UFRPE

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA

Modelos para avaliação de desempenho e disponibilidade de serviços hospedados em nuvens privadas

Wenderson de Souza Leonardo

Recife - PE, Março 2025

Wenderson de Souza Leonardo

Modelos para avaliação de desempenho e disponibilidade de serviços hospedados em nuvens privadas

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Pró-Reitoria de Pós-Graduação da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Universidade Federal Rural de Pernambuco

Pró-Reitoria de Pós-Graduação

Programa de Pós-Graduação em Informática Aplicada

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Recife - PE

Março 2025

Dados Internacionais de Catalogação na Publicação
Sistema Integrado de Bibliotecas da UFRPE
Bibliotecário(a): Auxiliadora Cunha – CRB-4 1134

L581m Leonardo, Wenderson de Souza.
Modelos para avaliação de desempenho e disponibilidade de serviços hospedados em nuvens privadas / Wenderson de Souza Leonardo. – Recife, 2025.
89 f.; il.

Orientador(a): Gustavo Rau de Almeida Callou.

Dissertação (Mestrado) – Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, BR-PE, 2026.

Inclui referências.

1. Computação em nuvem. 2. Balanceamento de carga em sistemas distribuídos. 3. Disponibilidade. 4. Medidores de fluxo 5. Tempo de reação. I. Callou, Gustavo Rau de Almeida, orient. II. Título

CDD 004

Wenderson de Souza Leonardo

Modelos para avaliação de desempenho e disponibilidade de serviços hospedados em nuvens privadas

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Pró-Reitoria de Pós-Graduação da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em Recife - PE, «Março» «31», «2025»:

Prof. Dr. Gustavo Rau de Almeida Callou (Orientador)
Universidade Federal Rural de Pernambuco

Prof. Dr. Carlos Julian Menezes Araújo
Universidade Federal do Cariri

Prof. Dr. Erica Teixeira Gomes de Sousa
Universidade Federal Rural de Pernambuco

Recife - PE
Março 2025

«Dedico este trabalho a minha família.»

Agradecimentos

Agradeço ao professor Gustavo Callou pela oportunidade e a apoio na condução deste trabalho. Agradeço tambem aos colegas do grupo Pedal, especialmente a Alison da Silva, Thiago Valentim e Ivson Borges por me auxiliar na condução desta pesquisa.

"A persistência é o caminho do êxito."
(Charles Chaplin)

Resumo

A tecnologia tem se tornado cada vez mais uma parte integrante do cotidiano humano, e, com isso, surge a necessidade de desenvolver soluções que atendam às demandas diárias por meio de aplicações adequadas. Nesse contexto, novas tecnologias têm surgido como alternativas para suprir essa demanda, caracterizada pela necessidade de alta conectividade e acesso a recursos sob demanda. A computação em nuvem tem se consolidado como uma solução estratégica para atender às crescentes exigências de conectividade e flexibilidade no acesso a recursos computacionais. Com a possibilidade de acessar recursos sob demanda e a escalabilidade que oferece, as empresas estão progressivamente migrando para ambientes de nuvem privada, a fim de garantir maior controle sobre a segurança e a gestão de seus dados. No entanto, essa migração traz desafios, como a identificação de componentes críticos, essencial para implementar mecanismos de redundância eficazes, reduzir riscos de falhas e prejuízos, e garantir um ambiente estável, escalável, de altos desempenho e disponibilidade. Em ambientes de nuvem com múltiplas instâncias de um serviço, o balanceamento de carga é essencial para distribuir eficientemente o tráfego, evitar sobrecargas e melhorar o desempenho do sistema. O presente trabalho propõe modelos baseados em Redes de Petri Estocásticas (SPN) e Diagramas de Blocos de Confiabilidade (RBD) para avaliar um ambiente de nuvem privada, considerando métricas como disponibilidade, vazão e tempo de resposta. Através da modelagem do sistema, foram realizados estudos de caso com diferentes enfoques para analisar o desempenho e a disponibilidade de um serviço Web instalado nesse ambiente. O serviço avaliado é o Nextcloud, que oferece funcionalidade de sistema de arquivos como serviço, sendo disponibilizado por múltiplas máquinas virtuais (VMs). A distribuição das requisições entre as VMs foi gerenciada pelo Nginx, escolhido por sua simplicidade e facilidade de configuração como balanceador de carga. O primeiro estudo de caso visa analisar as diferenças na disponibilidade e no desempenho de diversas distribuições de VMs em um servidor com dois hosts. O segundo estudo de caso avalia o sistema a partir de uma abordagem de projeto de experimentos (DoE), utilizando um design fatorial de dois níveis (2^k). O terceiro estudo de caso foca na avaliação da disponibilidade, aplicando a técnica do índice de importância da disponibilidade para identificar os componentes mais impactantes no sistema. O quarto estudo de caso investiga os diferentes tempos de chegada dos usuários ao sistema e analisa como isso afeta o desempenho global do ambiente. Os resultados dos experimentos demonstram que a análise dos componentes pode levar à identificação de pontos mais suscetíveis a falhas, além de evidenciar que a implementação de redundâncias para esses componentes críticos contribui para a melhoria das métricas de desempenho e disponibilidade.

