.5, na página 89. Foram utilizados, doze (12) fatores e dois (2) níveis para o cálculo do DoE, por meio do *software Minitab* (MATHEWS, 2005).

Em razão de termos um número de fatores maior que cinco, recomenda-se adotar fatorial fracional com um número reduzido de execuções necessárias para construir um experimento (MATHEWS, 2005). Os valores adotados correspondem aos níveis máximos e mínimos de taxa de falha e de reparo do sistema. Os valores dos níveis mínimos das taxas de falha e reparo foram adotados de acordo com (BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009).

No entanto, para os níveis máximos de taxa de falha, foi utilizado um período de falha de 4380 horas, 6 (seis) meses. Entendemos que esse período seja razoável para ocorrer uma

falha nos componentes e subsistemas analisados, uma vez que não empregam a redundância. Por outro lado, os valores máximos adaptados para as taxas de reparo foram em torno de 100% do valor mínimo, considerando que há uma equipe exclusiva de manutenção.

Foram obtidos 16 cenários a partir do *Minitab*, ferramenta estatística utilizada para calcular o DoE. O resultado da simulação foram gerados a partir dos modelos RBD e CTMC (Figuras 7.2 e 7.5) e da Equação 7.5 de disponibilidade, na página 89. Na Tabela 7.6 mostramos o *ranking* de sensibilidade obtido por meio da análise realizada pela estratégia DoE. Os parâmetros taxa de reparo do *Frontend* (μ_f) e a taxa de falha do *Node* (λ_n) são os principais parâmetros do sistema.

Tabela 7.6: *Ranking* de sensibilidade obtidos por meio das estratégias DoE e Diferença Percentual

Parâmetros	**DoE -**$SS(\|A\|)$	Parâmetro	**DP- -**$SS(\|A\|)$
μ_f	0,010570	μ_f	0,00566905338673
λ_n	0,008163	λ_f	0,00511843110575
μ_{in}	0,007677	λ_{vlc}	0,00273877590683
λ_{vol}	0,007579	μ_n	0,00225794562354
μ_n	0,005622	λ_n	0,00167770041730
λ_{ap}	0,005616	μ_{vlc}	0,00148511046863
λ_{vlc}	0,005507	λ_{ap}	0,00103840298868
μ_{vol}	0,004782	μ_{ap}	0,00063346356017
λ_{vm}	0,004685	μ_{in}	0,00005119201992
μ_{vlc}	0,004316	μ_{vol}	0,00001000128544
μ_{ap}	0,003945	λ_{vm}	0,00000177471889
λ_f	0,000141	λ_{vol}	0,00000090769640

A estratégia **diferença de percentual** (DP) é uma das aplicadas ao modelo de fluxo contínuo de vídeo e por esse método é possível identificar os componentes que afetam a disponibilidade do sistema. A Tabela 7.6 apresenta o *ranking* da análise de sensibilidade em ordem descendente. Essa estratégia apresentou em primeiro lugar o mesmo parâmetro, μ_f que a estratégia *design of experiment*. Os parâmetros λ_n e μ_{in} ocupam a segunda e terceira posições do *ranking*. Observamos ainda que o parâmetro λ_n aparece na quinta posição do *ranking* da estratégia DP, enquanto esse mesmo aparece na segunda posição do *ranking* na estratégia DoE.

Para estratégia **DASI** utilizaremos a seguinte Equação 7.7, para obtermos o índice de

sensibilidade dessa estratégia, para o serviço de *Streaming* de Vídeo.

$$
\begin{aligned}
DASI^{*}_{\theta_i}(A_{S1}) = & \frac{\sum_{i=1}^{n} \frac{\partial A_f}{\partial \theta}(\frac{\theta}{A})|_{\theta=\theta i}}{n} \times A_n \times A_v \times A_{VoD1} + \\
& Af \times \frac{\sum_{i=1}^{n} \frac{\partial A_n}{\partial \theta}(\frac{\theta}{A})|_{\theta=\theta i}}{n} \times A_v \times A_{VoD1} + \\
& Af \times A_n \times \frac{\sum_{i=1}^{n} \frac{\partial A_v}{\partial \theta}(\frac{\theta}{A})|_{\theta=\theta i}}{n} \times A_{VoD1} + \\
& Af \times A_n \times A_v \times \frac{\sum_{i=1}^{n} \frac{\partial A_{VoD1}}{\partial \theta}(\frac{\theta}{A})|_{\theta=\theta i}}{n}
\end{aligned}
\tag{7.7}
$$

As expressões das derivadas parciais para o subsistema A_f, A_{Node} e A_v, estão descritas no Apêndice A.

As expressões das derivadas parciais para o subsistema A_{VoD1} com relação a cada um dos parâmetros ($\frac{\partial A_{VoD1}}{\partial \lambda_{vm}}$, $\frac{\partial A_{VoD2}}{\partial \lambda_{ap}}$, $\frac{\partial A_{VoD1}}{\partial \lambda_{vlc}}$, $\frac{\partial A_{VoD1}}{\partial \mu_{in}}$, $\frac{\partial A_{VoD1}}{\partial \mu_{ap}}$ e $\frac{\partial A_{VoD1}}{\partial \mu_{vlc}}$) do modelo CTMC não são descritas aqui, mas o leitor pode vê-las no Apêndice A.

A Tabela 7.7 apresenta o *ranking* da análise de sensibilidade em ordem decrescente. Os dados de entrada utilizados para esse *ranking* foram obtidos utilizando uma faixa de valores entre -50% a +50% do valor nominal do parâmetro. Para esse cálculo posicionamos o valor nominal no centro e calculamos o percentual sugerido para uma amostra de 11 valores, conforme dito na seção 4.2, página 49.

Tabela 7.7: *Ranking* de sensibilidade obtido por meio das estratégias DASI e SDP no modelo *Baseline*

Parâmetro	$DASI^{*}_{\theta i}(\lvert A_{S1}\rvert)$	**Parâmetro**	$SDP^{*}_{\theta i}(\lvert A_{S1}\rvert)$
μ_n	0,098386059	λ_n	$1,8850725 \times 10^{-3}$
μ_f	0,005861235	λ_f	$1,5866281 \times 10^{-5}$
λ_f	0,005331803	μ_f	$1,5866281 \times 10^{-3}$
λ_{vlc}	0,00296418	λ_{vlc}	$8,8157368 \times 10^{-6}$
μ_{vlc}	0,002754658	μ_{vlc}	$8,8157368 \times 10^{-6}$
λ_n	0,001884206	μ_n	$5,5918029 \times 10^{-6}$
μ_{ap}	0,001391681	λ_{ap}	$3,7634921 \times 10^{-6}$
λ_{ap}	0,001265953	μ_{ap}	$3,7621848 \times 10^{-6}$
μ_{vol}	0,000011290	μ_{vol}	$2,9719022 \times 10^{-8}$
λ_{vol}	0,0000099999	λ_{vol}	$2,9719022 \times 10^{-8}$
λ_{vm}	0,0000051851	μ_{in}	$1,9777666 \times 10^{-8}$
μ_{in}	0,0000001542	λ_{vm}	$1,5409453 \times 10^{-8}$

Identificamos no topo do *ranking* da análise de sensibilidade para a estratégia DASI, os componentes μ_n, μ_f, λ_f, λ_{vlc}, como os principais componentes candidatos a ações de melhorias do sistema. Essa estratégia nos traz a possibilidade de não calcular o índice de sensibilidade para um único parâmetro de configuração mas, nos permite a flexibilidade da variação de valores. Comparando as estratégias DoE e DP com a DASI, identificamos que os parâmetros μ_n, μ_f, λ_f,

λ_{vlc} e λ_n, estão entre as primeiras posições do *ranking* das três estratégias.

Realizando uma comparação da estratégia DASI na Tabela 7.7, com os resultados obtidos a partir do *ranking* de sensibilidade da estratégia sensibilidade diferencial paramétrica, apresentada na Tabela 7.7. Identificamos que o *ranking* obtido pela estratégia DASI é similar ao obtido pela EAS paramétrica. Isso indica que a estratégia DASI obteve um resultado coerente em relação à estratégia sensibilidade diferencial paramétrica.

Desse modo, isso indica que a alteração realizada na estratégia proporcionou resultados satisfatórios. Os resultados são equivalentes, no entanto, na estratégia DASI a taxa de reparo dos componentes aparecem antes da taxa de falha na maioria dos casos. Esse evento é interessante, em virtude desse tempo ser pequeno e na maioria das vezes dispomos de apenas uma equipe de manutenção para realizar a manutenção em toda a infraestrutura. Essa informação pode ser valiosa para prever o planejamento com relação às equipes de manutenção.

Aplicando a estratégia **Importância crítica para a disponibilidade - ICD**, observamos o resultado do *ranking* de sensibilidade da Tabela 7.8 para o caminho operacional. O resultado aponta que o *Frontend* e o Serviço são os componentes mais relevantes do sistema. Por outro lado, o resultado do *ranking* de sensibilidade da Tabela 7.8 para o caminho com falha elencou o Volume e o *Node* como os componentes mais importantes do sistema.

Tabela 7.8: *Ranking* de sensibilidade para o caminho operacional e não operacional

Parâmetro	Operacional	**Parâmetro**	**Com falha**
Frontend	1	Volume	1
Serviço	0,997785769	Node	0,838934368
Node	0,993091545	Serviço	0,635291278
Volume	0,989370762	Frontend	0,539065231

7.1.8 Identificar os componentes relevantes para as EAS

Nessa etapa identificaremos os componentes comuns às estratégias utilizadas nesse estudo de caso. Na Tabela 7.9 mostramos o *ranking* obtido a partir da métrica estabelecida por meio da Equação 5.1, no Capítulo 5, página 63. Esse resultado apresenta um *ranking* a partir da combinação das estratégias: *design of experiments* (DoE), diferença percentual e DASI. A estratégia variação um por um e ICD não foram contabilizadas no *ranking* da Tabela 7.9. A estratégia ICD apresenta como resultado o componente, essa estratégia é diferente das demais, que apresentam os parâmetros para cada componente. Por outro lado, a análise da estratégia variação um por um não oferece como resposta final um *ranking* de classificação. Desse modo, essas duas estratégias não participaram da métrica estabelecida pela Equação 5.1, no entanto, elas participam da análise realizada e estão presentes na composição do resultado.

Tabela 7.9: Ranking produzido a partir da avaliação de EAS em conjunto

Parãmetro	**Valor**
μ_n	0,100114945
μ_f	0,019169671
λ_n	0,004731074
λ_f	0,004348233
λ_{vlc}	0,002635873
μ_{in}	0,002564701
λ_{vol}	0,001895826
λ_{ap}	0,001242587
μ_{vlc}	0,001149682
μ_{ap}	0,000636631
μ_{vol}	0,000600005
λ_{vm}	0,000521188

Os valores na Tabela 7.9 indicam a posição da classificação final para os componentes do sistema. Apesar do resultado da avaliação da estratégia ICD não aparecer diretamente na Tabela 7.9, ele está presente e foi considerado na análise. A estratégia ICD aponta os componentes *Frontend* e Serviço, no caminho operacional, e os componentes Volume e *Node*, no caminho com falha, como os principais componentes do sistema.

Na Tabela 7.9 é possível identificar que estes componentes da estratégia ICD estão representados por meio dos parâmetros μ_f, λ_f, que correspondem ao componente *Frontend* ocupando a segunda e a quarta posições do *ranking*, os parâmetros λ_n e μ_n que correspondem ao componente *Node*, ocupando a primeira e a terceira posição do *ranking* e o componente Serviço que está representado por meio dos parâmetros λ_{vlc}, μ_{in}, λ_{ap}, μ_{ap} e λ_{vm}, ocupando algumas posições do *ranking*. Os parâmetros λ_{vol} e μ_{vol} que correspondem ao componente Volume ocupando a sétima e a penúltima posições do *ranking*.

Todos esses parâmetros têm um impacto direto na disponibilidade do sistema, no entanto observa-se que os parâmetros μ_n,μ_f, λ_n são os principais pontos da infraestrutura que aparecem em destaque nas primeiras posições do *ranking*. Apresentando esses parâmetros em termos de componentes temos: *Frontend* e *Node*. Desse modo, o componente para iniciarmos uma ação é *Frontend*, através dos parâmetros μ_f, λ_f, ou o componente *Node*, por meio dos parâmetros λ_n μ_n, e o Serviço por meio dos parâmetros λ_{vlc} e μ_{in}.

Observa-se que cada estratégia de análise de sensibilidade produz diferentes resultados no seu *ranking* quando comparadas; alguns parâmetros aparecem em posições diferentes, desencontradas ou alternadas, por isso em (FREY; MOKHTARI; DANISH, 2003) esse resultado é esperado e recomenda-se utilizar mais de uma EAS para avaliação de um sistema. Nesse estudo, optamos por realizar essa análise, de forma a obter resultados com alta confiabilidade a respeito de qual componente deve ser analisado para proporcionar ações de melhoria. Os modelos para representação de mecanismos de redundância são algumas das possíveis ações de melhoria que podem ser implementadas a fim de garantir a disponibilidade do sistema.

A partir da análise de sensibilidade realizada, um dos componentes é identificado como elemento chave para prover melhoria do sistema foi o componente *Node*. Desse modo, abordaremos um modelo com mecanismo de redundância *cold standby* implementado no *Node* e avaliaremos a disponibilidade após a implementação da redundância.

Inicialmente, utilizaremos um modelo RBD de alto nível, da Figura 7.7, para representar a disponibilidade do sistema para o serviço *Streaming* de Vídeo. Esse modelo representa uma arquitetura que possui uma máquina para o *Frontend* e duas máquinas para os *Nodes*. Além disso a máquina do *Frontend* é a responsável pelo controle dos *Nodes*. Essa arquitetura é representada pelo bloco RBD SSMARKOV. O bloco Volume, em série com o SSMARKOV, é o responsável pelo armazenamento dos vídeos.

Figura 7.7: Modelo RBD para sistema *streaming* de vídeo

A Equação da disponibilidade do sistema representada pelo modelo da Figura 7.7, é obtida através da Equação 7.8:

$$A_s = A_{SSMARKOV} \times A_{VOLUME} \tag{7.8}$$

Em virtude de algumas particularidades, como por exemplo o tempo de ativação do *Node* que o modelo de redundância possui, o modelo RBD sozinho, não conseguiria representar esse detalhe, por isso foi necessário refinar o bloco RBD SSMARKOV para uma cadeia de *Markov*. Desta modo, o bloco RBD SSMARKOV foi representado através da cadeia de *Markov* da Figura 7.8.

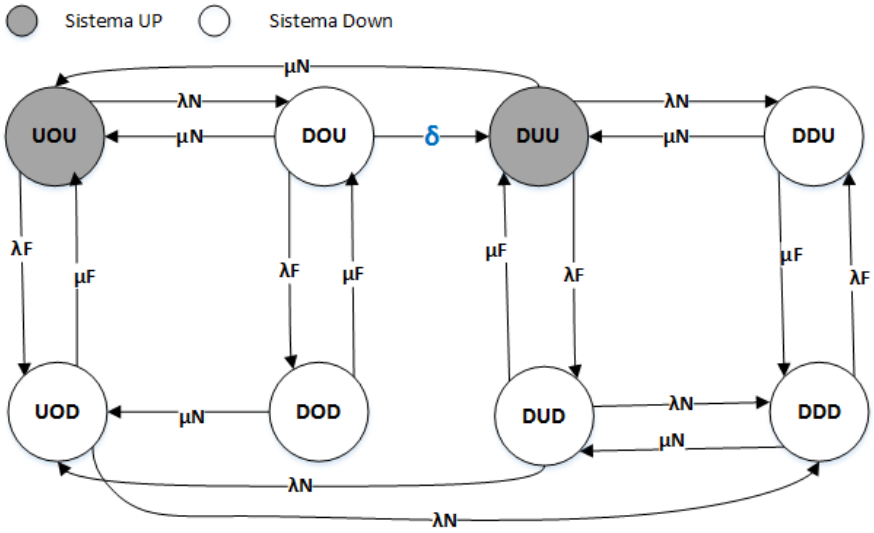


Figura 7.8: Modelo *cold standby*

Esse modelo foi apresentado em detalhes com relação ao modo de funcionamento e com as equações que determinam a disponibilidade do mesmo na seção 6.2, página 68.

Os parâmetros de MTTF e MTTR da Tabela 7.10 foram utilizados como dados de entrada para o modelo *cold standby* da cadeia de *Markov* da Figura 7.8 e estão consolidados na Tabela 7.11.

Tabela 7.10: Parâmetros de entrada para o modelo RBD cold standby

Módulo	Componente	MTTF	MTTR
Frontend	HW	8760 h	100 min
	SO	2895 h	1 h
	CLC	788,4 h	1 h
	CC	788,4 h	1 h
	SC	788,4 h	1 h
	Walrus	788,4 h	1 h
Node	KVM	2990 h	1 h
	NC	788,4 h	1 h
	HW	8760 h	100 min
	SO	2895 h	1 h
	VM	2880 h	0,019166 h
	VLC	788,4 h	1 h
	APACHE	336 h	1 h
Volume	Volume	100000 h	1 h
SSMARKOV	SSMARKOV	81,97 h	0,4470 h

O valor de ativação utilizado para o *Node* secundário foi baseado em (DANTAS et al., 2012), (BEZERRA et al., 2014),(XU et al., 2014). Após a aplicação desses mecanismos de redundância, a disponibilidade encontrada foi de 0,994559.

Tabela 7.11: Parâmetros de entrada consolidados do modelo cold standby

Parâmetros	Descrição	Valores
λ_n	taxa de falha do *Node*	$1/150,0142$
μ_n	taxa de reparo do *Node*	$1/0,9232$
λ_f	taxa de falha do *frontend*	$1/180,72$
μ_f	taxa de reparo do *frontend*	$1/0,96999$
δ	taxa de ativação do *Node* secundário	$1/0,01667$

Após a aplicação desse mecanismo de redundância *cold standby* saímos de uma disponibilidade de 0,988571 e um *downtime* de 100,12 horas para uma disponibilidade de 0,994559 que representa um *downtime* de 47,66 horas, isso significa uma redução no tempo de indisponibilidade de 47%.

Na próxima seção apresentaremos um estudo de caso sobre arquitetura do serviço de *Streaming* de Vídeo, implementando a redundância no componente *Node*.

7.2 Arquitetura de serviço de *Streaming* de Vídeo com redundância *Warm standby* no *Node*

Esse estudo de caso tem por objetivo implementar os resultados obtidos na análise de sensibilidade realizada no estudo de caso 1 para a arquitetura básica do serviço de *Streaming* de Vídeo com a finalidade de verificar as melhorias na disponibilidade, além de identificar os possíveis pontos que requerem melhoria na disponibilidade do sistema.

Desse modo, uma nova arquitetura é considerada a partir da inserção de mais um *Node* na arquitetura básica do serviço de *Streaming* de Vídeo. A arquitetura com redundância no *Node* tem sua infraestrutura representada pela Figura 7.9.