Palavras-chaves: Computação em Nuvem, Balanceamento de Carga, Disponibilidade, Vazão, Tempo de Resposta.

Abstract

Technology has increasingly become an integral part of human daily life, leading to the need for solutions that meet daily demands through appropriate applications. In this context, new technologies have emerged as alternatives to address these needs, characterized by the demand for high connectivity and on-demand access to resources. Cloud computing has consolidated itself as a strategic solution to meet the growing demands for connectivity and flexibility in accessing computational resources. With the ability to access resources on demand and the scalability it offers, companies are progressively migrating to private cloud environments to ensure greater control over the security and management of their data. However, this migration presents several challenges, such as the identification of critical components—an essential step for implementing effective redundancy mechanisms, reducing the risk of failures and associated losses, and ensuring a stable, scalable environment with high performance and availability. In cloud environments with multiple instances of a given service, load balancing is crucial for efficiently distributing traffic, preventing overloads, and improving overall system performance. This study proposes models based on Stochastic Petri Nets (SPN) and Reliability Block Diagrams (RBD) to evaluate a private cloud environment, taking into account metrics such as availability, throughput, and response time. Through system modeling, case studies with distinct perspectives were conducted to analyze the performance and availability of a web service deployed in this environment. The evaluated service is Nextcloud, which provides file system functionality as a service and is hosted across multiple virtual machines (VMs). Request distribution among the VMs was managed by Nginx, chosen for its simplicity and ease of configuration as a load balancer. The first case study aims to analyze the differences in availability and performance across various virtual machine (VM) distributions on a server with two hosts. The second case study evaluates the system from a design of experiments (DoE) approach, utilizing a two-level factorial design (2^k). The third case study focuses on evaluating availability by applying the availability importance index technique to identify the most impactful components in the system. The fourth case study investigates the varying arrival times of users to the system and analyzes how this affects the overall performance of the environment. The experimental results demonstrate that component analysis can lead to the identification of the most failure-prone points, as well as highlight that implementing redundancies for these critical components contributes to improvements in performance and availability metrics.

Keywords: Cloud Computing, Load Balancing, Availability, Throughput, Response Time.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CPVM	Console Proxy Virtual Machine
DDFTP	Dual Direction Downloading Algorithm from FTP servers
DoE	Design of Experiments
IAAS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
LB	Load Balancer
MRM	Markov Reward Models
MTTA	Mean Time To Activate
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
PAAS	Platform as a Service
RBD	Reliability Block Diagram
SAAS	Software as a Service
SPN	Stochastic Petri Net
SSVM	Secondary Storage Virtual Machine
VM	Virtual machine
VR	Virtual router

Sumário

1 Introdução

Com o avanço contínuo da tecnologia, os dispositivos eletrônicos têm se tornado progressivamente mais acessíveis e presentes na vida cotidiana de uma grande parcela da população mundial. Este fenômeno tem propiciado que um número crescente de indivíduos passe pelo processo de modernização, por meio da atualização de suas ferramentas para versões tecnológicas mais avançadas. Exemplos disso são a transição dos antigos telefones fixos para os celulares, e destes para os *smartphones*, que funcionam, na prática, como computadores compactos que se acomodam na palma da mão. Nesse contexto, o uso de computadores e celulares tem deixado de ser meramente instrumental, para se transformar em componentes essenciais do cotidiano. Além de sua utilização pessoal, esses dispositivos passaram a ser aplicados em diversas áreas, abrangendo desde a indústria automotiva até o setor de pecuária e agricultura.

Além de possibilitar a criação de versões aprimoradas de dispositivos já existentes, o avanço da tecnologia também implica uma característica adicional: a capacidade dos dispositivos modernos de se conectarem à Internet, um recurso que, originalmente, era restrito aos computadores. A introdução dessa funcionalidade, que permite o acesso à web a partir de qualquer lugar por meio de dispositivos pessoais compactos, como *smartphones*, *tablets* e *notebooks*, conferiu um novo significado ao uso da Internet. O propósito da rede, que inicialmente era mais voltado para atividades profissionais, passou a ter um uso predominantemente pessoal. Esse novo uso inclui desde a realização de pesquisas com fins educacionais até a troca de mensagens entre indivíduos, evidenciando a crescente diversificação das aplicações da Internet ao longo do tempo.

A combinação de tecnologia acessível e a possibilidade de conexão à Internet em qualquer lugar gerou o hábito de buscar soluções online. Esse fenômeno abriu um novo nicho para as empresas de tecnologia: a oferta de serviços pela web. Como se esse fosse o próximo passo aguardado pela humanidade, um número crescente de indivíduos e organizações passou a adotar a proposta de serviços digitais. Exemplos disso incluem serviços de correio eletrônico (e-mail), de edição de documentos (pacote office) e de armazenamento em nuvem. Esse movimento tem levado cada vez mais pessoas a permanecerem conectadas enquanto realizam suas tarefas cotidianas.

A partir da crescente necessidade de permanência constante na conexão, os usuários passaram a buscar soluções que atendam às suas demandas de maneira online e remota. Com base nesse conceito de satisfazer suas necessidades à distância, enquanto se encontra conectado, diversas soluções foram desenvolvidas, como serviços de pedidos online, dispositivos inteligentes conectados à rede e até aplicações para versões virtuais de documentos.

Contudo, para atender a essa demanda em expansão, os provedores de serviços precisam contar com cada vez mais poder computacional.

Nesse contexto, novos paradigmas computacionais têm surgido como formas de solucionar essas novas demandas. Dentre esses paradigmas, destacam-se alguns mais amplamente conhecidos, como a computação em nuvem e a Internet das coisas, que visam, respectivamente, fornecer recursos computacionais sob demanda por meio da Internet e interconectar objetos do cotidiano à rede.

Ambientes de computação em nuvem têm emergido como soluções eficazes para atender às necessidades de resolução de problemas por meio da web. A computação em nuvem, ao empregar técnicas de virtualização, possibilita o uso de máquinas virtuais com diversas configurações, sendo limitada apenas pelos recursos computacionais disponíveis nas máquinas físicas subjacentes. Esses ambientes funcionam como ferramentas para o compartilhamento de recursos computacionais, além de softwares e aplicações em geral (DANTAS et al., 2020).

Esses ambientes, caracterizados por alta performance, elasticidade, escalabilidade, disponibilidade, confiabilidade e baixo custo (BAUER; ADAMS, 2012), proporcionam a flexibilidade necessária para atender a diversas demandas. Além disso, são capazes de oferecer uma ampla gama de serviços, como software, plataforma e até interfaces como serviço, incluindo armazenamento de dados online, correio eletrônico e pacotes de produtividade, como o Office online. Tais características fazem da computação em nuvem uma solução cada vez mais essencial para empresas e usuários individuais, ao permitir o acesso e o gerenciamento de recursos de maneira eficiente e conveniente.

Atualmente, existem diversas soluções desse tipo fornecidas por grandes empresas, como Amazon AWS, Microsoft Azure e Google Cloud, todas oferecendo serviços de computação em nuvem terceirizados, conhecidos como nuvens públicas. Essas empresas são responsáveis por garantir a entrega de seus serviços com elevados níveis de segurança, disponibilidade, confiabilidade e desempenho. No entanto, mesmo grandes corporações podem falhar devido a sobrecargas de acessos ou falhas de software, que podem resultar de insuficiência de segurança ou de uma estimativa inadequada das necessidades computacionais do serviço.

Devido às crescentes preocupações com a segurança dos dados, muitas empresas têm se afastado do uso de serviços de computação em nuvem fornecidos por provedores públicos, optando por construir seus próprios ambientes de nuvem privados. Dessa forma, as empresas obtêm maior controle sobre os serviços e a segurança das suas informações. No entanto, esses serviços privados exigem uma alta demanda por disponibilidade e confiabilidade, uma vez que são essenciais para assegurar a continuidade das operações e a integridade dos dados. Assim, para oferecer um serviço de computação em nuvem privada, é crucial contar com recursos adequados e desempenho satisfatório, que atendam

às exigências e garantam a qualidade e a segurança do serviço prestado.