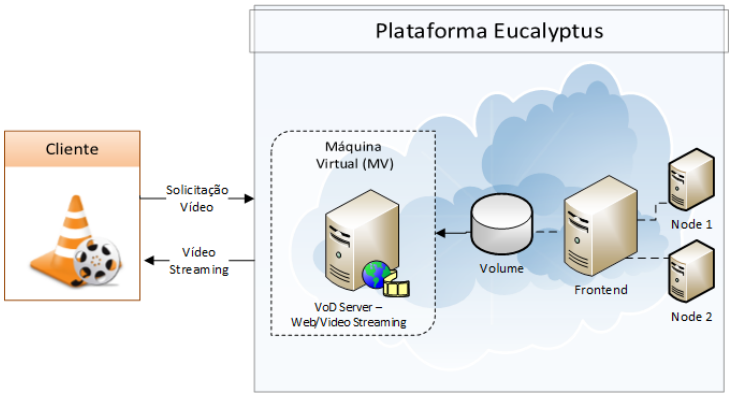


Figura 7.9: Arquitetura de *streaming* vídeo redundante

Nessa Figura, podemos observar que a infraestrutura é composta por três máquinas: uma para o *Frontend* e outras duas para os *Nodes*.

O funcionamento da arquitetura do serviço de *Streaming* de vídeo redundante é idêntico ao da arquitetura básica, a diferença é que, agora essa infraestrutura dispõe de dois *Nodes* trabalhando em redundância no modo *warm standby*.

7.2.1 Descrição do sistema

O funcionamento da arquitetura da Figura 7.9 é análogo à arquitetura do estudo de caso 1, com a inclusão de mais um *Node*.

7.2.2 Definir métricas de interesse

Essa etapa exige uma definição das métricas de interesse a respeito do que se propõe analisar. Inicialmente, analisaremos a disponibilidade e o *downtime* dessa arquitetura e para isso é necessário identificar todas as taxas de falha (MTTF) e reparo (MTTR) de todos os componentes envolvidos na arquitetura.

7.2.3 Construir modelo de alto nível

Os modelos RBD e CTMC foram utilizados para representar os subsistemas da arquitetura apresentada na Figura 7.9. Esses modelos foram combinados e constitui num modelo hierárquico. A arquitetura foi dividida em três partes: *Frontend*, Volume e Serviço que são representados pelo modelo RDB de alto nível representado na Figura 7.10.

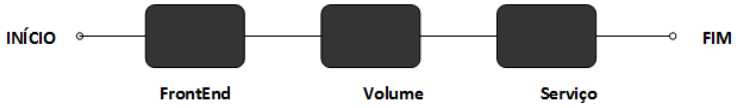


Figura 7.10: Modelo RBD para arquitetura Redundante

7.2.4 Construir modelos detalhados

O subsistema *Frontend* é representado por um modelo RBD, como mostrado na Figura 7.11. Este subsistema é composto por *Hardware* (HW), Sistema Operacional (SO), e os seguintes

Figura 7.11: Modelo RBD do *Frontend*

componentes do *Eucalyptus*: CLC - Controlador da Nuvem, CC - Controlador do *Cluster*, SC - Controlador de Armazenamento e *Walrus*. Igualmente na arquitetura básica, o subsistema do Volume é o dispositivo Network Attached Storage (NAS) responsável por armazenar a coleção de vídeos.Utilizamos dados de MTTF e MTTR para calcular a disponibilidade desse dispositivo. O subsistema serviço é refinado por um modelo CTMC (ver Figura 7.12), que permite calcular valores de disponibilidade para serem considerados no modelo RBD de alto nível da Figura 7.10. O modelo CTMC foi proposto devido à dependência entre os componentes do sistema.

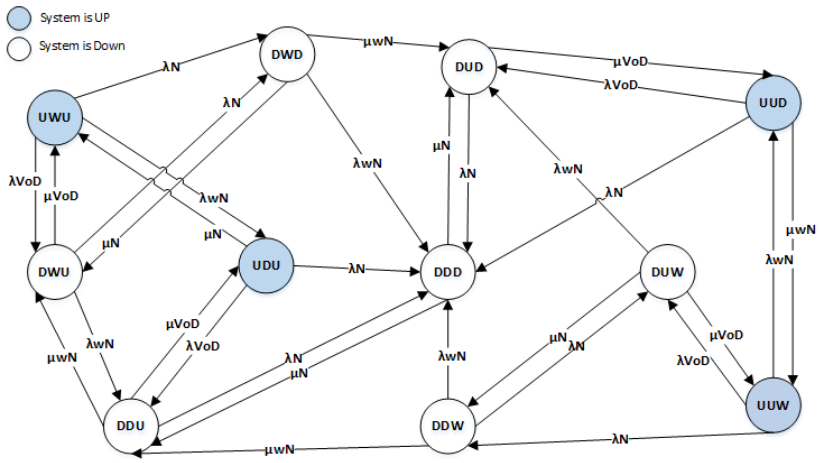


Figura 7.12: Modelo CTMC para o subsistema do serviço

Esse modelo representa a disponibilidade da infraestrutura subsistema serviço em termos da disponibilidade dos *Nodes*. O detalhe de funcionamento desse modelo, bem como os ajustes realizados na Equação de disponibilidade foram apresentados na Seção 6.4, página 76.

A Equação de fórmula fechada para calcular a disponibilidade da infraestrutura completa, A_{S_2} , pode ser obtida como demonstrado na Equação 7.9.

$$A_{S_2} = A_f \times A_v \times A_{VoD2} \tag{7.9}$$

A_f e A_v podem ser calculadas a partir do RBD da Figura 7.10, enquanto A_{VoD2} é calculado da Equação 7.10. Na Equação, A_f, A_v, e A_{VoD2} correspondem a disponibilidade do *Frontend*, Volume, e Serviço, respectivamente.

$$A_{VoD2} = \frac{\alpha(2\beta^2\alpha_1 + \beta\alpha_2\beta_1 + \alpha_3)\beta_1^2 + \alpha_4\beta_1^3}{\alpha_5(\beta^2\alpha_1(\lambda N + 2\mu N) + \beta\varphi\beta_1 + \varphi_1\beta_1^2 + \beta_7\beta^3)}, \tag{7.10}$$

7.2.5 Definição dos parâmetros de entrada

A Tabela 7.12 são apresentadas as taxas MTTF e MTTR para os elementos *Frontend* e Volume desta arquitetura. Esses valores foram obtidos a partir de (DANTAS et al., 2012),(BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009). O cálculo de métricas de dependabilidade para o módulo *Frontend* produziu um MTTF de 180,72 horas e um MTTR de 0,96 horas.

Tabela 7.12: Parâmetros de entrada para o módulo RBD do *Frontend* e Volume

Módulo	Componente	MTTF	MTTR
Frontend	HW	8760 h	100 min
	SO	2895 h	1 h
	CLC	788,4 h	1 h
	CC	788,4 h	1 h
	SC	788,4 h	1 h
	Walrus	788,4 h	1 h
Volume	Volume	100000 h	1 h

Na Tabela 7.13 são apresentadas os parâmetros para os blocos do modelo RBD para a estrutura apresentada na Figura 7.10. Os valores de MTTF e MTTR dos módulos *Frontend* e Volume são baseados em (DANTAS et al., 2012),(BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009).

Tabela 7.13: Entrada dos Parâmetros para o RBD de alto nível

Componente	MTTF	MTTR
Frontend	180,72 h	0,96999 h
Volume	100000 h	1 h
Serviço	149,987435 h	0,37903 h

A disponibilidade do módulo de serviço é calculada a partir da CTMC representada na Figura 7.12. Na Tabela 7.14 são apresentados todos os valores dos parâmetros usados para o cálculo da disponibilidade do módulo de serviço com redundância no *Node* da Figura 7.12. Os valores são baseados em análises apresentadas em (DANTAS et al., 2012),(BEZERRA et al., 2014),(KIM; MACHIDA; TRIVEDI, 2009).

Tabela 7.14: Parâmetros de entrada para o CTMC do serviço com redundância.

Parâmetros	**Descrição**	**Valores** (h^{-1})
λ_{N1}=λ_{N2}=λ_N	Tempo médio para falha do *Node*	1/481,83
λ_{wN1}=λ_{wN2}=λ_{wN}	Tempo médio para falha do *Node* em *standby*	1/578,196
μ_{N1}=μ_{N2}=μ_N	Tempo médio para reparo do *Node*	1/0,91
μ_{wN1}=μ_{wN2}=μ_{wN}	Tempo médio para reparo do *Node* em *standby*	1/0,0333
λ_{VoD}	Tempo médio para falha do serviço	1/217,779
μ_{VoD}	Tempo médio para instanciar o serviço	1/0,0275

Os valores de λ e μ representam taxas para falha e para reparo respectivamente. Os valores referentes ao tempo para falha e para reparo do NC foram obtidos em (KIM; MACHIDA; TRIVEDI, 2009). O valor de falha referente ao NC em *standby* é o tempo para falha de um *Node* ativo acrescido de 20% (DANTAS et al., 2012), já que ele encontra-se ligado, mas não operacional; por outro lado, o tempo de reparo é o tempo de ativação do componente em *standby*, que ocorre em um tempo médio de dois minutos com uso da aplicação Heartbeat (HEARTBEAT, 2014). Os tempos para falha do Serviço são resultados do CTMC obtido no estudo de caso anterior. O tempo médio para instanciar o Serviço é representado pela taxa μ_{VoD}, que resulta da Equação 7.11 a seguir:

$$\mu_{VOD} = TNODES + \mu_{in}, \tag{7.11}$$

Em que, TNODES é o tempo médio para ativar o Node, cujo valor é obtido através da aplicação *Heartbeat* (HEARTBEAT, 2014). Essa aplicação tem a função de enviar mensagens de um *Node* para o outro *Node*, detectando quando o *Node* ativo der sinais de falha (não respondendo à mensagem), para que o NC em *standby* possa ser ativado, sem que haja paralisação ou atraso crítico para o usuário final. A taxa μ_{in} foi obtida através da Equação 7.12, que representa o tempo para instanciar uma nova VM e foi abordada no primeiro estudo de caso.

$$\mu_{in} = \frac{1}{MTTI} = \frac{1}{MTVM + MTSS} \tag{7.12}$$

7.2.6 Avaliação do modelo hierárquico

Calculamos as medidas de disponibilidade usando os parâmetros de entrada mencionados no modelo hierárquico. Ambos, RBDs e CTMC foram resolvidos numericamente e realizamos a análise de dependabilidade alcançando os resultados de disponibilidade apresentados na Tabela 7.15.

Tabela 7.15: Medidas de disponibilidade e *downtime* da arquitetura redundante do serviço de *streaming* de vídeo.

Medidas	**Valores**
Disponibilidade	0,994401
Número de 9's	2,5118
Downtime Anual	49,05 horas

Assim, os correspondentes valores de disponibilidade de cada um dos subsistemas foram usados no modelo RBD de alto nível. O resultado apresentou um aumento na disponibilidade para 2,5118 noves, o que representa uma diminuição no downtime de $51,01\%$ em comparação à arquitetura básica. Apesar desse sistema apresentar uma melhoria na disponibilidade após a implementação da redundância, ainda vislumbramos futuras melhorias. Para isso aplicaremos as estratégias de análise de sensibilidade que nos auxiliarão na descoberta desse caminho.

7.2.7 Estratégias de análise de sensibilidade nos Modelos

A Equação 7.13 foi utilizada para o cálculo da disponibilidade a partir da EAS variação um por um. A variação do parâmetro foi realizada um de cada vez. A Equação 7.13 pode também ser apresentada por meio da Equação 7.14, uma vez que a disponibilidade do serviço é representada por um bloco RBD, mas possui uma cadeia de *Markov* como submodelo conforme apresentado na Seção 7.2.4.

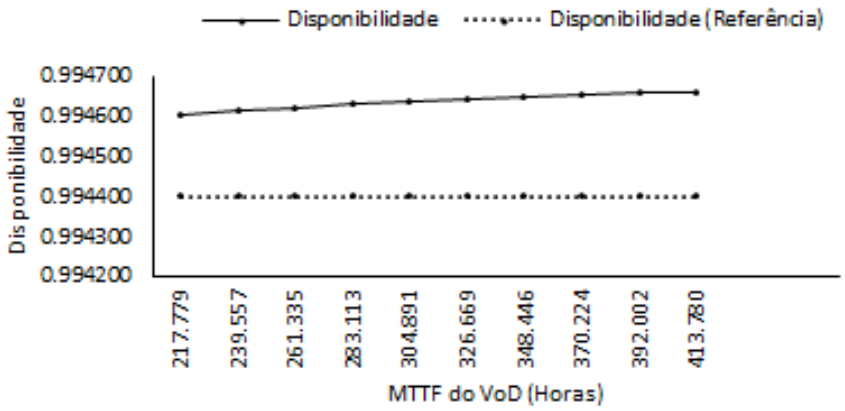
$$A_{S2} = A_f \times A_v \times A_{VoD2} \tag{7.13}$$

$$A_{S2} = A_f \times A_v \times \frac{\alpha(2\beta^2\alpha_1 + \beta\alpha_2\beta_1 + \alpha_3)\beta_1^2 + \alpha_4\beta_1^3}{\alpha_5(\beta^2\alpha_1(\lambda N + 2\mu N) + \beta\varphi\beta_1 + \varphi_1\beta_1^2 + \beta_7\beta^3)} \tag{7.14}$$

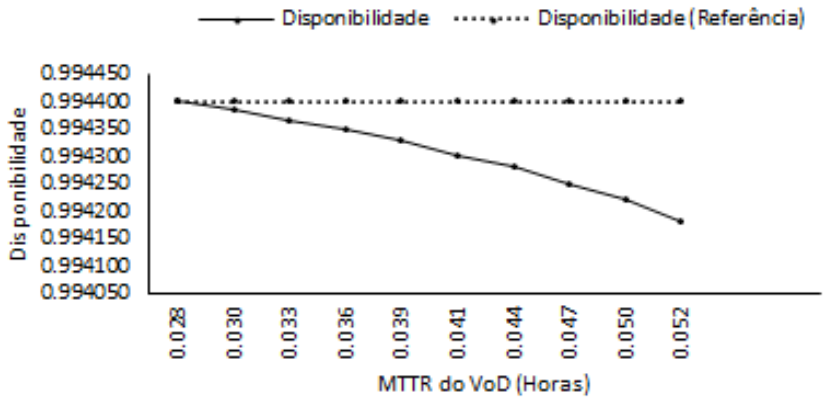
Nesta EAS em razão deste estudo de caso utilizar a mesma plataforma de nuvem *Eucalyptus* do estudo de caso 1, não seria necessário apresentar o comportamento dos componentes referente aos módulos *Frontend* e *Node*, desse modo abordaremos o comportamento apenas dos parâmetros que compõem o módulo Serviço.

A Figura 7.13 (a) e 7.13 (b) correspondem aos parâmetros relativos ao MTTF e MTTR do módulo VoD respectivamente. Os valores dos parâmetros MTTF do VoD foram variados ao longo de um intervalo de 217,77 a 413,78 horas, obtendo a disponibilidade no MTTF de 413,78 horas de 0,994663 o que significa uma redução no *downtime* de 2,29 horas em relação ao *downtime* no MTTF de 217,77 horas.

Com relação ao MTTR do VoD, os parâmetros foram variados de 0,028 a 0,052 horas. Com o MTTR de 0,052 horas obteve-se uma disponibilidade de 0,994180, o que significa que saiu de um *downtime* de 49,04 horas para 50,98 horas. Esse resultado indica que a indisponibilidade teve um acréscimo de 0.0222%.



(a) MTTF



(b) MTTR

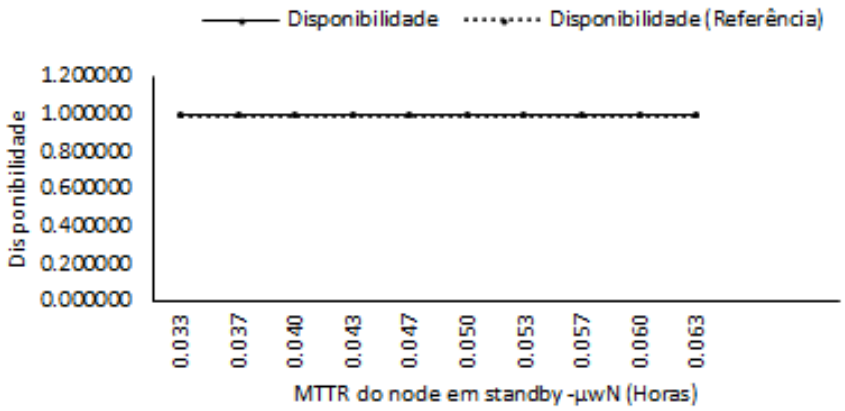
Figura 7.13: MTTF e MTTR do VoD em Horas

A Figura 7.14 (a) e 7.14 (b) correspondem às taxas para Falha e para reparo λ_{WN} e μ_{WN} do componente *Node* em *standby* respectivamente. Esses tempos são pequenos e a variação proposta foi de 10%, nos tempos para falha e para reparo, nãohavendo possibilidade de verificar o efeito dessa variação na disponibilidade do sistema. Provavelmente, se esses tempos fossem simulados em proporções maiores observaríamos um decréscimo na disponibilidade da arquitetura, na medida em que aumentássemos esses parâmetros.

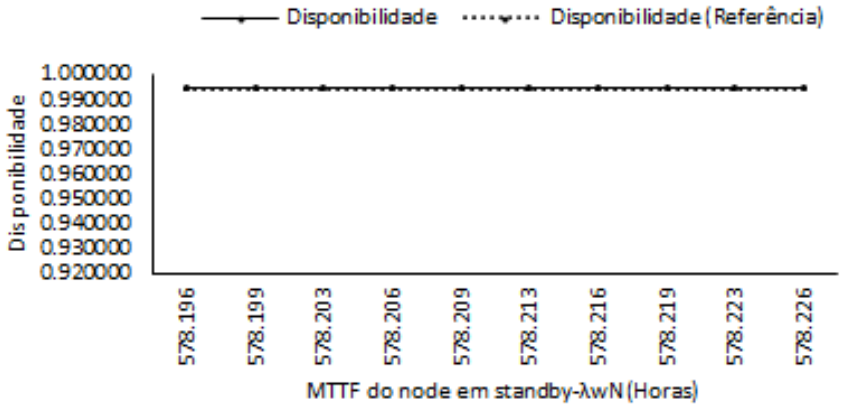
A análise realizada por meio da EAS variação um por um indica além dos parâmetros dos componentes *Frontend* e *Node*, os parâmetros do componente Serviço representados por λ_{VoD} e μ_{VoD}. Esses são parâmetros críticos do sistema de infraestrutura de nuvem com redundância.

Foi Aplicada a **estratégia de *design of experiment* (DoE)** para os modelos RBD e CTMC das Figuras 7.10 e 7.12 respectivamente. Os valores de entrada utilizados para os níveis máximos e mínimos seguiram os mesmos critérios do Estudo de Caso 01.

A Tabela 7.16 mostra o *ranking* de sensibilidade obtida pela análise de sensibilidade realizada pelo DoE. Essa Tabela destaca nas duas primeiras posições que são as taxas para falha λ_{VoD} e para reparo μ_{VoD}, que correspondem ao módulo do Serviço, como os principais



(a) MTTR



(b) MTTF

Figura 7.14: MTTF e MTTR do *Node* em *standby* em horas

parâmetros do sistema, e na sequência destacamos ainda a taxa para falha e para reparo do *Node* em standby e a taxa para falha e para reparo do componente Frontend (λ_f e μ_f) que estão entre os componentes mais relevantes do sistema.

A estratégia **diferença de percentual (DP)** é uma das estratégias que foram aplicadas aos modelos das Figuras 7.10 e 7.12, e com esse método foi possível também identificar os componentes que afetam a disponibilidade dos sistema. A Tabela 7.16 além de conter o resultado do *ranking* estratégia DoE, também apresenta o *ranking* da análise de sensibilidade em ordem descrescente para a estratégia DP.