No entanto, à medida que as organizações adotam essa tecnologia, a complexidade dos sistemas em nuvem aumenta, tornando imprescindível a análise detalhada de sua arquitetura. Identificar os componentes críticos de um sistema em nuvem é uma etapa fundamental para garantir a continuidade dos serviços e a integridade dos dados. Esses componentes críticos são aqueles cuja falha pode ocasionar interrupções significativas ou perda de dados, afetando diretamente a operação e a reputação das empresas. Assim, um planejamento adequado de redundância, recuperação e monitoramento contínuo são essenciais para mitigar riscos e assegurar o desempenho e a segurança dos sistemas em nuvem.

A aplicação de redundância para componentes críticos emerge como uma estratégia eficaz para mitigar riscos e aumentar a resiliência do sistema. A redundância, que envolve a duplicação de componentes ou funções, não apenas assegura a continuidade da disponibilidade dos serviços, mas também oferece uma camada adicional de segurança contra falhas inesperadas. Este trabalho explora as razões e vantagens de realizar uma análise dos sistemas em nuvem, destacando a importância de identificar os componentes críticos e implementar estratégias de redundância. Ao abordar esses aspectos, busca-se contribuir para a construção de ambientes em nuvem mais robustos e confiáveis, capazes de atender às crescentes demandas do mercado e garantir a satisfação dos usuários.

1.1 Motivação

O hábito de permanência constante na conexão, aliado ao problema da mobilidade urbana, tem levado o ser humano a um novo nível de sedentarismo, no qual o desejo de sair de casa é suprimido pela conveniência de ter suas necessidades atendidas no conforto do lar. Consequentemente, um número crescente de indivíduos tem optado pelo uso de serviços online, como entregas, operações bancárias via web, entre outros.

Com o advento da pandemia de COVID-19, a necessidade de realizar tarefas por meio da web, como o trabalho remoto, reuniões virtuais, aulas a distância, entre outras, experimentou um impulso significativo. Esse cenário tornou ainda mais imprescindível a realização de atividades cotidianas e profissionais através da Internet. Dessa forma, reforçou-se a importância das aplicações online, que se tornam cada vez mais essenciais para a resolução de uma ampla gama de questões humanas, oferecendo praticidade e acessibilidade em diversas esferas da vida cotidiana.

Aplicações web exigem uma grande provisão de recursos computacionais, e a computação em nuvem surge como uma alternativa eficaz para atender a essas demandas. Nesse contexto, grandes empresas oferecem serviços de nuvem para suprir essa necessidade. No entanto, certas limitações de controle sobre a nuvem e questões relacionadas à segurança

têm levado organizações a adotar nuvens privadas para suas atividades.

Embora um projetista tenha controle total sobre o gerenciamento de sua nuvem, é possível que ele atinja um ponto de bloqueio, onde os recursos computacionais se mostram insuficientes e a causa dessa limitação não é imediatamente identificada. Nesse contexto, a avaliação da disponibilidade e do desempenho torna-se uma ferramenta essencial para garantir a qualidade do serviço e para identificar pontos de melhoria no sistema.

Diante disso, o presente trabalho tem como foco a avaliação das métricas de disponibilidade e desempenho do serviço Nextcloud (Nextcloud GmbH, 2024), hospedado em um ambiente de nuvem privada gerenciado pelo Apache CloudStack (APACHE, 2012), sendo consideradas múltiplas instâncias contendo o serviço. O Nextcloud é um sistema de arquivos acessível através de interfaces móveis, desktop e web. Por meio desse ambiente, alunos, professores e pesquisadores podem colaborar em tempo real em documentos, trocar mensagens, realizar videochamadas, acessar e-mails e agendar compromissos. Para garantir uma melhor distribuição das requisições entre as múltiplas instâncias do serviço, está sendo utilizado load balancing com o Nginx (F5, Inc., 2024), escolhido por sua simplicidade e eficiência na configuração e gerenciamento de tráfego. Outros possíveis serviços que poderiam ser utilizados são: Seafile, SparkleShare, FileCloud, Filestash e Pingvin. Os estudos de caso apresentados têm como objetivo mensurar essas métricas em diferentes cenários, que representam variações nas características do ambiente ou do serviço avaliados. O intuito é fornecer subsídios aos projetistas para que possam estimar as vantagens das diferentes configurações do sistema, mesmo antes de colocá-lo em produção.

1.2 Objetivos

O principal objetivo deste trabalho é a proposição de uma estratégia baseada em modelos para a avaliação de desempenho e disponibilidade de um serviço hospedado em um ambiente de nuvem privada, considerando múltiplas instâncias do serviço e balanceamento de carga entre elas. Para atingir esse objetivo principal, foram estabelecidos os seguintes objetivos específicos:

- Propor uma estratégia para aferir o desempenho de um serviço hospedado em nuvem privada através de um conjunto métricas, como vazão e tempo de resposta.
- Propor modelos de desempenho para representar um serviço hospedado em nuvem.
- Propor modelos de disponibilidade que represente o ambiente de nuvem privada.
- Propor um modelo combinado que avalie o impacto da disponibilidade no desempenho de um serviço hospedado em nuvem privada.

- Propor abordagens basedas em projeto de experimentos e índice da importância da disponibilidade para identificar os componentes de maior impacto no sistema.
- Propor extensões para os modelos criados, considerando redundâncias para os componentes mais relevantes, as quais representam maior robustez no serviço e/ou ambiente de nuvem e apresentam melhorias para a disponibilidade e o desempenho do sistema.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 2 expõe os conceitos básicos necessários para uma compreensão aprofundada deste estudo. o Capítulo 3 apresenta os trabalhos relacionados encontrados na literatura. O Capítulo 4 descreve a metodologia utilizada e a arquitetura do ambiente de testes. O Capítulo 5 discute os modelos propostos e suas extensões. O Capítulo 6 apresenta os estudos de caso relacionados ao modelo. Por fim, o Capítulo 7 conclui o trabalho, mostra as contribuições e aponta os futuros direcionamentos desta pesquisa.

2 Referencial Teórico

Este capítulo busca proporcionar uma breve explanação sobre alguns conceitos, os quais são fundamentais para o entendimento deste trabalho.

2.1 Computação em Nuvem

A computação em nuvem pode ser definida como um modelo que permite acesso a uma rede compartilhada com um conjunto de recursos computacionais (como hardware, software, armazenamento, rede, aplicações e serviços) de maneira ubíqua, conveniente e sob demanda (CLOUD, 2011). Podendo ser adquiridos rapidamente e liberados com mínimo esforço gerencial ou interação com o provedor de serviços. De forma genérica, a infraestrutura de nuvem pode ser descrita como um conjunto de hardware e software ajustados de forma a abstrair dos usuários a complexidade e organização da infraestrutura computacional. Desenvolvido com o objetivo de fornecer serviços de baixo custo e de fácil acesso, bem como de garantir características tais como disponibilidade e escalabilidade (RUSCHEL; ZANOTTO; MOTA, 2010).

Dentre as características oferecidas pela computação em nuvem, as mais essenciais são: serviço sob demanda, amplo acesso pela rede, agrupamento de recursos, elasticidade rápida e mensuração de serviço. Além dessas, existem outras características são desejáveis por impactarem diretamente na disponibilidade e no desempenho dos serviços ofertados, como escalabilidade, segurança, resiliência e (ARAUJO et al., 2014). Algumas modalidades de serviços de computação em nuvem que existem são:

- Infraestrutura como Serviço (Infrastructure as a Service – IaaS) - modelo de serviço onde são fornecidos recursos de computação como hardware, armazenamento e rede. Neste modelo o usuário tem controle sobre qual sistema operacional instalar, o armazenamento e os aplicativos. Alguns exemplos de IaaS são o Apache CloudStack, Amazon EC2 e Google Cloud.

- Plataforma como Serviço (Platform as a Service – PaaS) - modelo de serviço onde ocorre uma espécie de aluguel de um sistema operacional, no qual podem ser instalados aplicativos, sendo seu foco o desenvolvimento e execução de aplicações. Os exemplos de PaaS são Google Cloud Plataform, Microsoft Azure e Red Hat OpenShift.

- Software como Serviço (Software as a Service – SaaS) - modelo de serviço em que é fornecido um aplicativo executado em uma infraestrutura de nuvem, sendo utilizados

no próprio navegador sem a necessidade de instalação. O usuário pode apenas modificar as possíveis configurações do aplicativo. Os exemplos de SaaS são Google Apps, Dropbox e Microsoft Office 365.

Existem quatro tipos de modelos de computação em nuvem (CLOUD, 2011), sendo elas:

- Nuvem pública: o modelo comum de computação em nuvem, onde grandes empresas donas de uma infraestrutura em nuvem oferecem de forma gratuita ou paga serviços de nuvem ao público geral.
- Nuvem privada: similar a nuvem publica, mas neste modelo a infraestrutura em nuvem é dedicada exclusivamente para uma organização, sendo a melhor opção para negócios dinâmicos mas requer controle direto sobre seus ambientes.
- Nuvem comunitária: modelo onde várias organizações que possuem políticas, objetivos e preocupações similares compartilham uma infraestrutura de nuvem, agindo como uma infraestrutura multi-locatário.
- Nuvem híbrida: modelo cuja infraestrutura da nuvem é formada por uma composição de dois ou mais modelos de nuvens (privada, pública e comunitária). Sendo as atividades sensíveis executadas na parte privada, enquanto as não sensíveis na parte pública.