Essa estratégia destaca os componentes μ_f, λ_f, μ_n e λ_n no topo do *ranking* de sensibilidade com relação ao impacto na disponibilidade do sistema de *Streaming* de Vídeo. Além disso, os parâmetros μ_{wN} e λ_{vol} são os componentes que ocupam uma posição de menor relevância no *ranking* de sensibilidade com relação à disponibilidade do sistema de *Streaming* de Vídeo.

O *ranking* de sensibilidade da **estratégia DASI** foi obtido por meio da Equação 7.15, para os modelos das Figuras 7.10 e 7.12.

Tabela 7.16: *Ranking* de sensibilidade obtidos através do DoE e DP

Parâmetros	**DoE -** $SS(\|A\|)$	**Parâmetros**	**DP -** $SS(\|A\|)$
λ_{VoD}	0,000270	μ_f	0,00566864
μ_{VoD}	0,000352	λ_f	0,00511872
λ_{wN}	0,000045	μ_n	0,00222435
μ_{wN}	0,000082	λ_n	0,00172078
λ_f	0,007805	μ_{VoD}	0,00092771
μ_f	0,002899	λ_{VoD}	0,00011987
λ_n	0,002794	μ_{vol}	0,00001007
μ_n	0.001335	λ_{wN}	0,00000705
λ_{vol}	0,000105	μ_{wN}	0,00000100
μ_{vol}	0,000153	λ_{vol}	0,00000100

$$DASI^*_{\theta_i}(A_{S2}) = \frac{\sum_{i=1}^{n} \frac{\partial A_f}{\partial \theta}(\frac{\theta}{A_{S2}})|_{\theta=\theta i}}{n} \times A_v \times A_{VoD2} + \quad (7.15)$$
$$Af \times \frac{\sum_{i=1}^{n} \frac{\partial A_v}{\partial \theta}(\frac{\theta}{A_{S2}})|_{\theta=\theta i}}{n} \times A_{VoD2} +$$
$$Af \times A_v \times \frac{\sum_{i=1}^{n} \frac{\partial A_{VoD2}}{\partial \theta}(\frac{\theta}{A_{S2}})|_{\theta=\theta i}}{n}$$

As expressões das derivadas parciais para o subsistema A_f, A_v estão descritas no Apêndice A. As derivadas parciais para o subsistema A_{VoD2} com relação a cada um dos parâmetros ($\frac{\partial A_{VoD2}}{\partial \mu_{VoD}}$, $\frac{\partial A_{VoD2}}{\partial \lambda_{VoD}}$, $\frac{\partial A_{VoD2}}{\partial \lambda_N}$, $\frac{\partial A_{VoD2}}{\partial \mu_N}$, $\frac{\partial A_{VoD2}}{\partial \lambda_{WN}}$ e $\frac{\partial A_{VoD2}}{\partial \mu_{WN}}$) do modelo CTMC não são descritas aqui, mas o leitor pode vê-las no Apêndice A.

Na Tabela 7.17 são apresentados o *ranking* da análise de sensibilidade em ordem decrescente para a estratégia DASI, com a finalidade de gerar os resultados dessa classificação. Os dados de entrada foram igualmente obtidos de acordo com os critérios utilizados no estudo de caso 01, na seção 7.1.7, página 89.

Tabela 7.17: *Ranking* de sensibilidade por meio das estratégias DASI e SDP para o sistema redundante

Parâmetro	$DASI^*_{\theta i}(\|A_{S2}\|)$	**Parâmetro**	$SDP^*_{\theta i}(\|A_{S2}\|)$
μ_f	0,005861225208	μ_f	0,005338710000
λ_f	0,005331802213	λ_f	0,005338710000
μ_n	0,002049195663	λ_N	0,001933190000
λ_n	0,001932307554	μ_N	0,001867330000
μ_{VoD}	0,000412384957	μ_{VoD}	0,000183315000
λ_{VoD}	0,000126247799	λ_{VoD}	0,000126252000
μ_{vol}	0,000017099727	λ_{Vol}	0,000009999900
λ_{vol}	0,000009999877	μ_{vol}	0,000009999900
λ_{wN}	0,000008896625	λ_{wN}	0,000008908720
μ_{wN}	0,000000117839	μ_{WN}	0,000000107137

Identificamos no topo do *ranking* da análise de sensibilidade para a estratégia DASI os componentes μ_f, λ_f, μ_n como os principais componentes candidatos a ações de melhorias do sistema. Por outro lado, o componente μ_{wN} é o de menor importância como apontado por essa

estratégia. Essa análise nos oferece a possibilidade de não calcular a estratégia de sensibilidade para apenas um parâmetro específico de configuração, o que nos permite a flexibilidade da variação de valores. Isso proporciona uma melhor resposta na disponibilidade do sistema.

Comparando o resultado obtido por meio da estratégia DASI (ver Tabela 7.17) com a da sensibilidade diferencial paramétrica (ver na Tabela 7.17), podemos observar que a estratégia DASI, apresenta um *ranking*, em que a taxa para reparo aparece em destaque na maioria dos casos em relação à estratégia sensibilidade diferencial paramétrica, exceto na penúltima posição da Tabela 7.17.

Essa particularidade também identificada, nesse estudo de caso, direciona para um diferencial dessa estratégia, em virtude de apontar primeiro a taxa para reparo do componente ao invés da taxa para falha. Registramos que na maioria dos casos, a taxa para reparo é pequena, no entanto, é importante que os administradores estejam atentos a ela, uma vez que nem sempre as empresas dispõem de mais de uma equipe de manutenção.

Aplicando a **estratégia ICD** para os modelos da Figura 7.10, o resultado do *ranking* de sensibilidade obtido para o caminho operacional, indicou o *Frontend* e o Serviço como os componentes mais relevantes do sistema, conforme Tabela 7.18.

Tabela 7.18: *Ranking* de sensibilidade ICD para o caminho operacional e com falha

Parâmetro	**Operacional**	**Parâmetro**	**Falha**
Frontend	1	Volume	1
Serviço	0,989851071	Serviço	0,957057607
Volume	0,989370762	Frontend	0,047235356

Por outro lado, o resultado do *ranking* de sensibilidade obtido para o caminho com falha, aponta o Volume e o Serviço como os componentes mais importantes do sistema, de acordo com a Tabela 7.18.

7.2.8 Identificar os componentes relevantes para as EAS

Nessa etapa, identificamos os componentes relevantes às estratégias utilizadas. Na Tabela 7.19 são apresentados o *ranking* de sensibilidade obtido após a avaliação em conjunto das estratégias: *design of experiment* - DoE, diferença percentual e DASI.

Na Tabela 7.19 não é projetado nenhum dado sobre a estratégia ICD e variação um por um. A ICD foi utilizada para analisar os componentes e não os parâmetros do modelo. Portanto, não pode ser comparada com as outras abordagens, que avaliam os parâmetros dos RBDs e do modelo CTMC na mesma Tabela. Para a estratégia variação um por um, isso aconteceu porque essa estratégia não apresenta o seu resultado no formato de um *ranking*, tornando inviável a análise em conjunto na mesma Tabela. No entanto, essas estratégias estão incluídas nas análises.

Tabela 7.19: Ranking produzido a partir da avaliação de EAS em conjunto

Parâmetro	**Valor**
μ_f	0,012013039
λ_f	0,006786262
μ_n	0,001591391
λ_n	0,001312416
μ_{vol}	0,00044402
μ_{VoD}	0,000444020
λ_{VoD}	0,000311020
λ_{wN}	0,000035967
μ_{wN}	0,000020624
λ_{vol}	0,000013017

A ICD aponta *Frontend* e Serviço (no caminho operacional), Volume e Serviço (no caminho com falha) como os principais componentes do sistema. Na Tabela 7.19 é possível identificar os componentes apontados pela estratégia ICD, *Frontend* e Serviço, estão representados na avaliação realizada em conjunto pelas estratégias por meio dos parâmetros λ_{VoD}, μ_{VoD} que correspondem ao componente Serviço e que ocupam a oitava e nona posições da Tabela 7.19. Os parâmetros λ_f e μ_f correspondem ao componente *Frontend* ocupando as duas primeiras posições do *ranking*.

Os parâmetros μ_f, λ_f e μ_n, λ_n, μ_{Vol} correspondem à avaliação realizada pelas estratégias em conjunto. Após essa análise, observa-se que o componente *Frontend* está no topo do *ranking* dessa avaliação. Esse resultado revela a necessidade de implementar mais uma máquina com a função *Frontend*, uma vez que essa arquitetura possui duas máquinas para o componente *Node* e apenas uma na função de *Frontend*. Essa indicação é interessante em razão desse componente ser o administrador dos recursos da plataforma de nuvem *Eucalyptos*.

Além disso, é importante destacar que a estratégia de análise de sensibilidade DoE, não elegeu nas primeiras posições os parâmetros referentes ao componente *Frontend*. Esses parâmetros aparecem na sexta e sétima posições do *ranking*, no entanto as estratégias DASI, DP e ICD apontam-nos como o mais relevante. Desse modo, na medida em que aplicamos mais de uma estratégia e relacionamos os seus resultados, temos oportunidade de capturar eventos dessa natureza.

Registramos ainda que o resultado da análise de sensibilidade no Estudo de caso 1, apontou o componente *Node* como o mais importante componente da infraestrutura. Desse modo, ao implementarmos mais um *Node* na arquitetura básica, saimos de uma disponibilidade de 0,988571 com *downtime* de 100,12 horas para uma disponibilidade de 0,994401 com *downtime* de 49,95 horas para uma arquitetura com dois *Nodes*.

Esse resultado ressalta uma melhoria significativa na disponibilidade e uma redução de mais de 50% no tempo de inatividade do sistema. Por outro lado, ainda identificamos que essa infraestrutura pode ser melhorada se atuarmos na sequência nos parâmetros μ_f, λ_f e μ_n, λ_n, μ_{Vol}.

7.3 Avaliação da plataforma *OpenMobster*

Esse estudo de caso tem por objetivo aplicar a metodologia proposta neste trabalho, no serviço MBaaS (*Mobile Backend-as-a-Service*), em virtude de ser um serviço emergente (ROWINSKI, 2012), (HURBEAN; FOTACHE, 2013), que faz parte do âmbito da *Mobile Cloud Computing* (MCC) (LANE, 2015). O avanço da tecnologia móvel tende a transferir suas bases de dados para centros de dados distribuídos, provendo assim serviços por meio de aplicações móveis, tais como o MBaaS. Este serviço refere-se a um modelo que possibilita aos desenvolvedores associar o *Backend* de suas aplicações ao armazenamento em nuvem (COSTA, 2015).

7.3.1 Plataforma *OpenMobster*

Os dispositivos móveis representam, hoje, uma parte essencial da rotina diária dos usuários. Os usuários podem ter uma experiência rica com muitos serviços de aplicações móveis. Os dispositivos móveis podem acessar serviços em nuvem e melhorar substancialmente o seu desempenho (FERNANDO; LOKE; RAHAYU, 2013). Além disso, os dispositivos de computação móvel e suas aplicações podem ser habilitados com o apoio de computação em nuvem para uma variedade de serviços de armazenamento e processamento de transferência de dispositivos móveis para a nuvem (FERNANDO; LOKE; RAHAYU, 2013). Isso muitas vezes é conseguido, por meio de um serviço móvel *backend* conhecido como plataforma MBaaS, principalmente para as funcionalidades de armazenamento em nuvem.

Mobile Backend-as-a-Service (MBaaS) é um modelo de serviço recentemente introduzido para atender às necessidades específicas de aplicações em Computação Móvel em Nuvem (MCC) (LANE, 2015). O MBaaS permite que os desenvolvedores possam conectar-se à infraestrutura de suas aplicações para armazenamento em nuvem. Além disso, o *MBaaS* oferece outras características tais como notificações *MBaaS* de *push* e integração com serviços de redes sociais (SAREEN, 2013), (COSTA et al., 2015).

A arquitetura da plataforma *OpenMobster* é ilustrada na Figura 7.15. Ela estabelece uma interface com clientes móveis, proporcionando-lhes serviços de *backend* que podem cobrir categorias, tais como aplicações de *e-mail*, serviços *Web* e mobilização de banco de dados.

A Plataforma *OpenMobster* é um exemplo de um robusto MBaaS de código aberto que pode ser implantado em uma infraestrutura privada (COSTA et al., 2015). O MBaaS tem por objetivo proporcionar um ambiente para ajudar a indústria de aplicações móveis, reduzindo o tempo para comercialização (COSTA et al., 2015). Dessa forma, os atributos de dependabilidade, tais como disponibilidade, confiabilidade e segurança podem ser identificados como características críticas (MATOS et al., 2012).

O ambiente MBaaS que não atende níveis satisfatórios de disponibilidade pode levar a perdas, comprometendo a credibilidade dos usuários sobre o serviço (KIM; MACHIDA; TRIVEDI, 2009). Compreender a disponibilidade de um sistema é essencial para o planejamento

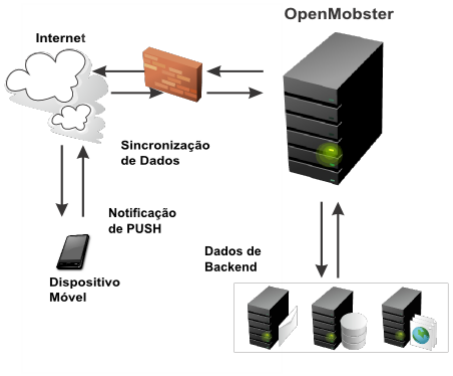


Figura 7.15: Arquitetura OpenMobster

das políticas de manutenção (KIM; MACHIDA; TRIVEDI, 2009).

Na Figura 7.16 apresentamos a arquitetura para a plataforma *MBaaS* que foi inserida em uma infraestrutura de nuvem privada configurada com a plataforma *Eucalyptus*. O dispositivo móvel do cliente se conecta ao serviço através da *internet*. Nessa Figura podemos observar em maior detalhes a infraestrutura de nuvem privada em que o serviço é prestado, principalmente por um servidor *Web*, com a máquina Virtual Java (JVM) e um banco de dados. O servidor *Web* utilizado pela plataforma é o Jboss (FLEURY; REVERBEL, 2003), que é um servidor de aplicações baseado em Java. Dessa forma é necessário que a Java *Virtual Machine* esteja instalado no servidor, pois sem ela o servidor Jboss não é capaz de ser executado.

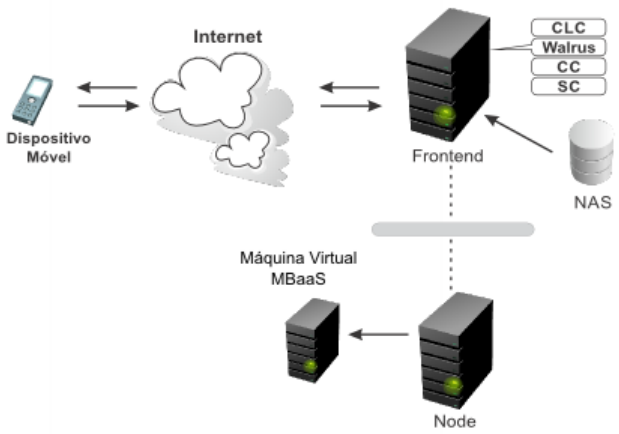


Figura 7.16: Arquitetura OpenMobster na plataforma *Eucalyptus*

O componente de banco de dados está instalado na máquina de interface e está disponível para a ligação à plataforma MBaaS. O JVM, o servidor *Web*, e a plataforma MBaaS são instalados em uma máquina virtual, que por sua vez é instanciada sobre os recursos físicos geridos pelo controlador do *Node*. O *Frontend* consiste em *hardware*, sistema operacional, um controlador de Nuvem (CLC), Controlador de *Cluster* (CC), o Controlador de Armazenamento (SC) e o *Walrus*. O NAS (*Network Attached Storage)* corresponde ao dispositivo de armazenamento do serviço. A máquina *Node* consiste em *hardware*, sistema operacional, KVM (que é o *hypervisor*

da máquina virtual), e Controlador do *Node* (NC). Não detalharemos os componentes *Frontend* e *Node* em razão de serem os mesmos componentes da infraestrutura de nuvem configurada com o *Eucalyptus* apresentados nos estudos de casos 1 e 2.

A configuração do *hardware* da máquina utilizada no serviço MBaaS na plataforma *OpenMobster* foi um Processador Intel core i7, 4 GB de RAM, HD 500 GB SATA, com o serviço do OpenMobster, e com os *softwares* *JBoss*5.1.0*GA* e o *Java*1.7.0$_2$5 instalados. As máquinas do ambiente da nuvem utilizaram o *Eucalyptus* versão 3.2.2 e o sistema operacional *CentOS* CENTOS (2014) versão 6.4. Este computador estava conectado através de uma rede *ethernet* utilizando cabeamento CAT 5e, através de um *switch* de 8 portas 10/100mbs, o sistema operacional utilizado na máquina foi o *Ubuntu* 12.04.

Após apresentação da arquitetura do *OpenMobster*, iniciaremos a aplicação da metodologia proposta para a arquitetura da Figura 7.16.

7.3.2 Definir as métricas de interesse

Essa etapa exige a definição das métricas de interesse a respeito do que se propõe analisar. Inicialmente, analisaremos a disponibilidade e o *downtime* dessa arquitetura e para isso é necessário identificar todas as taxas para falha (MTTF) e para reparo (MTTR) de todos os componentes envolvidos na arquitetura.

7.3.3 Construir modelo de alto nível

Os modelos RBD e CTMC foram utilizados para representar os subsistemas da arquitetura apresentada na Figura 7.16. Estes modelos foram combinados e constituem um modelo hierárquico. O modelo RBD representa a infraestrutura do sistema de alto nível, incluindo a nuvem privada (ver Figura 7.17). A arquitetura foi dividida em quatro subsistemas: *Frontend, Node*, NAS e Subsistema MBaaS, que são representados pelo modelo RDB de alto nível, esse modelo RBD representa a plataforma *OpenMobster*. O RBD da Figura 7.17 tem um bloco que representa o subsistema MBaaS, que foi refinado por meio de um CTMC.

Figura 7.17: Modelo RBD para arquitetura *OpenMobster*

7.3.4 Construir modelos detalhados

O subsistema *Frontend* é representado por um modelo RBD, como mostrado na Figura 7.18.

Figura 7.18: Modelo RBD do *Frontend*

Esse subsistema é composto por *hardware*, sistema operacional, e os seguintes componentes do *Eucalyptus*: CLC (Controlador da Nuvem), CC (Controlador do Cluster), SC (Controlador de Armazenamento) e Walrus.

Na Figura 7.19 apresentamos o modelo RBD que representa o *Node*. Além do sistema operacional e *hardware*, que também estão presentes no *Frontend*, o *Node* tem o *hypervisor* KVM (HU et al., 2010) e o controlador de *Node* do *Eucalyptus*. O *Node* está operacional, se e somente se, todos esses componentes estão ativos (sem falha).

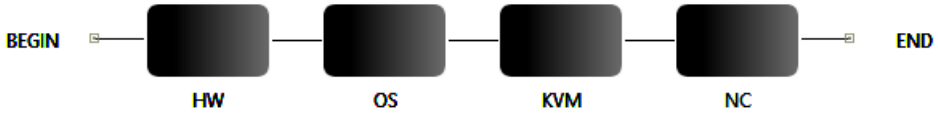


Figura 7.19: Modelo RBD do *Node*

O subsistema NAS, *Network Attached Storage* é o responsável pelo armazenamento, o qual é criado com recursos físicos da máquina *Frontend*, mais especificamente pelo controlador de armazenamento que é representado pelos MTTF e MTTR para todo o dispositivo.