2.2 Apache CloudStack

O Apache CloudStack é um software de código aberto cuja finalidade é a implantação e gerenciamento grandes redes de máquinas virtuais sob a forma de um IaaS altamente escalável e disponível (APACHE, 2012). Criado em 2008 por uma start-up com o nome inicial de VMOps. Já em 2010, adquiriu o nome Cloud.com e foi lançado o CloudStack. Sendo vendido em 2011 para Citrix Systems, que no ano seguinte doou para Apache Software Foundation. Usado por grandes empresas como Dell, Huawei e Walt Disney. Os recursos dentro da infraestrutura em nuvem do Apache CloudStack são:

- Regiões: uma coleção de uma ou mais zonas geograficamente próximas gerenciadas por um, ou mais servidores de gerenciamento.
- Zonas: é equivalente a um único datacenter. Uma zona consiste em uma ou mais pods e armazenamento secundário.
- Pods: um pod geralmente é um rack ou fileira de racks que inclui um switch de camada 2 e um ou mais clusters.

- Clusters: um cluster consiste em um ou mais hosts homogêneos e armazenamento primário.
- Hosts: um único nó de computação dentro de um cluster; muitas vezes um hipervisor.
- Armazenamento Primário: um recurso de armazenamento fornecido a um único cluster para a execução real de imagens de disco da instância.
- Armazenamento Secundário: um recurso de toda a zona que armazena modelos de disco, imagens ISO e instantâneos.

A Figura 1 ilustra um exemplo da arquitetura do Cloudstack. Sendo formada por uma máquina contendo o gerenciador Cloudstack com um banco de dados, outra máquina fica responsável pelo armazenamento primário e armazenamento secundário, enquanto a última máquina será responsável pela virtualização, que no caso do exemplo é utilizado um hypervisor KVM. O KVM é usado para virtualizar as três máquinas virtuais de sistema que são: Roteador Virtual (Virtual Router - VR), Máquina Virtual de Console de Proxy (Console Proxy Virtual Machine - CPVM) e Máquina Virtual de Armazenamento Secundário (Secondary Storage Virtual Machine - SSVM). Sendo responsáveis o VR pela comunicação de rede entre as instâncias e a internet, a CPVM pelo acesso ao console das instâncias via web e a SSVM pelo armazenamento de templates e ISOs de sistemas operacionais.

2.3 Balanceamento de Carga

Balanceamento de carga é uma peça fundamental na melhoria de performance de ambientes em nuvem e sistemas distribuídos. Sendo um mecanismo cuja função é redistribuir a carga de trabalho sobre os recursos de um sistema para os recursos equivalentes de um ou mais outros nós de uma rede sem que nenhuma tarefa sendo executada seja eliminada (MORE; HIRAY, 2012). Sendo assim, o balanceamento de carga entre vários nós de um sistema de nuvem se torna um grande desafio, pois a carga pode ser de vários tipos como rede, memoria, CPU, etc. Logo é de suma importância que a divisão da carga entre os múltiplos nós seja a melhor possível para evitar perda de performance.

Dentre os objetivos do balanceamento de carga temos:

- Manter a estabilidade do sistema.
- Melhorar a performance e a eficiência.
- Minimizar o tempo de espera.
- Melhorar a distribuição do uso de recursos.

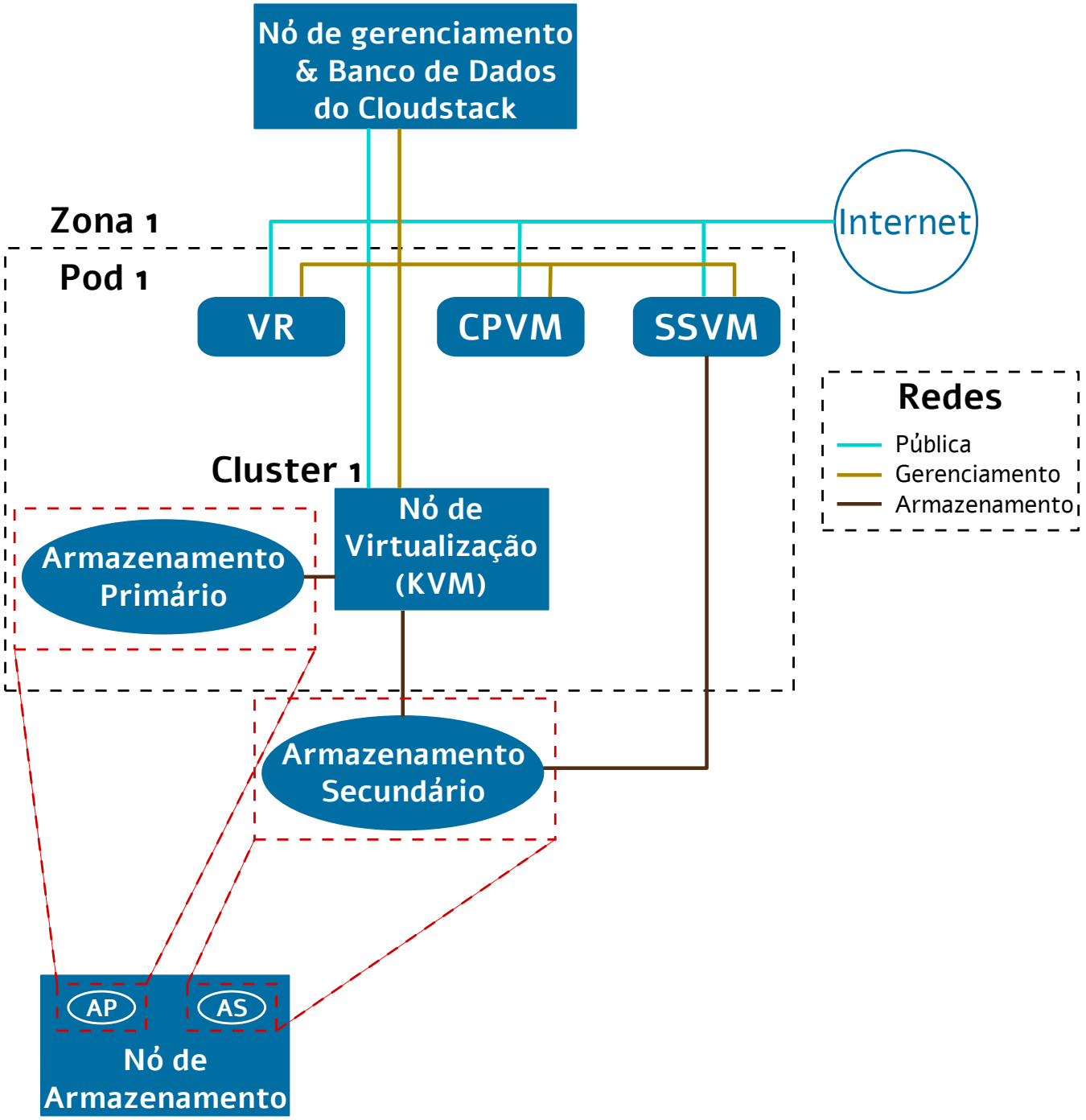

Figura 1 – Exemplo de Arquitetura do Cloudstack

Existem basicamente dois tipos de algoritmos de balanceamento de carga. O algoritmo de balanceamento estático – este tipo de algoritmo de balanceamento requer um conhecimento prévio sobre os recursos do sistema. Entre eles a capacidade de processamento, memória e armazenamento. A decisão da distribuição da carga não depende do estado atual do sistema (DESAI; PRAJAPATI, 2013), não alterando os processos ativos em tempo de execução para fazer alterações na carga do sistema. Entretanto não são naturalmente flexíveis para lidar com mudanças dinâmicas nos recursos do sistema durante o tempo de execução. Sendo este tipo de algoritmo mais indicado para sistemas cujos ambientes sejam

estáveis e homogêneos.

Algoritmo de balanceamento dinâmico – este tipo de algoritmo de balanceamento é mais flexível e não requer nenhuma informação prévia sobre os recursos do sistema, visto que a distribuição da carga é baseada no estado atual do sistema em tempo de execução (DESAI; PRAJAPATI, 2013). Sendo indicado para sistemas heterogéneos, este tipo de algoritmo faz escolhas em tempo de execução, trazendo melhorias na performance se comparado com o algoritmo estático. Entretanto, conforme os atributos de balanceamento se tornam mais complexos, acabam por se tornar ineficientes e gerar sobrecarga mais que o necessário, o que culmina numa perda de desempenho.