O subsistema MBaaS é refinado por uma CTMC (ver Figura 7.20), o que permite calcular os valores de disponibilidade para serem considerados no modelo RBD de alto nível. Um modelo CTMC foi proposto devido à interdependência entre os componentes do sistema.

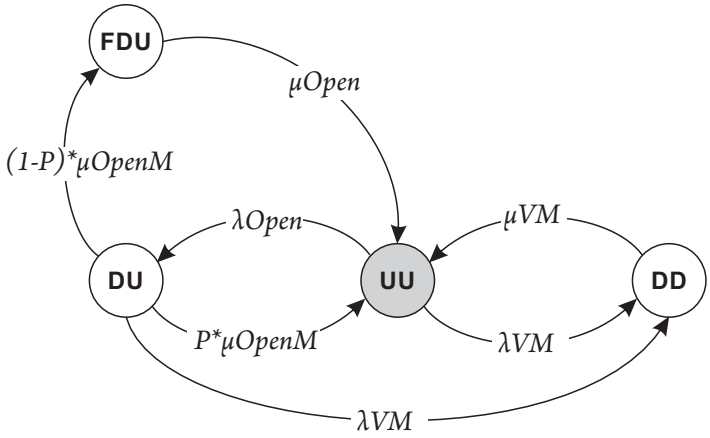


Figura 7.20: Modelo CTMC para o subsistema MBaaS

Utilizamos um modelo de CTMC para representar o comportamento detalhado do subsistema MBaaS, em razão da necessidade de representar interações dinâmicas como instanciação de máquina virtual, bem como a falha e reparação da plataforma OpenMobster. O submodelo

para o subsistema *MBaas* é representado na Figura 7.20. A partir da CTMC é possível calcular a disponibilidade desse subsistema que é então utilizado pelo modelo de alto nível RBD da arquitetura de nuvem. Esta abordagem permite ainda, a derivação da fórmula fechada para a disponibilidade do sistema no estado estacionário.

A CTMC representada na Figura 7.20 consiste nos seguintes componentes: a plataforma *OpenMobster* e a máquina Virtual (VM). A CTMC é composta por quatro estados: UU, DU, DD, e FDU. Temos usado uma notação para os estados do modelo que é baseado na condição atual de cada componente, em que (U) é o estado *UP*, ou operacional, (D) é o estado *Down*, não operacional. A seguinte sequência foi adotada, em que a primeira letra está relacionado com a plataforma *OpenMobster* e a segunda letra está relacionada com a VM. Os círculos transparentes da CTMC na Figura 7.20 representam os estados onde o sistema está inativo ou não disponível devido a uma falha. O círculo cinza indica o estado em que o sistema está ativo. A probabilidade do sistema está disponível, e a probabilidade dele estar no estado UU.

Do estado UU, os seguintes estados podem ser alcançados: a plataforma *OpenMobster* pode falhar com uma taxa λ_{Open}, atingindo o estado DU. Alternativamente, a partir do estado UU, uma VM pode falhar com uma taxa λ_{VM} e atingir o estado DD. A partir do DU, o estado UU pode ser alcançado por meio da expressão: $P \times \mu_{OpenM}$, em que μ_{OpenM} é a taxa de recuperação automática da plataforma e P é a probabilidade de sucesso para a recuperação automática. No estado DU, o estado FDU pode ser alcançado, o que representa o fracasso do processo de recuperação automática com uma taxa de $(1-P) \times \mu_{OpenM}$.

Ainda no estado DU, a VM pode falhar e o modelo atinge o estado DD com taxa para falha λ_{VM}. Do estado FDU, o estado UU pode ser alcançado com uma taxapara reparo μ_{Open} que representa a inicialização do serviço manual. No estado DD, a VM pode ser instanciada com taxa μ_{VM} que já inclui a configuração instantânea da plataforma *OpenMobster*, de modo que o sistema retorna à condição disponível representada pelo estado UU.

λ_{Open} e μ_{Open} representam as taxas para falha e para reparo da plataforma *MBaaS OpenMobster*, respectivamente. μ_{VM} corresponde ao inverso do tempo de instanciação da VM mais o tempo de inicialização da plataforma *MBaaS*. A Fórmula para a disponibilidade do subsistema *MBaaS* foi obtida a partir da CTMC na Figura 7.20. A Equação 7.16 mostra a disponibilidade da plataforma MBaaS e também pode ser utilizada para calcular o índice de análise de sensibilidade.

$$A_{MBaaS} = \frac{(P \times \mu OpenM + \beta + \lambda VM)\mu Open \times \mu VM}{\lambda VM(P \times \mu OpenM + \beta + \lambda Open + (P \times \mu OpenM + \beta + \lambda Open + \lambda VM)\mu Open)\mu VM} \quad (7.16)$$

Em que:
$\beta = (1-P) \times \mu OpenM$

Uma Equação de Fórmula fechada para calcular a disponibilidade da infraestrutura completa, AS_3 , pode ser obtida como demonstrado na Equação 7.17. A_F, A_{Node}, A_{NAS} e A_{MBaaS} podem ser calculadas a partir do RBD da Figura 7.17, enquanto A_{MBaaS} é calculado por meio da Equação 7.16. Os valores de MTTF e MTTR usados para o RBD de alto nível da Figura 7.17, são os valores de MTTF e MTTR correspondentes a cada submodelo do *Frontend, Node*, NAS e *MBaaS*. A Equação 7.17 corresponde à disponibilidade total da infraestrutura, ao passo que A_F, A_{Node}, A_{NAS} e A_{MBaaS} correspondem à disponibilidade do *Frontend, Node*, NAS e MBaaS, respectivamente.

$$A_{S_3} = A_{Frontend} \times A_{Node} \times A_{NAS} \times A_{MBaaS} \tag{7.17}$$

7.3.5 Definição dos parâmetros de entrada

Na Tabela 7.20 apresentamos os parâmetros de MTTF e MTTR para os modelos do *Frontend*, *Node*, NAS e subsistema MBaaS. Esses valores foram obtidos a partir de (DANTAS et al., 2012),(BEZERRA et al., 2014) (KIM; MACHIDA; TRIVEDI, 2009). O cálculo de métricas de dependabilidade para os módulos do *Frontend* e *Node* (ver Figura 7.18 e Figura 7.19) produz um valor de MTTF de 180,72 horas e 481,83 horas respectivamente e um valor de MTTR de 0,96 horas e 0,91 horas, respectivamente. A disponibilidade do subsistema MBaaS foi calculada a partir da CTMC ilustrado na Figura 7.20, e todos os valores necessários para resolver o CTMC estão presentes na Tabela 7.21.

Tabela 7.20: Parâmetros RBD de entrada para a arquitetura na plataforma *Eucalyptus*

Módulo	**Componente**	**MTTF** (horas)	**MTTR** (horas)
Frontend	HW	8760	1.67
	OS	2895	1
	CLC	788,4	1
	CC	788,4	1
	SC	788,4	1
	Walrus	788,4	1
Node	KVM	2990	1
	NC	788,4	1
NAS	NAS	1440	3
Subsistema MBaaS	Subsistema	206,24	0,065944

Tabela 7.21: Parâmetros de entrada do subsistema MBaaS - CTMC

Parâmetro	Descrição	Taxa(1/h)
$\lambda Open$	Taxa para falha da plataforma MBaaS OpenMobster	0,004501552
$\mu Open$	Taxa para reparo da plataforma MBaaS OpenMobster	0,270914607
μVM	Taxa para instanciação de uma VM	98.039215686
λVM	Taxa para falha da VM	0,000347222
$\mu OpenM$	Taxa para reparo automático	74,626865671
P	Probabilidade de sucesso de recuperação do processo MBaaS OpenMobster	99%

As taxas para falha e para reparo da plataforma MBaaS foram obtidos por meio da CTMC da Figura 7.20. Além disso, as demais taxas como a taxa para reparo automática do serviço, o tempo para instanciar e para reparar uma VM foram obitidos em (COSTA, 2015). Além disso, a probabilidade de sucesso do processo de recuperação foi retirado de (BAUER E.; EUSTACE, 2011).

7.3.6 Avaliação do modelo hierárquico

Calculamos as medidas de disponibilidade usando os parâmetros de entrada mencionados no modelo hierárquico. Ambos, RBD e CTMC foram resolvidos numericamente e realizamos a análise de dependabilidade alcançando os resultados de disponibilidade apresentados na Tabela 7.22. Na Tabela 7.22 apresentamos os resultados da disponibilidade do sistema, que é $99,0494\%$, ou seja, dois noves de disponibilidade. Portanto, o sistema tem um tempo de inatividade anual, *downtime*, de 83,27 horas.

Tabela 7.22: Medidas de disponibilidade e *downtime* da arquitetura OpenMobster

Métricas	Valores
Disponibilidade	0,9905601
Disponibilidade (9's)	2,022022
Downtime (h)	84,0469

7.3.7 Estratégias de análise de sensibilidade nos modelos

Aplicando a **estratégia variação um por um**, inicialmente utlizamos a Equação 7.18 para o cálculo da disponibilidade a partir da variação dos parâmetros de entrada da arquitetura do sistema. A Equação 7.18 pode também ser apresentada por meio da Equação 7.19, uma vezque a disponibilidade do subsistema MBaaS (serviço) é representada por um bloco RBD, mas possui uma cadeia de *Markov* como submodelo que foi apresentada na Seção 7.3.4.

$$A_{S_3} = A_{Frontend} \times A_{Node} \times A_{NAS} \times A_{MBaaS} \tag{7.18}$$

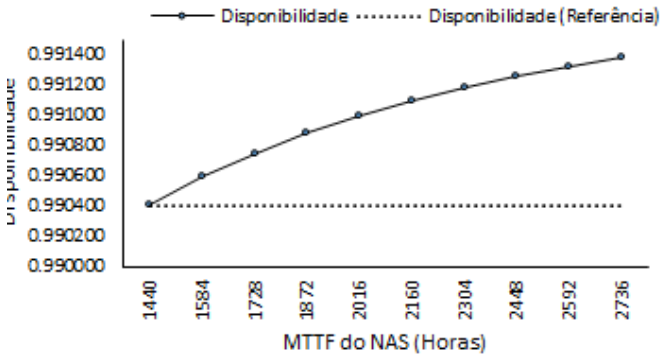
$$A_{S_3} = A_{Frontend} \times A_{Node} \times A_{NAS} \times \frac{(P \times \mu_OpenM + \beta + \lambda VM)\mu Open \times \mu VM}{\lambda VM(P \times \mu OpenM + \beta + \lambda Open + (P \times \mu OpenM + \beta + \lambda Open + \lambda VM)\mu Open)\mu VM} \tag{7.19}$$

Em que:
$\beta = (1-P) \times \mu OpenM.$

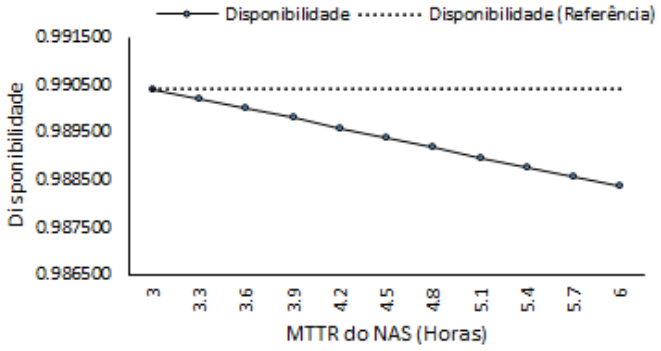
Igualmente realizado no Estudo de caso 2, não apresentaremos a análise realizada pela EAS variação um por um, para os parâmetros do *Frontend*, Node e Máquina Virtual, uma vez que já foram abordados no Estudo de caso 1. Avaliaremos os comportamentos dos módulos correspondentes ao NAS e ao subsistema MBaaS, exceto a VM.

Nas Figuras 7.21 (a) e 7.21 (b) apresentamos a variação dos parâmetros MTTF e MTTR relacionados com o componente NAS, respectivamente. O MTTF λ_{NAS} do subsistema NAS foi variado num intervalo de 1440 a 2736 horas. Os resultados da disponibilidade desse parâmetro começam em 0,990405601 e terminam em 0,991381901. Esse resultado produz a redução no *downtime* de 84.04 para 75,49 horas, representando uma redução de 10,17% ao ano.

O MTTR do subsistema NAS (μ_{NAS}) sofreu uma variação de 3 a 6 horas, nessa variação observamos um aumento no *downtime* de 84,04 horas para 102,04 horas. Esse resultado destaca a importância de recuperar o sistema no menor tempo possível, a fim de minimizar o impacto na disponibilidade do sistema.



(a) MTTF



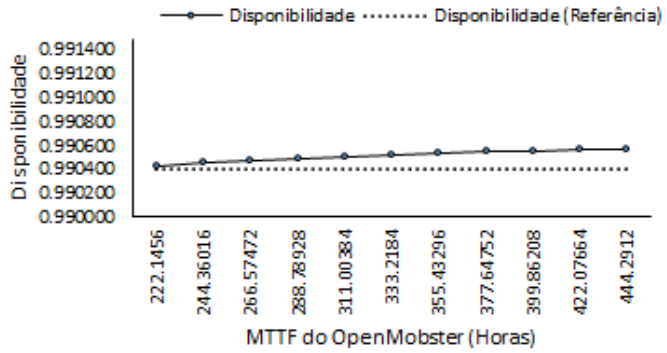
(b) MTTR

Figura 7.21: MTTF e MTTR do *NAS*

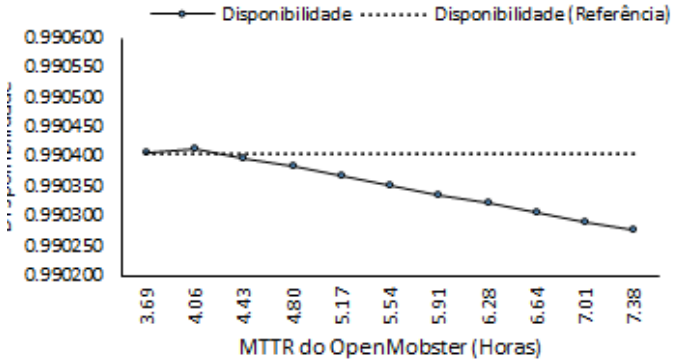
As Figuras 7.22 (a) e 7.22 (b) apresentamos a variação dos parâmetros relacionados com a plataforma *OpenMobster*. O intervalo de variação do parâmetro $\lambda Open$ do subsistema *OpenMobster* está entre 221,14 a 444,29 horas. Ao atingir a taxa de falha de 444,49 horas teremos um *downtime* de 82,56 horas, o que representa uma redução de 1,59% no *downtime* do sistema em relação ao *downtime* inicial. Observamos que a influência do $\lambda Open$ é pequena na disponibilidade do sistema, mas mesmo assim produz uma redução no *downtime* ao longo do tempo de 0,12 horas em média.

O MTTR, $\mu Open$, do subsistema *OpenMobster* sofreu uma variação de $3,69$ a $7,38$ horas, nessa variação observamos um aumento no *downtime* de 83,84 horas para 85,18 horas, isso ocorre, em razão de estarmos aumentando o tempo de recuperação do sistema, o que afeta diretamente o aumento no tempo da indisponibilidade.

Na Figura 7.23 apresentamos a variação do parâmetro relacionado com o reparo automático da plataforma *OpenMobster* ($\mu OpenM$). Na medida em que variamos esse parâmetro, a disponibilidade inicia em 0,990405601 com *downtime* de 84,04 horas e atinge a disponibilidade de 0,990255578 com *downtime* de 85,36 horas. Esse decréscimo na disponibilidade ocorre em razão do aumento no tempo de recuperação desse componente.



(a) MTTF



(b) MTTR

Figura 7.22: MTTF e MTTR do $\mu Open$ em horas

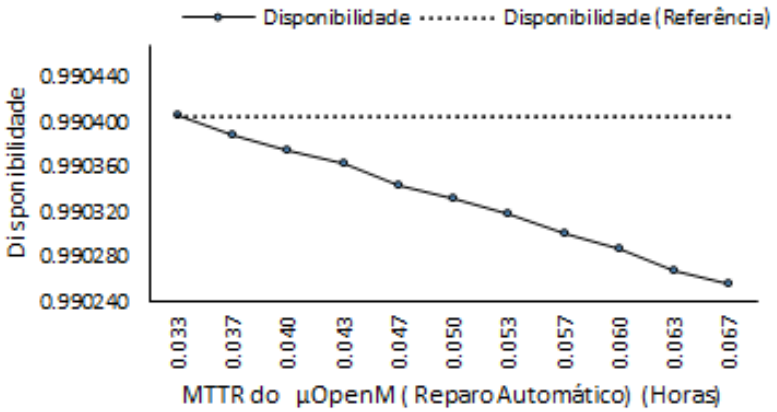


Figura 7.23: MTTF do $\mu OpenM$ em horas

A análise de sensibilidade realizada pela estratégia variação um por um para o componente NAS, indica um comportamento similar aos componentes *Frontend* e Máquina Virtual (VM) conforme apresentado no Estudo de caso 1. Ou seja, a variação dos parâmetros de taxa para falha e para reparo tem um impacto direto na disponibilidade do sistema. Em relação aos parâmetros do serviço representados pelo subsistema *MBaaS*, $\lambda Open$, $\mu Open$ e $\mu OpenM$

possuem um impacto relevante na disponibilidade desse subsistema.

Foi aplicada a estratégia de **design of experiment** para o modelo hierárquico das Figuras 7.17 e 7.20. Devido ao número de fatores excederem a quantidade de 5 (cinco) componentes, conforme orientação de (MATHEWS, 2005) adotamos o fatorial fracional, com o objetivo de investigar o efeito e relevância dos fatores escolhidos. Onze (11) fatores e dois (2) níveis foram utilizados para realizar análise por meio da estratégia DoE. Os valores adotados como entrada na ferramenta *Minitab* para essa estratégia seguiram os mesmos critérios adotados nos Estudos de caso 1 e 2, nas seções 7.1.7 e 7.2.7 nas páginas 89 e 103 respectivamente.

Na Tabela 7.23 apresentamos o *ranking* de sensibilidade obtido pela estratégia de sensibilidade DoE. Essa Tabela destaca na primeira e terceira posições as taxas para falha e para reparo do *Frontend* como os principais parâmetros do sistema. Na sequência destacamos ainda, a taxa para falha representada pelo parâmetro $\lambda Open$ que corresponde a taxa para falha da plataforma MBaaS *OpenMobster*, e a taxa para reparo do componente *Node*.

Tabela 7.23: Ranking de sensibilidade obtidos por meio das estratégias DoE e DP

Parâmetros	**DoE -** $SS(\|A\|)$	**Parâmetros**	**DP-**$SS(\|A\|)$
$\lambda_{Frontend}$	0,006282	$\mu_{Frontend}$	0,005669052
$\lambda Open$	0,004378	$\lambda_{Frontend}$	0,005118432
$\mu_{Frontend}$	0,003687	μ_{Node}	0,002257944
μ_{Node}	0,002749	μ_{NAS}	0,002079002
μ_{NAS}	0,002179	λ_{Node}	0,001677702
λVM	0,001515	λ_{NAS}	0,001395495
$\mu Open$	0,001368	$\lambda Open$	0,000214951
λ_{NAS}	0,001250	$\mu Open$	0,000103910
λ_{Node}	0,001121	$\mu OpenM$	0,000089700
μVM	0,000807	μVM	0,000008025
$\mu OpenM$	0,000781	λVM	0,000001218

Foi aplicado a estratégia **diferença percentual** no modelo da plataforma *OpenMobster*, obtemos o *ranking* de sensibilidade apresentado na Tabela 7.23.

Essa estratégia destaca os parâmetros dos componentes $\mu_{Frontend}$, $\lambda_{Frontend}$, μ_{Node} nas três primeiras posições e λ_{Node} no topo do *ranking* de sensibilidade com relação ao impacto na disponibilidade da plataforma *OpenMobster*. Esse resultado é similar ao obtido pela estratégia DoE com $\lambda_{Frontend}$, $\mu_{Frontend}$ na primeira e terceira posições do *ranking*. Além disso, o parâmetro μ_{Vm} ocupa a penúltima posição de importância nas duas estratégias com relação à disponibilidade da plataforma *OpenMobster*.

Na aplicação da **estratégia DASI** foi utilizada a Equação 7.20, para obtermos o índice de sensibilidade DASI para a arquitetura da Figura 7.16, vide página 110, modelada por meio das Figuras 7.17 e 7.20, vide páginas 111 e 112. Os resultados da estratégia $DASI^{*}_{\theta}$ (A) foram calculados por meio das ferramentas *Mercury* (SILVA et al., June 22-25,2015. Rio de Janeiro,RJ,Brazil) e *Wolfram Mathematica* (MATHEMATICA, 2012).

$$DASI^{*}_{\theta_i}(A_{S3}) = \frac{\sum_{i=1}^{n} \frac{\partial A_{Frontend}}{\partial \theta}(\frac{\theta}{A_{S3}})|_{\theta=\theta i}}{n} \times A_{Node} \times A_{NAS} \times A_{MBaaS} + A_{Frontend} \times \frac{\sum_{i=1}^{n} \frac{\partial A_{Node}}{\partial \theta}(\frac{\theta}{A_{S3}})|_{\theta=\theta i}}{n} \times A_{NAS} \times A_{MBaaS} + A_{Frontend} \times A_{Node} \times \frac{\sum_{i=1}^{n} \frac{\partial A_{NAS}}{\partial \theta}(\frac{\theta}{A_{S3}})|_{\theta=\theta i}}{n} \times A_{MBaaS} A_{Frontend} \times A_{Node} \times A_{NAS} \frac{\sum_{i=1}^{n} \frac{\partial A_{MBaaS}}{\partial \theta}(\frac{\theta}{A_{S3}})|_{\theta=\theta i}}{n} \quad (7.20)$$

A correspondente expressão para a A_{MBaaS} é representada por meio da Equação 7.16 apresentada na seção 7.3.4, página 113. Para essa análise, variamos cada parâmetro de acordo com os critérios estabelecidos na Seção 4.2, página 49.

Na Tabela 7.24 apresentamos o *ranking* da análise de sensibilidade em ordem decrescente para a estratégia DASI.

Tabela 7.24: Ranking de sensibilidade para as estratégias DASI e SDP para o *OpenMobester*

Parâmetro	$DASI^{*}_{\theta i}(\lvert A_{S3}\rvert)$	**Parâmetro**	$SDP^{*}_{\theta i}(\lvert A_{S3}\rvert)$
λ_{Open}	0,0399362314	$\mu_{Frontend}$	0,005338710
$\mu_{Frontend}$	0,0058612220	$\lambda_{Frontend}$	0,005338710
μ_{NAS}	0,0022851681	μ_{NAS}	0,002079000
λ_{NAS}	0,0020779418	λ_{NAS}	0,002079000
μ_{Node}	0,0020721550	μ_{Node}	0,001885070
λ_{Node}	0,0018841998	λ_{Node}	0,001885070
μ_{Open}	0,0006840823	λ_{Open}	0,000226429
μ_{OpenM}	0,0002301356	μ_{OpenM}	0,000224164
$\lambda_{Frontend}$	0,0000563777	μ_{Open}	0,000166122
λ_{VM}	0,0000035400	μ_{VM}	0,0000035411
μ_{VM}	0,0000000467	λ_{VM}	0,0000035400

Identificamos no topo do *ranking* da estratégia DASI os componentes λ_{Open}, $\mu_{Frontend}$ e μ_{NAS} como os principais componentes candidatos a ações de melhorias do sistema. Por outro lado, o componente μ_{VM} é o de menor importância apontado por ela. Essa estratégia nos traz a possibilidade de não calcular apenas o índice de sensibilidade para um parâmetro específico da configuração, ela nos permite a flexibilidade da variação de valores, o que favorece uma melhor resposta na disponibilidade do sistema.

Realizando uma comparação da estratégia DASI com os resultados obtidos a partir do *ranking* pela estratégia SDP, apresentada na Tabela 7.24. Observamos que o *ranking* obtido pela estratégia DASI é aproximado ao obtido pela estratégia SDP, conforme Tabela 7.24, no entanto com pequenas diferenças. Isso indica que a estratégia DASI obteve um resultado coerente em relação à estratégia SDP, desse modo o resultado indica que a alteração proposta no método proporcionou resultados satisfatórios. Praticamente os resultados são equivalentes, no entanto,

na estratégia DASI, a taxa para reparo dos componentes aparecem antes da taxa para falha, com exceção da taxa para falha relativa ao *OpenMobster* ($\lambda Open$) e a máquina Virtual (λVM) que apareceram antes da taxa para reparo. Esse evento é interessante em virtude desse tempo ser pequeno e na maioria das vezes dispomos de apenas uma equipe de manutenção para toda a infraestrutura.

O $\lambda Open$ foi capturado como o primeiro parâmetro do *ranking* na estratágia DASI esse resultado, quando comparado ao resultado da análise diferencial paramétrica, é diferente, no entanto a estratégia DASI mostra a relevância desse parâmetro, visto que ele é responsável pelo funcionamento da plataforma OpenMobster e o não funcionamento do mesmo implica na paralisação do serviço.

Aplicando a **estratégia ICD**, obtivemos o resultado do *ranking* de sensibilidade para o caminho operacional, que pode ser observado na Tabela 7.25, indicando que os subsistemas *Frontend* e NAS são os componentes mais relevantes do sistema. Por outro lado, o resultado do *ranking* de sensibilidade para o caminho com falha, indicou que o subsistema *Frontend* e MBaaS são os componentes mais importantes desse caminho. O componente *Node* é o de menor relevância em ambos os caminhos.

Tabela 7.25: Ranking de sensibilidade ICD para o caminho operacional e com falha

Parâmetro	Operacional	**Parâmetro**	Com falha
Frontend	1	*Frontend*	1
NAS	1	MBaaS	0,5957188
MBaaS	1	NAS	0,3881483
Node	0,9965398	*Node*	0,9965398

7.3.8 Identificar os componentes relevantes para as EAS

Nessa etapa, identificaremos os componentes comuns às estratégias utilizadas neste estudo de caso. Na Tabela 7.26 apresentamos o *ranking* resultante das combinações das estratégias *design of experiment*, DASI e DP.

Tabela 7.26: Ranking produzido a partir da avaliação de EAS em conjunto

Parâmetro	**Valor**
$\lambda Open$	0,042155939
$\lambda_{Frontend}$	0,00884748
$\mu_{Frontend}$	0,005690662
μ_{Node}	0,001854329
μ_{NAS}	0,001717273
λ_{NAS}	0,000908318
λ_{Node}	0,000774129
$\mu Open$	0,000306143
λVM	0,000252965
$\mu OpenM$	0,000109734
μVM	0,000081507

Na Tabela 7.26 não foi projetado nenhum resultado direto sobre a estratégia ICD e variação um por um, os motivos foram explicados nos estudos de casos anteriores. No entanto, enfatizamos que elas participaram da análise. A estratégia ICD aponta os componentes *Frontend* e NAS, no caminho operacional, *Frontend* e MBaaS, no caminho com falha, como os principais componentes do sistema.

Na Tabela 7.26 é possível identificar que os componentes apontados pela estratégia ICD estão representados por meio dos parâmetros $\lambda Open$ que corresponde ao subsistema MBaaS, ocupando a primeira posição do *ranking*, os parâmetros $\lambda_{Frontend}$ e $\mu_{Frontend}$ que correspondem ao módulo do *Frontend*, ocupando as segunda e terceira posições do *ranking*, λ_{NAS} e μ_{NAS} que correspondem ao módulo NAS, ocupando a quinta e a sexta posições da classificação.

Esses parâmetros correspondem à análise realizada por mais de uma estratégia de análise de sensibilidade. Todos estes parâmetros têm um impacto direto na disponibilidade do sistema. Esse resultado proporciona um nível de confiança alto para atuar sobre esses componentes com precisão, pois reflete o resultado da avaliação de mais de uma EAS. Para os componentes encontrados sugerimos que tenham uma atenção especial. como por exemplo, a aplicação de mecanismos de redundância, com o objetivo de melhorar a disponibilidade da arquitetura do sistema.

Desse modo, após a aplicação da análise de sensibilidade realizada, identificamos que os componentes *Frontend*, *Node* e NAS causam o maior impacto na disponibilidade da plataforma *OpenMobster*. Portanto, aplicaremos o mecanismo de redundância, ativo-ativo no *Frontend*, e avaliaremos a disponibilidade após a implementação da redundância.

Na Figura 7.24 apresentamos o modelo SPN adotado para estimar a disponibilidade de um sistema com redundância ativo-ativo. Este modelo foi baseado na cadeia de *Markov* apresentada em (BAUER; ADAMS; EUSTACE, 2011).

Nessa configuração, consideramos que duas infraestruturas de nuvem HW1 e HW2 estão executando a plataforma *OpenMobster* em paralelo e simultaneamente. Cada infraestrutura HW1 e HW2 pode ser representada pelo modelo RBD (ver Figura 7.17) que contém os seguintes

componentes: *Frontend*, *Node*, NAS e o subsistema MBaaS. Se uma infraestrutura falhar ao fornecer as funcionalidades do serviço MBaaS, a outra infraeestrutura pode lidar com a carga sem perda de dados.

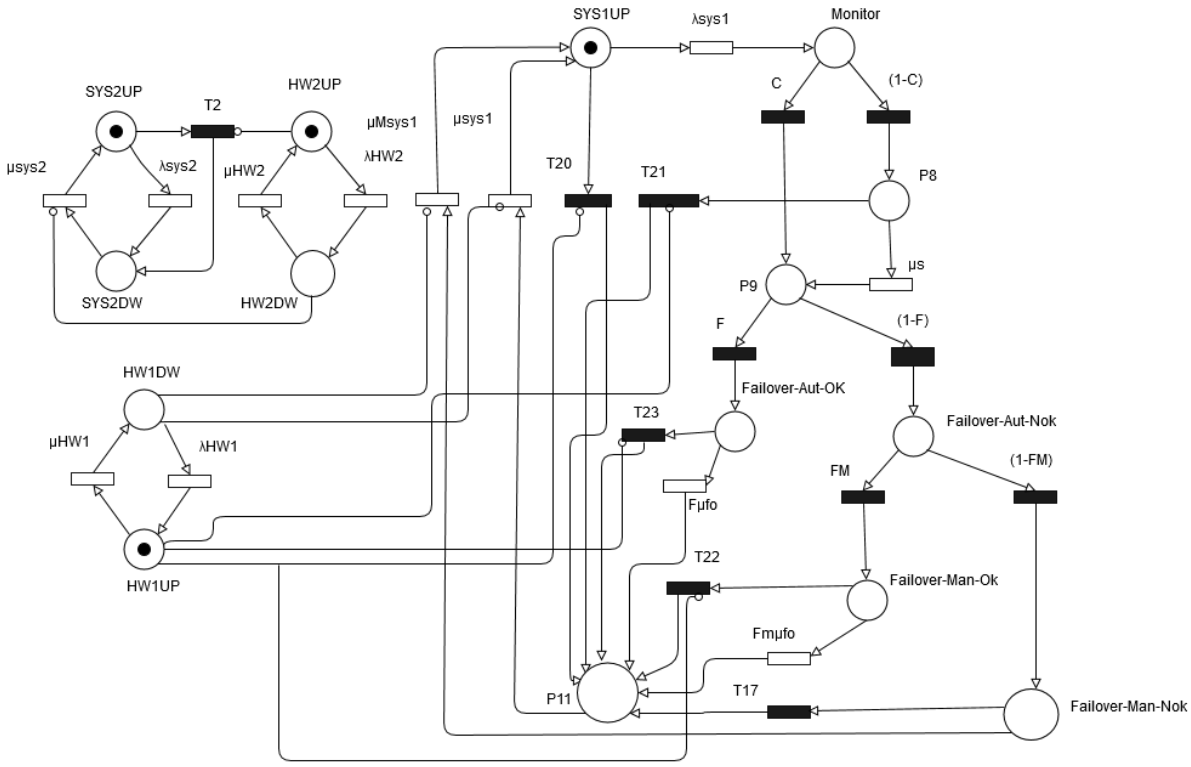


Figura 7.24: Modelo ativo-ativo para a plataforma *OpenMobster*

O funcionamento desse modelo foi detalhado na seção 6.3, na página 71 para o serviço Streaming de Vídeo. Aqui o serviço a ser considerado é o MBaaS na plataforma *OpenMobster*. Alguns lugares e transições mudaram de nome, bem como a forma de encontrar alguns resultados para as transições, por isso, para facilitar a compreensão apresentaremos novamente o funcionamento desse modelo, considerando as alterações realizadas. Os *tokens* nos lugares SYS2UP e SYS1UP indicam os estados onde o serviço está operacional nos *hardwares* HW2 e HW1, respectivamente. Os lugares SYS2DW e SYS1DW indicam estados com defeito para o serviço.

O lugare SYS2UP correponde à aplicação 2, e quando essa aplicação 2 falha, ocorre um evento relacionado à transição λ_{sys2}, que alcança o lugar SYS2DW. Isso significa que a aplicação não está operacional. Após o evento de falha da aplicação 2 podemos iniciar o processo de recuperação da mesma através da transição μ_{sys2}. No entanto, para existir a condição de recuperação da aplicação, é preciso que o Hardware 2 (HW2) esteja operacional, ou seja, o lugar HW2UP esteja com uma marcação. Caso ocorra a falha no HW2, através do evento relacionado à transição λ_{HW2} a aplicação para de funcionar automatataticamente sem tempo associado à sua paralisação, levando ao lugar HW2DW. A recuperação do hardware 2 ocorre através do evento relacionado a transição μ_{HW2}, levando-o novamente ao estado operacional HW2UP. A transição

T2 faz o controle para que o evento de falha da aplicação para que o mesmo não ocasione falha no *hardware*.

No *hardware* HW1 funciona o serviço 1, o mesmo para de funcionar, a carga de trabalho gerada por esta aplicação é absorvida pelo *hardware* 2. A falha da aplicação é representada pelo evento da transição λ_{sys1}, que pode tomar dois caminhos, sendo eles um caminho coberto e outro não coberto. O caminho coberto significa que o evento ocorreu com sucesso e que o monitoramento do serviço é feito de forma automática. Este evento é representado pela transição C. O caminho não coberto significa que não houve sucesso na cobertura e que o monitoramento acontecerá de forma manual, sendo representado pela transição $1-C$.

Continuando a análise do modelo, seguindo o caminho do monitoramento autómátco, quando a aplicação SYS1UP falha a mesma é representada pelo evento λ_{sys1}, que alcança o lugar **Monitor**. A transição C representa que o evento ocorre com 90% de sucesso, e nesse momento isso quer dizer que o processo de *failover* automático é iniciado com ativação da transição F (o percentual de sucesso do *failover* é de 99%). Esse processo assume que 50% do tráfego antes da falha já foi processado pela máquina em que ocorreu a falha e que os outros 50% serão direcionados para a máquina que está operacional.

Isso ocorre no primeiro momento da falha, o lugar $Failover-Aut-OK$ representa este estado transitório. No entanto, após algum tempo, 100% do tráfego da máquina com falha, no caso a aplicação 1, será direcionada para a máquina que está operacional, este evento será representado através da ativação da transição F que é o tempo de detecção do failover automático, atigindo portanto, o lugar P11 com uma das máquinas funcionando com 100% da carga até que recuperemos a máquina com falha na aplicação. A recuperação da aplicação com falha é realizada através da ativação da transição μ_{sys1} chegando ao lugar SYS1UP e restabelecendo a aplicação 1.

Existe a possibilidade do *failover* automático não funcionar em 1% (BAUER; ADAMS; EUSTACE, 2011) dos casos. Nesse caso o chaveamento da carga de trabalho ocorrerá de forma manual, e a transição $1-F$ é ativada chegando no lugar $Failover-Aut-NOK$ e nesse momento atingimos a transição FM indicando que o processo iniciará de forma manual. Ao atingir o lugar $Failover-Man-Ok$ teremos um estado momentâneo em que a carga de trabalho está sendo migrada de uma máquina para outra máquina. Ao acionarmos a transição $Fm\mu_{fo}$ que representa o tempo de ativação manual do processo de *failover* ao chegar no lugar P11 todo o tráfego já terá sido absorvido pela outra máquina e permanecerá nesse lugar até que a aplicação seja restabelecida. O restabelecimento da aplicação acontecerá com ativação da transição μ_{sys1}.

Assim como o processo de *failover* automático, o processo de *failover* manual pode falhar em 1% (BAUER; ADAMS; EUSTACE, 2011), no modelo quando esse evento ocorre, temos a ativação da transição $1-FM$. Nesse caso quando isso ocorre o lugar $Failover-Man-NoK$ é atingindo e na sequencia a transição $\mu_M sys1$ é acionada, e a aplicação será recuperada de forma manual, levando um tempo maior do que no processo automático. As transições T17,T21,T22 e T23 garantem o funcionamento do *Hardware 1* em estado operacional quando estão acontecendo

os processos de monitoramento coberto ou não, *failover* automático ou manual, manutenção automática ou manual. A Equação da disponibilidade para este modelo da Figura 7.24 é obtido através da Equação 7.21 e está presente na seção 6.3, no entanto para facilitar a compreensão apresentaremos novamente.

$$A = P(\#SYS2UP = 1 \quad OR \quad \#SYS1UP = 1) \tag{7.21}$$

Através do modelo da Figura 7.24, é possível calcular a disponibilidade orientada a capacidade (COA). A disponibilidade orientada a capacidade é calculada a partir deste modelo SPN por meio da Equação 7.22. A Equação 7.22 destina-se a capturar a capacidade computacional, quando os dois servidores estão trabalhando simultaneamente e quando ocorre uma falha em um dos dois servidores.

$$COA = \frac{P(\#SYS2UP = 1 \quad AND \quad \#SYS1UP = 1) \times 2 + \alpha + \theta}{2} \tag{7.22}$$

Em que:
$\alpha = P(\#SYS2UP = 1 \quad AND \quad \#SYS1UP = 0)$;
$\theta = P(\#SYS2UP = 0 \quad AND \quad \#SYS1UP = 1)$.

Na Tabela 7.27 apresentamos os valores das transições usadas no modelo de redundância ativo-ativo. Esses valores foram obtidos de (BAUER; ADAMS; EUSTACE, 2011) e foram usados no modelo de rede de Petri para o sistema MBaaS *OpenMobster*, e posteriormente para todo o sistema. Diferente do modelo ativo-ativo apresentado no Capitulo 6, página 71, para o serviço de *streaming* de vídeo, as transições mu_{HW1}, μ_{HW2}, λ_{HW2} e λ_{HW1} do modelo SPN do sistema MBaaS *OpenMobster* foram utilizadas os valores MTTF e MTTR resultantes do cálculo da disponibilidade desses três componentes: *Frontend, Node*,*NAS*, conectado em um modelo RBD, ou seja, $A_{Frontend} \times A_{Node} \times A_{NAS}$. Para isso, o cálculo usou os parâmetros de entrada da Tabela 7.20. Nas transições λ_{sys1}, λ_{sys2}, μ_{sys1}, μ_{sys2} do modelo SPN utilizaram os valores MTTF e MTTR resultantes do subsistema MBaaS representado pelo modelo CTMC da Figura 7.20.

Para essa configuração de parâmetros apresentados encontramos um valor de 99,99402% (ver Tabela 7.28) para a disponibilidade da plataforma *OpenMobster* MBaaS hospedada na nuvem usando o mecanismo de redundância ativo-ativo. Essa disponibilidade corresponde a cerca de 0,52 horas de inatividade em um ano. A aplicação do mecanismo de redundância ativo-ativo reduziu o *downtime* em torno de 99% em comparação com o sistema sem redundância. Esse resultado destaca a adequação da análise de sensibilidade proposta para a identificação de soluções eficazes para melhorar esse sistema.

Para esse modelo ativo-ativo é possível calcular a disponibilidade orientada a capacidade, COA, indicando o poder computacional do sistema. Nesse caso, temos duas máquinas que

Tabela 7.27: Atributos das transições do modelo ativo-ativo em rede de Petri.

Transição	Tempo (horas)	Peso	Prioridade	Tipo	Concorrência
$C = FM$	-	0,90	1	Imediata	-
$1-C = 1-FM$	-	0,10	1	Imediata	-
F	-	0,99	1	Imediata	-
$1-F$	-	0,01	1	Imediata	-
$T2 = T17 = T20 = T21 = T22 = T23$	1	1	1	Imediata	-
$\mu_{sys2} = \mu_{sys1}$	0,065944	1	1	Exponencial	Exclusive Server
$\lambda_{sys1} = \lambda_{sys2}$	206,237685	1	1	Exponencial	Exclusive Server
$\lambda_{HW1} = \lambda_{HW2}$	120,43	1	1	Exponencial	Exclusive Server
$\mu_{HW1} = \mu_{HW2}$	1,1278	1	1	Exponencial	Exclusive Server
$\mu_s = \mu_{Msys1}$	0,5	1	1	Exponential	Exclusive Server
$F\mu_{fo}$	0,003	1	1	Exponencial	Exclusive Server
$Fm\mu_{fo}$	0,495	1	1	Exponencial	Exclusive Server

Tabela 7.28: Resultado da comparação entre sistemas

Métrica	Sistema com redundância	Sistema na Nuvem sem redundância
Disponibilidade (%)	99,99402	99,0494
Downtime (hours)	0,52	83,2725

trabalham em paralelo e quando uma falhar, a outra máquina assume a carga de trabalho da máquina que falhou. O valor de COA para o sistema ativo-ativo da plataforma *OpenMobster* com redundância é de 1,98005. Isso equivale a $99,0025\%$ da capacidade total do sistema. Portanto, essa é a fração esperada da carga de trabalho total que pode ser processada ao longo do tempo, mesmo na presença de falhas.

7.4 Considerações Finais

Os estudos de caso abordados neste capítulo, apresentaram aspectos relevantes relacionados à disponibilidade em ambientes de *Streaming* de Vídeo e do subsistema MBaaS da plataforma *OpenMobester*. Os resultados obtidos oferecem um aporte para que administradores desses ambientes possam analisar as melhores infraestruturas a serem adotadas do ponto de vista da disponibilidade e do *downtime*.

Inicialmente no primeiro estudo de caso, observamos que a partir de uma arquitetura básica para o serviço de *Streaming* de Vídeo na plataforma de nuvem *Eucalyptus* foi possível aplicar as estratégias de análise de sensibilidade variação um por um, diferença percentual, *design of experiments* - DoE e combiná-las com as estratégias DASI e ICD. A implementação das estratégias DASI e ICD foram importantes por que proporcionam o cálculo do índice de sensibilidade a partir de vários parâmetros de entrada e pela possibilidade de verificar os pontos de redundância considerando o sistema funcionando e o sistema não funcionando, respectivamente.

Essas estratégias conjuntamente com a modelagem hierárquica foram utilizadas para

identificar os pontos que requerem melhoria na disponibilidade desses ambientes. Diante dos pontos identificados, o mecanismo de redundância *cold standy* foi aplicada no componente *Node* da infraestrutura, visando melhorar a disponibiliddae do sistema.

Na sequência, no segundo estudo de caso, após a análise de sensibilidade realizada na arquitetura básica e com o conhecimento do componente mais importante, implementamos a redundância *warm standby* no componente indicado pelas EAS, no caso o *Node*. Verificamos que a disponibilidade do sistema atingiu o valor de 0,9944 e *downtime* de 49,05 horas.

Para finalizar, o Estudo de caso 3 proporcionou a implementação dessa metodologia num serviço diferente do serviço de *Streaming* de Vídeo, para isso foram necessários alguns ajustes nos modelos. O serviço escolhido foi o *MBaaS*, hospedado na plataforma *OpenMobster* provido por uma infraestrutura de nuvem *Eucalyptus*, em virtude da necessidade de armazenamento de dados dos dispositivos móveis na nuvem e pelo crescimento de números de usuários que usam esses dispositivos. Os ajustes realizados no modelo consistiu em usar para as transições μ_{HW1}, μ_{HW2}, λ_{HW2} e λ_{HW1} do modelo SPN do sistema MBaaS *OpenMobster* os valores MTTF e MTTR resultantes do cálculo da disponibilidade dos três componentes: *Frontend, Node*,*NAS*, conectado em um modelo RBD, ou seja, $A_{Frontend} \times A_{Node} \times A_{NAS}$. Para isso, o cálculo usou os parâmetros de entrada da Tabela 7.20.

Com relação às duas estratégias propostas DASI e ICD, complementamos que na estratégia DASI, os resultados foram próximos aos obtidos em relação à EAS diferencial paramétrica. No entanto essa estratégia ao apresentar o seu *ranking* apresentou na maioria dos casos, que a taxa para reparo aparece em destaque em detrimento da taxa para falha do componente. Esse evento é importante, em virtude desse tempo ser pequeno e, além disso as empresas disporem de apenas uma equipe de manutenção.

Ademais, a estratégia ICD possibilitou a avaliação da infraestrutura a partir da análise do comportamento do sistema quando os seus componentes estão funcionando e quando eles não estão funcionando. Isso é relevante, uma vez que podemos avaliar o sistema também na condição de falha. Além dessa estratégia referir-se ao componente, e não aos seus parâmetros individuais, possibilitando uma visão ampla do sistema.

8

Conclusões

Os sistemas computacionais, em particular, a computação em nuvem, apresentam alguns desafios que precisam ser superados, tais como o planejamento de infraestruturas de nuvens que favoreçam a manutenção dos níveis de serviços acordados entre o provedor e o cliente, bem como o planejamento das infraestruturas de modo a garantir a disponibilidade na ocorrência de eventos de falha e em atividades, quer seja de manutenção preventiva, preditiva ou de reparo. Planejar esses ambientes de modo que os critérios de qualidade sejam mantidos é uma atividade fundamental para garantir a permanência do negócio e a credibilidade do sistema perante o usuário.

As estratégias de análise de sensibilidade foram utilizadas para identificar a ocorrência de tendências de comportamento dos ambientes de infraestrutura de nuvem, visando minimizar as suas paralisações. Desse modo, esta Tese apresenta como principal contribuição a proposição e adaptação das estratégias de análise de sensibilidade DASI e ICD, respectivamente. A partir da comparação e avaliação em conjunto dessas estratégias com as existentes, sendo elas: variação um por um, o *design of experiment*, e a diferença percentual, surge um outro *ranking*.

Esse *ranking* foi obtido com a verificação da posição que cada componente ocupava no *ranking* de cada estratégia, e para essa posição foi atribuído um peso. De modo que, o *ranking* produzido teve a contribuição dessas estratégias associadas. O *ranking* produzido, por esse resultado, traz os componentes mais significativos do sistema, para os quais serão aplicadas soluções que favoreçam a performance do sistema.

O índice gerado pela estratégia DASI eliminou a dependência de um único parâmetro de entrada no cálculo do *ranking* de sensibilidade e além disso o *ranking* gerado apresentou na maioria dos casos, nas primeiras posições a taxa para reparo do componente, em detrimento da taxa para falha. Esse resultado é interessante, em razão das empresas possuirem, normalmente, apenas uma equipe de manutenção e isso pode ser um fator a ser analisado com um certo grau de atenção.

Por outro lado, o índice gerado pela estratégia ICD possibilitando identificação dos componentes críticos propondo a avaliação da arquitetura por meio de dois caminhos. Um deles é um caminho operacional, onde a métrica de interesse é avaliada quando os componentes estão

funcionando e um outro chamado de não operacional, onde é verificada a métrica de interesse considerando que os componentes falharam. Além disso, essa estratégia está relacionada, ao fato de que ela refere-se ao componente, e não aos seus parâmetros individuais, isso possibilita uma visão ampla do sistema, tornando-a relevante num processo de análise.

Os estudos de caso utilizados para validação da metodologia proposta foram relacionados aos serviços de *Streaming* Vídeo e MBaaS na plataforma *OpenMobster*, ambos hospedados na plataforma de nuvem *Eucalyptus*. A escolha desses serviços ocorreram por razões que norteam o crescimento desses no mercado mundial, pela necessidade de transferência, processamento e disponibilidade dos mesmos. Esses requisitos a computação em nuvem, suporta e proporciona para os seus usuários.

As estratégias de análise de sensibilidade DASI e ICD quando associadas às estratégias variação um por um, com a diferença percentual e com o *design of experiments* – DoE, promoveram resultados satisfatórios na identificação dos pontos que requerem melhoria na disponibilidade desses ambientes, conforme observado nos estudos de caso. Além disso, com a identificação, verificamos que a implementação dos modelos para representação de mecanismo de redundância proporcionaram um aumento na disponibiliddae do sistema e consequentemente uma redução do *downtime* dos mesmos.

A validação da metodologia por meio dos estudos de casos produziram elementos para auxiliar os administradores a atuarem com precisão sobre os parâmetros das infraestruturas, a fim de cumprir as determinações de funcionamento do sistema, perante o usuário com o mínimo de interrupção, garantido os acordos de serviços definidos. Os modelos elaborados para a representação dos mecanismos de redundância foram o *cold standby*, o *warm standby* e ativo-ativo, e auxililiaram na manutenção desses acordos.

8.1 Contribuições

Dentre as contribuições deste trabalho, destacamos a proposição e adaptação das estratégias de análise de sensibilidade, como a mais significativa dessa pesquisa. Essas são o Índice discreto médio de sensibilidade, DASI e índice de importância crítica para a disponibilidade, ICD. As estratégias de análise de sensibilidade, DASI e ICD foram utilizadas conjuntamente com as outras estratégias na construção de um índice de sensibilidade que estabelece um *ranking* com os principais componentes presentes em todas as estratégias. Esse *ranking* servirá para orientação dos gestores e projetistas na escolha dos pontos que requerem a implementação de melhorias na disponibilidade dessas infraestruturas de forma responsiva.

A proposição da estratégia ICD, possibilitou a identificação de pontos de redundância no caminho, quando o sistema está operacional e quando o mesmo não está operacional. Por outro lado, a implementação da estratégia DASI proporcionou como resultado a possibilidade da aplicação da estratégia em mais de um parâmetro de entrada, eliminando a dependência de um único parâmetro. Adicionalmente essa estratégia apresenta no seu *ranking*, em primeira posição,

as taxas para reparo dos componentes em detrimento da taxa para falha, isto foi observado na maioria dos estudos de casos analisados.

Além do estabelecimento e adaptação de estratégias, um conjunto de modelos hierárquicos foram elaborados para a avaliação da dependabilidade, mais especificamente as métricas disponibilidade e *downtime*. Esses foram elaborados para avaliação da disponibilidade de nuvens privadas, aplicadas em serviços de *Streaming* de Vídeo e *MBaaS* que permitiram a obtenção de conclusões importantes sobre esses ambientes.

Adicionalmente aos modelos hierárquicos dispomos do conjunto de modelos que foram construídos especialmente para serem usados na representação dos mecanismos para redundância com as particularidades específicas de cada mecanismo. É importante ressaltar que a proposição de um procedimento para aplicação em conjunto das estratégias de análise de sensibilidade integrada à implementação de modelos hierárquicos e modelos para representação de mecanismos de redundância, num ambiente de nuvem, é uma contribuição dessa pesquisa.

A união desses aspectos faz uma diferença na composição dessa metodologia, em detrimento aos trabalhos relacionados com essa pesquisa. Essas ações possibilitam a escolha apropriada do componente ou parâmetro para a aplicação dos modelos para representação dos mecanismos de redundância, visando a otimização da disponibilidade desses ambientes de nuvem.

8.2 Trabalhos futuros

Após a conclusão dessa pesquisa, verificou-se que a mesma pode ser ampliada por meio de propostas futuras, como por exemplo, realizar a análise de outros atributos de dependabilidade em ambientes de infraestrutura de nuvem, além da disponibilidade e *downtime* anual.

Realizar a análise de desempenho das plataformas de *Streaming* de Vídeo e *OpenMobster* com o serviço MBaaS, em um ambiente de nuvem privada, com o propósito de verificar o comportamento de VMs sobre as métricas de utilização de memória, utilização de CPU e disco, além de tipos diferentes de formatos de vídeos. Uma avaliação sobre o desempenho desses componentes, considerando os níveis diferentes de cargas de trabalho requisitadas pelo usuário podem ser desenvolvidas, uma vez que não foram analisadas nessa pesquisa. Além da proposta citada, pode-se destacar em ampliar para implementação automática das estratégias de análise de sensibilidade, apresentando resultados individuais de cada estratégia, bem como resultados consolidados das estratégias adotadas.

Uma limitação adicional desse trabalho consiste na avaliação da metodologia proposta, em uma plataforma de nuvem diferente da plataforma *Eucalyptus*, como por exemplo as plataformas de nuvens *Openstack*, *Cloudstack* e *Openbula* com o objetivo de verificar o comportamento desses outros tipos de *cloud*. Finalizando, outras estratégias de análise de sensibilidade poderiam ser investigadas no uso dos ambientes de infraestrutura de nuvem, tais como Regressão e Anova, que não foram consideradas nessa tese.

Referências Bibliográficas

ADHIKARI, V. K. et al. Measurement Study of Netflix, Hulu, and a Tale of Three CDNs. , [S.l.], 2014.

ALBUQUERQUE JÚNIOR, G. A. d. Modelagem e Avaliação de Desempenho Operacional e Ambiental em Cadeias de Suprimentos Verdes. , [S.l.], 2013.

AMARI, S. V.; PHAM, H.; MISRA, R. B. Reliability Characteristics of-out-of-Warm Standby Systems. **Reliability, IEEE Transactions on**, [S.l.], v.61, n.4, p.1007–1018, 2012.

ARAUJO, J. et al. Software aging issues on the eucalyptus cloud computing infrastructure. In: SYSTEMS, MAN, AND CYBERNETICS (SMC), 2011 IEEE INTERNATIONAL CONFERENCE ON, Anchorage. **Anais...** [S.l.: s.n.], 2011. p.1411–1416.

ARMBRUST, M. et al. **Above the clouds**: a berkeley view of cloud computing. [S.l.: s.n.], 2009.

ARMBRUST, M. et al. A view of cloud computing. **Communications of the ACM**, [S.l.], v.53, n.4, p.50–58, 2010.

AVIZIENIS, A. et al. **Fundamental concepts of dependability**. [S.l.]: University of Newcastle upon Tyne, Computing Science, 2001.

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. **Dependable and Secure Computing, IEEE Transactions on**, [S.l.], v.1, n.1, p.11–33, 2004.

BALBO, G. Introduction to stochastic Petri nets. In: **Lectures on Formal Methods and PerformanceAnalysis**. [S.l.]: Springer, 2001. p.84–155.

BARRENTINE, L. B. **An introduction to design of experiments** : a simplified approach. [S.l.]: ASQ Quality Press, 1999.

BAUER E., A. R.; EUSTACE, D. **Beyond Redundancy**: how geographic redundancy can improve service availability and reliability of computer-based systems. Hoboken, NJ, USA: John Wiley l& Sons, Inc., 2011.

BAUER, E.; ADAMS, R. **Reliability and availability of cloud computing**. [S.l.]: John Wiley & Sons, 2012.

BAUER, E.; ADAMS, R.; EUSTACE, D. **Beyond redundancy**: how geographic redundancy can improve service availability and reliability of computer-based systems. [S.l.]: John Wiley & Sons, 2011.

BEZERRA, M. C. et al. Availability modeling and analysis of a vod service for eucalyptus platform. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 2014. **Anais...** [S.l.: s.n.], 2014. p.3779–3784.

BIRNBAUM, Z. W. **On the importance of different components in a multicomponent system**. [S.l.]: DTIC Document, 1969.

BLUM, R. **Linux command line and shell scripting bible**. [S.l.]: John Wiley & Sons, 2008. v.481.

BOX, G. E. P.; WILSON, K. B. On the Experimental Attainment of Optimum Conditions. **Journal of the Royal Statistical Society. Series B (Methodological)**, [S.l.], v.13, n.1, p.1–45, 1951.

BRILHANTE, J. et al. Dependability models for Eucalyptus infrastructure clouds considering VM life-cycle. In: SYSTEMS, MAN AND CYBERNETICS (SMC), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.1336–1341.

BRUNI, A. L.; FAMÁ, R.; SIQUEIRA, J. d. O. Análise do risco na avaliação de projetos de investimento: uma aplicação do método de monte carlo. **Caderno de pesquisas em Administração**, [S.l.], v.1, n.6, p.62–74, 1998.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. M. **Cloud computing**: principles and paradigms. [S.l.]: John Wiley & Sons, 2010. v.87.

CALLOU, G. et al. An Integrated Modeling Approach to Evaluate and Optimize Data Center Sustainability, Dependability and Cost. **Energies**, [S.l.], v.7, n.1, p.238–277, 2014.

CAMPOLONGO, F. et al. Screening methods. **Wiley Series in Probability & Statistics**, [S.l.], 2000.

CAMPOLONGO, F.; SALTELLI, A. Sensitivity analysis of an environmental model: an application of different analysis methods. **Reliability Engineering & System Safety**, [S.l.], v.57, n.1, p.49–69, 1997.

CAMPOLONGO, F.; TARANTOLA, S.; SALTELLI, A. Tackling quantitatively large dimensionality problems. **Computer physics communications**, [S.l.], v.117, n.1, p.75–85, 1999.

CARDOSO, F. C. Conceitos de rede virtual privada para streaming seguro de vídeo. **Universidade São Franscisco**, [S.l.], 2010.

CENTOS. **CentOS Project**. Disponível em: `http://www.centos.org/`>. Acessado em: 10 de mar. 2016.

ČEPIN, M. **Assessment of Power System Reliability**: methods and applications. [S.l.]: Springer Science & Business Media, 2011.

CHAGANTI, P. Cloud services for your virtual infrastructure, Part 1: infrastructure-as-a-service (iaas) and eucalyptus. **IBM developerWorks (December 2009)**, [S.l.], 2009.

CHUOB, S.; POKHAREL, M.; PARK, J. S. Modeling and Analysis of Cloud Computing Availability based on Eucalyptus Platform for E-Government Data Center. In: INNOVATIVE MOBILE AND INTERNET SERVICES IN UBIQUITOUS COMPUTING (IMIS), 2011 FIFTH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.289–296.

COSTA, I. d. O. Modelos par análise de disponibilidade em uma plataforma de mobile backend as a service. , [S.l.], 2015.

COSTA, I. et al. Availability Evaluation and Sensitivity Analysis of a Mobile Backend-as-a-service Platform. **Quality and Reliability Engineering International**, [S.l.], 2015.

COUTINHO, E. et al. Elasticidade em computaçao na nuvem: uma abordagem sistemática. **XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2013)-Minicursos**, [S.l.], 2013.

DANTAS, J. et al. An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism. , [S.l.], p.1664–1669, 2012.

DANTAS, J. et al. Hierarchical Model and Sensitivity Analysis for a Cloud-Based VoD Streaming Service. In: DEPENDABLE SYSTEMS AND NETWORKS WORKSHOP, 2016 46TH ANNUAL IEEE/IFIP INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2016. p.10–16.

DARADKEH, M.; CHURCHER, C.; MCKINNON, A. Interactive Visualization Techniques for Exploring Model Sensitivity. In: NEW ZEALAND COMPUTER SCIENCE RESEARCH CONFERENCE. **Proceedings...** [S.l.: s.n.], 2008. p.9–15.

DOWNING, D. J.; GARDNER, R.; HOFFMAN, F. An examination of response-surface methodologies for uncertainty analysis in assessment models. **Technometrics**, [S.l.], v.27, n.2, p.151–163, 1985.

EC2, A. Disponível em: `http://aws.amazon.com/pt/ec2/`>. Acessado em: 10 de jun. 2016.

EUCALYPTUS. **Open Source Private Cloud Software**. Disponível em: `https://www.eucalyptus.com/`>. Acessado em: 20 de aug. 2016.

EUCALYPTUS. **Official Documentation for Eucalyptus Cloud**. Disponível em: `https://www.eucalyptus.com/docs/eucalyptus/4.0/.`>. Acessado em: 23 de nov. 2016.

EUCALYPTUS. **CloudWatch Troubleshooting**. Disponível em: `https://github.com/eucalyptus/eucalyptus/wiki/CloudWatch-Troubleshooting/`>. Acessado em: 20 de jul. 2016.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: a survey. **Future Generation Computer Systems**, [S.l.], v.29, n.1, p.84–106, 2013.

FIGUEIRÊDO, J. J. C.; MACIEL, P. R. M. O. Análise de dependabilidade de sistemas data center baseada em índices de importância. , [S.l.], 2011.

FLEURY, M.; REVERBEL, F. The JBoss extensible server. In: ACM/IFIP/USENIX 2003 INTERNATIONAL CONFERENCE ON MIDDLEWARE. **Proceedings...** [S.l.: s.n.], 2003. p.344–373.

FRANCESCHINI, G.; MACCHIETTO, S. Model-based design of experiments for parameter precision: state of the art. **Chemical Engineering Science**, [S.l.], v.63, n.19, p.4846–4872, 2008.

FRANK, P. M. **Introduction to System Sensitivity Theory**. [S.l.]: Academic Press Inc., 1978.

FREY, H. C.; MOKHTARI, A.; DANISH, T. Evaluation of selected sensitivity analysis methods based upon applications to two food safety process risk models. **Dept. of Civil, Construction, and Environmental Eng., North Carolina State Univ., Raleigh, NC**, [S.l.], 2003.

FURHT, B.; ESCALANTE, A. **Handbook of cloud computing**. [S.l.]: Springer, 2010. v.3.

FURTADO, M. T.; REGO, G.; LOURAL, C. d. A. Prospecção tecnológica e principais tendências em telecomunicações. **Cadernos CPqD Tecnologia**, [S.l.], v.1, n.1, 2005.

GHOSH, R. et al. Scalable analytics for iaas cloud availability. , [S.l.], 2014.

GIORDANELLI, R.; MASTROIANNI, C. The cloud computing paradigm: characteristics, opportunities and research issues. **Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)**, [S.l.], 2010.

GONG, C. et al. The characteristics of cloud computing. In: PARALLEL PROCESSING WORKSHOPS (ICPPW), 2010 39TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.275–279.

GUIMARAES, A. P. et al. Availability analysis of redundant computer networks: a strategy based on reliability importance. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATION SOFTWARE AND NETWORKS (ICCSN 2011), 3. **Proceedings...** [S.l.: s.n.], 2011. p.328–332.

GUIMARÃES, A. P.; MACIEL, P. R.; MATIAS, R. An analytical modeling framework to evaluate converged networks through business-oriented metrics. **Reliability Engineering & System Safety**, [S.l.], v.118, p.81–92, 2013.

HAJIAN-HOSEINABADI, H.; GOLSHAN, M. E. H. Availability, reliability, and component importance evaluation of various repairable substation automation systems. **IEEE Transactions on Power Delivery**, [S.l.], v.27, n.3, p.1358–1367, 2012.

HAMBY, D. A review of techniques for parameter sensitivity analysis of environmental models. **Environmental Monitoring and Assessment**, [S.l.], v.32, n.2, p.135–154, 1994.

HAMBY, D. A comparison of sensitivity analysis techniques. **Health Physics**, [S.l.], v.68, n.2, p.195–204, 1995.

HE, P. et al. Evaluation and optimization of the mixed redundancy strategy in cloud-based systems. **China Communications**, [S.l.], v.13, n.9, p.237–248, 2016.

HEARTBEAT. Linux-HA Project. Disponível em: `http://www.linux-ha.org`. Acessado em: mar. de 2016.

HENDERSON-SELLERS, B.; HENDERSON-SELLERS, A. Sensitivity evaluation of environmental models using fractional factorial experimentation. **Ecological Modelling**, [S.l.], v.86, n.2, p.291–295, 1996.

HOFFMAN, F.; GARDNER, R. Evaluation of Uncertainties in Environmental Radiological Assessment Models. In: TILL, J.; MEYER, H. (Ed.). **Radiological Assessments**: a textbook on environmental dose assessment. Washington, DC: U.S. Nuclear Regulatory Commission, 1983. Report No. NUREG/CR-3332.

HORST H. HENN (AUTH.) JAN HLAVIčKA, E. M. A. P. e. **Dependable Computing — EDCC-3**: third european dependable computing conference prague, czech republic, september 15–17, 1999 proceedings. 1.ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 1999. (Lecture Notes in Computer Science 1667).

HU, R.; XIE, J. Optimal maintenance policies for a cold standby redundant system with two units. In: SERVICE OPERATIONS AND LOGISTICS, AND INFORMATICS, 2008. IEEE/SOLI 2008. IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. v.2, p.1774–1778.

HU, T. et al. Mttf of composite web services. In: PARALLEL AND DISTRIBUTED PROCESSING WITH APPLICATIONS (ISPA), 2010 INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2010. p.130–137.

HUANG, W.; LOMAN, J.; SONG, T. Reliability modeling of A warm standby redundancy configuration with active→ standby→ active units. In: RELIABILITY AND MAINTAINABILITY SYMPOSIUM (RAMS), 2014 ANNUAL. **Anais...** [S.l.: s.n.], 2014. p.1–5.

HURBEAN, L.; FOTACHE, D. Mobile technology: binding social and cloud into a new enterprise applications platform. **Informatica Economica**, [S.l.], v.17, n.2, p.73, 2013.

HUSEBY, A. B. Importance measures for multicomponent binary systems. **Preprint series. Statistical Research Report**, [S.l.], 2004.

HWANG, I.-S. et al. Scalable architecture for VOD service enhancement based on a cache scheme in an Ethernet passive optical network. **Optical Communications and Networking, IEEE/OSA Journal of**, [S.l.], v.5, n.4, p.271–282, 2013.

IMAN, R. L.; JOHNSON, M. E.; WATSON, C. C. Sensitivity analysis for computer model projections of hurricane losses. **Risk Analysis**, [S.l.], v.25, n.5, p.1277–1297, 2005.

INDEX, C. V. N. **Forecast and Methodology, 2010–2015, June 1, 2011**. Disponível em: `http://www.abed.org.br/censoead2014/CensoEAD2014_portugues.pdf`>. Acessado em: 3 de out 2016.

JAIN, R. **The Art of Computer Systems Performance Analysis**: techniques for experimental design, measurement, simulation and modeling. New York: Wiley-Interscience, 1991.

JULIAN MENEZES ARAUJO, C.; ROMERO MARTINS MACIEL, P. O. Avaliação e modelagem de desempenho para planejamento de capacidade de sistema de transferência eletrônica de fundos utilizando tráfego em rajada. , [S.l.], 2009.

JUNIOR, R. M. et al. Sensitivity analysis of availability of redundancy in computer networks. In: THE FOURTH INTERNATIONAL CONFERENCE ON COMMUNICATION THEORY, RELIABILITY, AND QUALITY OF SERVICE. **Anais...** [S.l.: s.n.], 2011. v.121.

KEESEE, W. **A Method of Determining a Confidence Interval for Availability**. [S.l.]: DTIC Document, 1965.

KEESEE, W. R. **A Method of Determining a Confidence Interval for Availability**. Point Mugu, California: Miscellaneous Publication, 1965.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. S. Availability modeling and analysis of a virtualized system. In: DEPENDABLE COMPUTING, 2009. PRDC'09. 15TH IEEE PACIFIC RIM INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2009. p.365–371.

KUO, W.; ZHU, X. **Importance measures in reliability, risk, and optimization**: principles and applications. [S.l.]: John Wiley & Sons, 2012.

KUO, W.; ZUO, M. **Optimal reliability modeling: principles and applications**. [S.l.]: John Wiley & Sons, 2003.

KUO, W.; ZUO, M. J. **Optimal reliability modeling**: principles and applications. [S.l.]: John Wiley & Sons, 2003.

LAL, R.; KUMAR, A.; JOSHI, D. 2-unit standby system with fault analysis. **Reliability, IEEE Transactions on**, [S.l.], v.29, n.5, p.431–432, 1980.

LANE, K. **Overview of the backend as a service (BaaS) space**. 2015.

LAPRIE, J. C. C.; AVIZIENIS, A.; KOPETZ, H. (Ed.). **Dependability**: basic concepts and terminology. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1992.

LENK, A.; PALLAS, F. Cloud standby system and quality model. **Int. J. Cloud Comput. IJCC**, [S.l.], v.1, n.2, p.48–59, 2013.

LEVITIN, G.; XING, L.; DAI, Y. Optimal Backup Distribution in 1-out-of-Cold Standby Systems. **Systems, Man, and Cybernetics: Systems, IEEE Transactions on**, [S.l.], v.45, n.4, p.636–646, 2015.

LINDEMANN, J. Towards abuse detection and prevention in iaas cloud computing. In: AVAILABILITY, RELIABILITY AND SECURITY (ARES), 2015 10TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.211–217.

MACIEL, P. et al. Dependability modeling. In: **Performance and Dependability in Service Computing**: concepts, techniques and research directions. Hershey: IGI Global, 2011.

MACIEL, P. et al. Performance and Dependability in Service Computing: concepts, techniques and research directions. , [S.l.], v.1, n.1, p.53–97, 2011.

MACIEL, P. et al. Dependability Modeling. , [S.l.], p.53 –97, 2012.

MACIEL, P. R.; LINS, R. D.; CUNHA, P. R. **Introdução às redes de Petri e aplicações**. [S.l.]: UNICAMP-Instituto de Computacao, 1996.

MAIER, H.; NORTON, J.; CROKE, B. A comparison of sensitivity analysis techniques for complex models for environment management. , [S.l.], 2005.

MALHOTRA, M.; TRIVEDI, K. S. Power-hierarchy of dependability-model types. **Reliability, IEEE Transactions on**, [S.l.], v.43, n.3, p.493–502, 1994.

MÁRIO LINS GALDINO, S.; ROMERO MARTINS MACIEL, P. O. ISPN: modelagem e avaliação estocástica intervalar. , [S.l.], 2009.

MARKETSANDMARKETS.COM. **Cloud Backend-as-a-service (BaaS)/ Mobile BaaS (MBaaS) Market - Global Advancements, Business Models, Technology Roadmap, Forecasts Analysis (2012-2017)**. Disponível em: <http://migre.me/qRtZw>. Acessado em: 13 dez. 2014.

MARSAN, M. A. et al. **Modelling with generalized stochastic Petri nets**. [S.l.]: John Wiley & Sons, Inc., 1994.

MARVIN RAUSAND, A. H. **System Reliability Theory**: models, statistical methods, and applications, second edition. 2.ed. [S.l.]: Wiley-Interscience, 2003.

MARWAH, M. et al. Quantifying the sustainability impact of data center availability. **SIGMETRICS Perform. Eval. Rev.**, New York, NY, USA, v.37, p.64–68, March 2010.

MATHEMATICA, W. Wolfram Research. **Inc., Champaign, Illinois**, [S.l.], 2012.

MATHEWS, P. G. **Design of Experiments with MINITAB**. [S.l.]: ASQ Quality Press, 2005.

MATOS, R. et al. Sensitivity analysis of server virtualized system availability. **Reliability, IEEE Transactions on**, [S.l.], v.61, n.4, p.994–1006, 2012.

MATOS, R. et al. Sensitivity analysis of a hierarchical model of mobile cloud computing. **Simulation Modelling Practice and Theory**, [S.l.], v.50, p.151–164, 2015.

MATOS, R.; MACIEL, P.; SILVA, R. Sensitive GRASP: combinatorial optimization of composite web services guided by sensitivity analysis. **International Journal of Web and Grid Services**, [S.l.], 2015. To be published.

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

MELO, A. M. d. Avaliação de Performabilidade de Riscos de Desenvolvimento em Projetos de Software. , [S.l.], 2014.

MELO, R. et al. Video on Demand hosted in Private Cloud: availability modeling and sensitivity analysis. In: IEEE INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS WORKSHOPS, 2015. **Anais...** [S.l.: s.n.], 2015. p.12–18.

MENASCE, D. A. et al. **Performance by design**: computer capacity planning by example. [S.l.]: Prentice Hall Professional, 2004.

MERCURY. **Mercury Tool**. Disponível em: https://sites.google.com/site/mercurytooldownload/>. Acessado em 20 de nov. 2016.

MILLS, K.; FILLIBEN, J.; DABROWSKI, C. An efficient sensitivity analysis method for large cloud simulations. In: CLOUD COMPUTING (CLOUD), 2011 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.724–731.

MONTGOMERY, D. C. **Design and Analysis of Experiments**. 8.ed. [S.l.]: Wiley, 2012.

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

NETFLIX. **Lessons Netflix Learned from the AWS Outage**. Disponível em `http://techblog.netflix.com/2011/04/lessons-netflix-learned-from-aws-outage.html` Acessado em: 13 dez. 2016.

NORTON, J. An introduction to sensitivity assessment of simulation models. **Environmental Modelling & Software**, [S.l.], v.69, p.166–174, 2015.

NURMI, D. et al. The eucalyptus open-source cloud-computing system. In: CLUSTER COMPUTING AND THE GRID, 2009. CCGRID'09. 9TH IEEE/ACM INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2009. p.124–131.

OAKLEY, J. E.; O'HAGAN, A. Probabilistic sensitivity analysis of complex models: a bayesian approach. **Journal of the Royal Statistical Society: Series B (Statistical Methodology)**, [S.l.], v.66, n.3, p.751–769, 2004.

OLIVEIRA, D. M.; MACIEL, P. R. M. O. Análise de Disponibilidade e Consumo Energético em Ambientes de Mobile Cloud Computing. , [S.l.], 2014.

OU, Y.; DUGAN, J. B. Sensitivity analysis of modular dynamic fault trees. In: COMPUTER PERFORMANCE AND DEPENDABILITY SYMPOSIUM, 2000. IPDS 2000. PROCEEDINGS. IEEE INTERNATIONAL. **Anais...** [S.l.: s.n.], 2000. p.35–43.

OU, Y.; DUGAN, J. B. Approximate sensitivity analysis for acyclic Markov reliability models. **IEEE Transactions on Reliability**, [S.l.], v.52, n.2, p.220–230, 2003.

PEIXOTO, M. L. M. **Oferecimento de QoS para computação em nuvens por meio de metaescalonamento**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

PIANOSI, F. et al. Sensitivity analysis of environmental models: a systematic review with practical workflow. **Environmental Modelling & Software**, [S.l.], v.79, p.214–232, 2016.

PREECE, R.; MILANOVIC, J. V. Assessing the Applicability of Uncertainty Importance Measures for Power System Studies. , [S.l.], 2015.

RAEDER, M. et al. Performance Prediction of Parallel Applications with Parallel Patterns Using Stochastic Methods. **Sistemas Computacionais (WSCAD-SSC), XII Simpósio em Sistemas Computacionais de Alto Desempenho**, [S.l.], p.1–13, 2011.

ROWINSKI, D. **The Rise of Mobile Cloud Services**: baas startups grow up. 2012.

SALTELLI, A. Sensitivity analysis: could better methods be used? **Journal of Geophysical Research: Atmospheres (1984–2012)**, [S.l.], v.104, n.D3, p.3789–3793, 1999.

SALTELLI, A. Sensitivity analysis for importance assessment. **Risk Analysis**, [S.l.], v.22, n.3, p.579–590, 2002.

SALTELLI, A. et al. **Sensitivity analysis**. [S.l.]: Wiley New York, 2000. v.1.

SALTELLI, A. et al. **Sensitivity analysis in practice**: a guide to assessing scientific models. [S.l.]: John Wiley & Sons, 2004.

SAREEN, P. Cloud Computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud. **International Journal of Advanced Research in Computer Science and Software Engineering**, [S.l.], v.3, n.3, 2013.

SATO, N.; TRIVEDI, K. S. Stochastic Modeling of Composite Web Services for Closed-Form Analysis of Their Performance and Reliability Bottlenecks. In: ICSOC. **Anais...** [S.l.: s.n.], 2007. p.107–118.

SILVA, A. N. da et al. Avaliaçao de desempenho da composiçao de web services usando redes de petri. In: BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS. CURITIBA, PARANÁ, BRAZIL. **Anais...** [S.l.: s.n.], 2006.

SILVA, B. et al. Dependability models for designing disaster tolerant cloud computing systems. In: DEPENDABLE SYSTEMS AND NETWORKS (DSN), 2013 43RD ANNUAL IEEE/IFIP INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.1–6.

SILVA, B. et al. Mercury: an integrated environment for performance and dependability evaluation of general systems. In: INDUSTRIAL TRACK AT 45TH DEPENDABLE SYSTEMS AND NETWORKS CONFERENCE, Rio de Janeiro. **Proceedings...** IEEE, 2015. (DSN 2015).

SILVA, B. et al. Mercury: an integrated for performance and dependability evaluation of general system. In: INDUSTRIAL TRACK AT 45TH DEPENDABLE SYSTEMS AND NETWORK CONFERENCE (DSN 2015). **Proceedings...** [S.l.: s.n.], June 22-25,2015. Rio de Janeiro,RJ,Brazil.

SOUSA, E. et al. Dependability evaluation of cloud infrastructures. In: SYSTEMS, MAN AND CYBERNETICS (SMC), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.1282–1287.

SOUSA, E. et al. A Modeling Approach for Cloud Infrastructure Planning Considering Dependability and Cost Requirements. **Systems, Man, and Cybernetics: Systems, IEEE Transactions on**, [S.l.], v.45, n.4, p.549–558, 2015.

SOUZA, E. Teixeira Gomes de. **Avaliação do impacto de uma política de manutenção na performabilidade de sistemas de transferência eletrônica de fundos**. [S.l.]: Universidade Federal de Pernambuco, 2009.

SOUZA MATOS JÚNIOR, R. de. An automated approach for systems performance and dependability improvement through sensitivity analysis of Markov chains. , [S.l.], 2011.

STANDART, N. I. of; TECHNOLOGY'S. **Cloud Computing**. Disponível em: `http://www.nist.gov`>. Acessado em jul 2016.

STARKE, P. H.; ROCH, S. INA: integrated net analyzer. **Reference Manual**, [S.l.], 1992.

SUN, Y.; XIAO, C.; LANG, Y. Predicating reservoir sensitivity rapidly with single-correlation analysis and multiple regression. In: INFORMATION TECHNOLOGY AND ARTIFICIAL INTELLIGENCE CONFERENCE (ITAIC), 2011 6TH IEEE JOINT INTERNATIONAL. **Anais...** [S.l.: s.n.], 2011. v.2, p.100–104.

TANNOUS, O. et al. Redundancy allocation for series-parallel warm-standby systems. In: INDUSTRIAL ENGINEERING AND ENGINEERING MANAGEMENT (IEEM), 2011 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.1261–1265.

TAURION, C. Cloud Computing: computação em nuvem: transformando o mundo da tecnologia da informação. **Rio de Janeiro: Brasport**, [S.l.], p.2, 2009.

TAY, Y. **Analytical Performance Modeling for Computer Systems**. 1.ed. [S.l.]: Morgan and Claypool Publishers, 2010. (Synthesis Lectures on Computer Science).

TIWARI, N. et al. Identification of critical parameters for MapReduce energy efficiency using statistical Design of Experiments. In: PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOPS, 2016 IEEE INTERNATIONAL. **Anais...** [S.l.: s.n.], 2016. p.1170–1179.

TRIVEDI, K. S. et al. Reliability analysis techniques explored through a communication network example. , [S.l.], 1996.

WEBER, T. S. Tolerância a falhas: conceitos e exemplos. **Apostila do Programa de Pós-Graduação–Instituto de Informática-UFRGS. Porto Alegre**, [S.l.], 2003.

WILLIAMS, J. R. et al. Sensitivity analysis in model-driven engineering. In: INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS. **Anais...** [S.l.: s.n.], 2012. p.743–758.

WULFF, S. S. A First Course in Design and Analysis of Experiments. **The American Statistician**, [S.l.], 2012.

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

XU, J. et al. Availability Modeling and Analysis of a Single-Server Virtualized System with Rejuvenation. **JSW**, [S.l.], v.9, n.1, p.129–139, 2014.

YOUTUBE. **Estatísticas**. Disponível em: `https://www.youtube.com/yt/press/pt-BR/statistics.html`. Acessado em out.2015.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of internet services and applications**, [S.l.], v.1, n.1, p.7–18, 2010.

ZHOU, Y.; FU, T. Z.; CHIU, D. M. Statistical modeling and analysis of p2p replication to support vod service. In: INFOCOM, 2011 PROCEEDINGS IEEE. **Anais...** [S.l.: s.n.], 2011. p.945–953.

Apêndice

A

Derivadas parciais dos estudos de casos

Nos estudos de casos 01, 02 e 03, as taxas de falha e de reparo do subsistema de *Frontend* são representados por λ_f e μ_f respectivamente e correspondem as derivadas parciais das expressões A.1 e A.2. A expressão da derivada parcial para A_{Node} e A_{NAS}, A_{MBaaS} e A_{Volume} são similar ao do subsistema *Frontend* uma vez que estes subsistemas são representados por modelos RBD.

$$\frac{\partial A_{Frontend}}{\partial \lambda_f} = -\frac{\mu_f}{(\mu_f + \lambda_f)^2} \tag{A.1}$$

$$\frac{\partial A_{Frontend}}{\partial \mu_f} = -\frac{\mu_f}{(\lambda_f + \mu_f)^2} + \frac{1}{\lambda_f + \mu_f} \tag{A.2}$$

A partir do estudo de caso 01, na Seção 7.1.7 página 89, as expressões das derivadas parciais para o subsistema do Serviço, A_{VoD1} do modelo CTMC são apresentados nas Equações A.3, A.4, A.5, A.6, A.7 e A.8, elas mostram as derivadas parciais para a Disponibilidade do subsistema do Serviço de *streaming* de vídeo.

$$\begin{aligned}\frac{\partial A_{VoD1}}{\partial \lambda_{vm}} = & \\ & - \frac{((\beta)(\lambda_{vm} + \mu_{vlc}(\beta_3) + (\beta_1)\lambda_{ap}\lambda_{vm} + \lambda_{ap}(\beta_3)\mu_{vlc})\mu_{in}}{(\beta_1)^2(\beta_2)(\gamma)(\lambda_{ap} + \beta)} \\ & - \frac{((\beta + \lambda_{vlc})(\alpha)(\beta_3) + (\beta_1)\lambda_{ap}\lambda_{vm} + \lambda_{ap}(\beta_3)\mu_{vlc})\mu_{in}}{(\beta_1)(\beta_2)(\gamma)^2(\lambda_{ap} + \beta)} \\ & - \frac{((\beta)(\alpha)(\beta_1) + (\beta_1)\lambda_{ap}\lambda_{vm} + \lambda_{ap}(\beta_3)\mu_{vlc})\mu_{in}}{(\beta_2)(\beta_2)(\alpha)(\lambda_{ap} + \beta)^2} \\ & + \frac{\mu_{in}((\beta)(\beta_3) + (\beta_3)(\beta_3) + (\beta(\beta_3) + (\beta_1)\lambda_{ap} + \kappa)}{(\beta_1)(\beta_2(\gamma)(\lambda_{ap} + \beta)} \\ & - \frac{((\beta)(\alpha)(\beta_3) + (\beta_1\lambda_{ap}\lambda_{vm} + \lambda_{ap}(\beta_3)\mu_{vlc})\mu_{in}}{(\beta_1)(\beta_2)^2(\gamma)(\lambda_{ap} + \beta)}\end{aligned} \tag{A.3}$$

$$\frac{\partial As}{\partial \lambda_{ap}} = \tag{A.4}$$
$$-\frac{((\alpha)(\beta)(\beta_3)+(\beta_1)\lambda_{ap}\lambda_{vm}+\lambda_{ap}\mu_{vlc}(\beta_3))\mu_{in}}{(\beta_1)(\gamma)(\beta_2)^2(\beta+\lambda_{ap})}$$
$$-\frac{((\alpha)(\beta)(\beta_3)+(\beta_1)\lambda_{ap}\lambda_{vm}+\kappa(\beta_3))\mu_{in}}{(\beta_1)(\gamma)(\beta_2)(\beta_+\lambda_{ap})^2}$$
$$+\frac{(\mu_{vlc}(\beta_3)+(\beta_1)\lambda_{vm})\mu_{in}}{(\beta_1)(\gamma)(\beta_2)(\beta+\lambda_{ap})}$$

$$\frac{\partial As}{\partial \lambda_{vlc}} = \tag{A.5}$$
$$-\frac{(\lambda_{ap}\mu_{vlc}(\beta_3)+\lambda_{ap}\lambda_{vm}(\beta_1+(\beta)(\alpha(\beta_3)\mu_{in}}{(\gamma)(\beta_1)^2(\lambda_{ap}+\beta(\beta_2)}$$
$$+\frac{(\lambda_{ap}\lambda_{vm}+(\alpha)(\beta_3))\mu_i n}{(\gamma)(\beta_1)(\lambda_{ap}+\beta)(\beta_2)}$$
$$-\frac{(\lambda_{ap}\mu_v lc(\beta_3)+\lambda_{ap}\lambda_{vm}(\beta_1)+(\beta)(\alpha)\beta_3)\mu_{in}}{(\alpha)(\beta_1)(\lambda_{ap}+\beta)^2(\beta_2)}$$

$$\frac{\partial As}{\partial \mu_{vlc}} = \tag{A.6}$$
$$-\frac{((\beta_3)\lambda_{ap}\mu_{vlc}+\lambda_{vm}(\beta_1)\lambda_{ap}+(\beta_3)(\beta)(\alpha))\mu_{in}}{(\gamma)(\beta_1)(\beta_1+\lambda_{ap}+\mu_{vlc})^2(\beta_2)}$$
$$+\frac{\mu_{in}((\beta_3)(\alpha)+(\beta_3)(\beta)+(\beta_3)\lambda_{ap})}{(\gamma)(\beta_1)(\beta+\lambda_{ap})(\beta_2)}$$

$$\frac{\partial As}{\partial \mu_{in}} = \tag{A.7}$$
$$-\frac{\mu_{in}((\beta_3)(\alpha)(\beta)+(\beta_3)\lambda_{ap}\mu_{vlc}+(\beta_1)\lambda_{ap}\lambda_{vm})}{(\beta_1)(\beta_1)(\lambda_{ap}+\beta)(\gamma)^2}$$
$$+\frac{(\beta_3)(\beta_3)(\beta)+(\beta_3)\lambda_{ap}\mu_{vlc}+(\beta_2)\lambda_{ap}\lambda_{vm}}{(\beta_1)(\beta_2)(\beta+\lambda_{ap})(\gamma)}$$

$$\frac{\partial As}{\partial \mu_{ap}} = \tag{A.8}$$
$$-\frac{\mu_{in}((\alpha)(\beta_3)+\lambda_{vm}\lambda_{ap}+\mu_{vlc}\lambda_{ap}+(\alpha)(\beta))}{(\gamma)(\beta_2)(\beta_1)(\beta+\lambda_{ap})}$$
$$-\frac{(\lambda_{vm}(\beta_2)\lambda_{ap}+\mu_{vlc}(\beta_3)\lambda_{ap}+(\alpha)(\beta)(\beta_3))\mu_{in}}{(\gamma)(\beta_2)(\beta_1)^2(\beta+\lambda_{ap})}$$
$$-\frac{(\lambda_{vm}(\beta_1)\lambda_{ap}+\mu_{vlc}(\beta_3)\lambda_{ap}+(\alpha)(\beta)(\beta_3))\mu_{in}}{(\gamma)(\beta_2)(\beta_1)(\beta+\lambda_{ap})^2}$$
$$-\frac{(\lambda_{vm}(\beta_1)\lambda_{ap}+\mu_{vlc}(\beta_3)\lambda_{ap}+(\alpha)(\beta)(\beta_3))\mu_{in}}{(\gamma)(\beta_2)^2(\beta_1)(\beta+\lambda_{ap})}$$

Onde:
$\beta = \mu_{ap} + \mu_{vlc} + \lambda_{vm} + \lambda_{vlc};;\beta_1 = \mu_{ap} + \lambda_{vlc};\beta_2 = \mu_{ap} + \lambda_{ap} + \lambda_{vm};$
$\beta_3 = \mu_{ap} + \lambda_{vm};\alpha = \mu_{vlc} + \lambda_{vm};\ \gamma = \lambda_{vm} + \mu_{in};\ \kappa = \lambda_{ap} + \mu_{vlc} + \lambda_{ap}.\lambda_{vm}.$

A partir do estudo de caso 02, na Seção 7.2.7, página 103, as seguintes Equações A.7, A.8, A.9, A.10 e A11, mostram as derivadas parciais para a disponibilidade do subsistema do Serviço, A_{VoD2}.

$$\frac{\partial A_{VoD2}}{\partial \mu_{VoD}} = + \frac{-(\alpha_1 \left(2\lambda_{wN^2}\alpha^2(2(\beta_4+\mu_N)+\alpha+(\alpha_3(\alpha_4))+\mu_{wN^2}+(\alpha_5)\mu_{wN^3})\right)}{(\beta 2\Omega 1+(\beta 2\Omega\mu_w N+\phi 1+(2\lambda_N\lambda_w N\phi)} + \frac{(\mu_N(2(\lambda_{wN^2}(\beta 3)(2(\beta 4)+\Omega 2)}{(\beta 2)(\lambda_w N^2)(\beta 3)(2(\beta 4)+\mu_N)\beta 8+\delta)\mu_w N+(\delta 1+(\beta 4)\beta 6+\beta 7+(\lambda_N\beta 8(\alpha 5))\mu_w N^3)} \quad \text{(A.9)}$$

$$\frac{\partial A_{VoD2}}{\partial \lambda_{VoD}} = \frac{\alpha_1(2\lambda_w N^2(\beta_3)(2(\beta 4)+\mu_N+(\alpha)\mu_{wN}+\alpha 3(\alpha 4)\mu_{wN^2}+(\alpha 5)\mu_w N^3}{\beta_2\lambda_{wN^2}(\alpha_2)2(\beta_3)(\lambda_N+2\mu_N)+\lambda_{wN^2}\lambda_N(\beta_4)(\lambda_N+3\lambda_{wN})\beta\mu_N+\phi_2+\phi_1+\gamma_1+\beta 4\beta 6+\gamma+\gamma_2} \quad \text{(A.10)}$$

$$\frac{\partial A_{VoD2}}{\partial \lambda_N} =$$
$$+\frac{-(\delta_1\left(2\lambda wN^2\Omega^2(2(\beta_5+\mu N)+\alpha+(\alpha_3(\phi_4))+\mu wN^2+(\alpha_5)\mu wN^3)\right)}{(\beta 2\Omega 1+(\beta 2\Omega+\phi_1+(2\lambda_N\lambda_w N\phi)}$$
$$+\frac{8\lambda wN((2(\lambda wN^2(\alpha 3)(2(\beta 4)+\Omega 2)}{(\sigma 2)(\lambda wN^2)(\beta 3)(2(\beta 4)+)\beta 8+\delta)+(\delta 1+(\beta 4)\phi 3+\alpha 5+(\beta 8(\alpha 5))\mu wN^3)}$$
$$+\frac{(\mu_N(2(\lambda_{wN^2}(\beta 3)(2(\beta 4)+\Omega 2)}{\phi_2(2\lambda wN^2(\beta_3)((\beta 2)(^2)(\beta 3)(2(\beta 4)+)\beta 8+\delta)+(\delta 1+(\beta 4)\beta 6+\alpha 2+(\lambda_N\beta 8(\gamma 1))\mu wN^3)} \quad \text{(A.11)}$$

$$\frac{\partial A_{VoD2}}{\partial \mu_N} =$$
$$+\frac{-(\gamma_1\left(2\lambda wN^2\alpha^2(2(\beta_4+\mu N)+\alpha+\alpha_3(11\lambda wN+))\mu wN^2+(\alpha_3(\alpha_4))+\mu wN^2+(\alpha_5)\mu wN^3)\right)}{(2(\phi 2+2\lambda)\alpha 2\Omega 1+(\beta 2\Omega\mu wN+\phi 1+(7\lambda_N\lambda_w N\phi)}$$
$$+\frac{(\mu_N(2(\lambda_{wN^2}(\beta 3)(2(\beta 4)+\Omega 2)}{\alpha_1(2N^2(\beta_3)((\beta 2)(^2)(\beta 3)(2(\beta 4)+)\beta 8+\delta)+(\delta 1+(\beta 4)\beta 6+\alpha 2+(\lambda_N\beta 8(\gamma 1))\mu wN^3)} \quad \text{(A.12)}$$

$$\frac{\partial A_{VoD2}}{\partial \lambda_{WN}} =$$
$$+\frac{-2(\alpha 2+2\lambda)(\alpha_1\left(2\lambda_{wN^2}\alpha^2(2(\beta_4+\mu_N)+\alpha+(\alpha_3(\alpha_4))+\mu_{wN^2}+(\alpha_5)\mu_{wN^3})\right)}{(\beta 2\Omega 1+(\beta 2\Omega\mu_w N+\phi 1+(3\lambda N12\lambda_N\mu wN^2\lambda_w N\phi)}$$
$$+\frac{(\alpha\mu_N(2(\lambda_{wN^2}(\beta 3)(2(\beta 4)+\Omega 2)}{(\beta 2)(\lambda_w N^2)(\beta 3)(2(\beta 4)+\mu_N)\beta 8+\delta)\mu_w N+(\delta 1+(\beta 4)\beta 6+\beta 7+(\lambda_N\beta 8(\alpha 5))\mu_w N^3)} \quad \text{(A.13)}$$

Onde:

$\alpha = \lambda_{wN}(4\lambda_{N^2} + 17\lambda_N\lambda_{wN} + 13\lambda_{wN^2} + 5\lambda_N\mu_N + 12\lambda_{wN}\mu_N + 2\mu_{N^2)}$;

$\alpha_1 = \mu_N\mu_{VoD}$;$\alpha_2 = \lambda_N + \lambda_{wN} + \mu_N$;$\alpha_3 = (8\lambda_{wN}(2\lambda_{wN} + \mu_N) + \lambda_N)$;$\alpha_4 = 11\lambda_{wN} + \mu_N$;

$\alpha_5 = 7\lambda_{wN} + 2\mu_N$;$\beta_{=}(8\lambda_{N^2} + 24\lambda_N\lambda_{wN}) + 13\lambda_{wN^2}$;

$\beta_1 = (7\lambda_N + 12\lambda_{wN})\mu_{N^2}$;$\beta_2 = (\lambda_N + \lambda_{VoD} + \mu_{VoD})^2$;$\beta_3 = \lambda_N + \lambda_{wN} + \mu_N$;$\beta_4 = \lambda_N + \lambda_{wN}$;

$\beta_5 = \lambda_{wN}(2\lambda_N(\beta_4)(\lambda_N + 3\lambda_{wN})$;$\beta_6 = (\lambda_N + 16\lambda_w N)\mu_N$;

$\beta_7 = (\lambda_N + 8\lambda_w N^2)\mu_w N^2$;$\beta_8 = (2\mu_N + \lambda_N)$;

$\beta_9 = \lambda_w N(4\lambda N^2 + 17\lambda_N\lambda_{wN} + 13\lambda_w N^2 + 5\mu_N + 12\lambda_{wN}\mu N + 2\mu N^2)\mu_w N$;$\gamma = (\lambda_N + 8\lambda wN\mu_{N^2})\mu wN^2$;$\gamma_1 =$
$2\lambda_N\lambda_{wN}(2\lambda_N + 3\lambda_{wN})$;$\gamma_2 = (\lambda_N(\lambda_{wN} + 2\mu_N) + \mu_N(7\lambda_{wN} + 2\mu_N))\mu_{wN^3}$;

$\gamma_3 = \mu_N(\lambda_N + 16\lambda_{wN})$;

$\Omega = (\lambda_w N^2(\alpha 2)(2(\beta 4) + \mu_N)(\lambda_N + 2\mu_N) + \beta 5 + (\beta)\mu_N + \beta 1 + 2\mu_{N^3})$;

$\Omega_1 = (\lambda_{wN^2}(\alpha 2)(2(\beta 4) + \mu_N)(\lambda_N + 2\mu N) + \beta 5 + \beta)\mu_N + \beta_1 + 2\mu_{N^3})\mu_{wN}$;$\Omega_2 = \mu_N + \beta 9 +$
$(8\lambda_w N(\beta 8) + \lambda_N(11\lambda_w N + \mu N))\mu_w N^2 + (\alpha 5)\mu_w N^3$;$\delta = \beta 5 + \beta + \beta 1 + 2\mu N^3$;

$\delta_1 = 2\lambda N\lambda wN(2\lambda N + 3\lambda wN)$;

$\phi = (2\lambda_N + 3wN) + (\beta 4)\beta 6 + (\beta 7 + (\lambda N\beta 8 + (\alpha 5))\mu_w N^3$;

$\phi_1 = 2\lambda N\lambda wN\phi$;$\phi_2 = (7\lambda_N + 12\lambda_{wN})\mu_{N^2} + 2\mu_{N^3}\mu_{wN}$