2.4 Rede de Petri

O conceito de redes de Petri teve sua origem em 1962, com Carl Adam Petri, em sua tese de doutorado na Faculdade de Matemática e Física da Universidade de Darmstadt, na Alemanha (PETRI, 1962). Sua teoria original foi concebida e desenvolvida como uma abordagem para a modelagem e análise de sistemas de comunicação. Redes de Petri são uma ferramenta de modelagem gráfica e matemática aplicável a diversos sistemas caracterizados por propriedades como concorrência, assincronicidade, distribuição, paralelismo, não determinismo e/ou estocasticidade (MURATA, 1989).

Diversos pesquisadores, com base nesse trabalho inicial, começaram a explorar novas representações e extensões para as redes de Petri originais. Essas ideias visavam permitir descrições mais precisas e a representação de características de sistemas que não podiam ser capturadas pelos modelos iniciais. A rede de Petri simples, a partir dessas propostas, deu origem a adaptações e extensões em diversas direções, com exemplos notáveis sendo as redes estocásticas, temporizadas, orientadas a objetos, de alto nível e coloridas.

As Redes de Petri de Lugar/Transição representam uma das propostas de extensão da rede de Petri original (DESEL; REISIG, 1996). Esta classe de redes de Petri é uma das mais relevantes e amplamente estudadas, sendo frequentemente referida simplesmente como rede de Petri. Uma rede de Petri de Lugar/Transição é um grafo direcionado bipartido. De acordo com (MURATA, 1989), uma Rede de Petri é definida formalmente como uma 5-tupla $P_N = \{P, T, F, W, M_0\}$, onde:

- P = p1, p2, ..., pn é um conjunto finito de lugares;
- T = t1, t1, ..., tn é um conjunto finito de transições;
- F $\subseteq$ (P $\times$ T) $\cup$ (T $\times$ P) é um conjunto de arcos;
- W : F $\rightarrow$ 1, 2, 3, ... é uma função de peso do arco;

- M0 : P $\rightarrow$ 0, 1, 2, 3, ... é a marcação inicial, P $\cap$ T = $\emptyset$ e P $\cup$ T $\neq \emptyset$

A rede de Petri pode ser decomposta em elementos fundamentais. Os lugares correspondem às variáveis que representam os possíveis estados do sistema. As transições são ações que podem ser executadas e cuja ativação exige que suas pré-condições individuais sejam atendidas, resultando em uma alteração no estado do sistema. Os arcos conectam os lugares e as transições, representando o fluxo de tokens ao longo do sistema. Por sua vez, o arco inibidor permite que o modelo verifique a presença de tokens no lugar de origem, inibindo a ativação da transição conectada. A transição só poderá ser ativada quando não houver tokens no lugar de origem (SILVA, 2016). Os tokens simbolizam os recursos disponíveis no sistema, os quais são consumidos e gerados pelas ações do sistema (transições). A distribuição dos tokens na rede reflete o estado atual do sistema. Graficamente, os lugares são representados por círculos, as transições por barras, os arcos por setas e os tokens por pontos. A Figura 2 ilustra os elementos constituintes de uma rede de Petri.

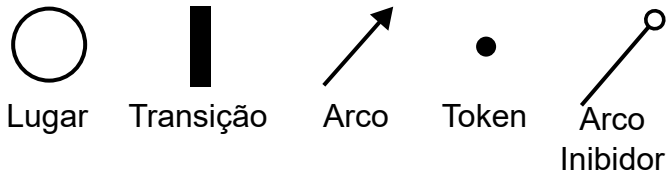

Figura 2 – Elementos de uma PN

2.4.1 Rede de Petri Estocástica

Redes de Petri constituem uma ferramenta para a modelagem e análise de eventos em sistemas cuja complexidade impede a descrição por meio de autômatos ou modelos de filas. O tempo (demora estocástica) e as escolhas probabilísticas representam aspectos essenciais para a avaliação de modelos de desempenho. Contudo, o conceito original de redes de Petri, concebido para a modelagem do comportamento lógico do sistema, descrevendo as relações entre os eventos possíveis, não contempla a noção de tempo.

Diversos pesquisadores, ao longo do tempo, propuseram diferentes abordagens para incorporar a definição de tempo às redes de Petri. As redes de Petri estocásticas (Stochastic Petri Nets – SPN) são uma das extensões que incorporam o conceito de tempo em seus modelos. As SPNs são amplamente utilizadas na modelagem de desempenho e dependabilidade de sistemas.

As SPNs apresentam dois conceitos que não estão presentes nas Redes de Petri convencionais: a transição estocástica e o a transição imediata. A transição estocástica está associada a um tempo, o qual segue uma distribuição exponencial. Esse tempo representa o intervalo necessário até que a transição esteja habilitada, correspondendo

ao período de execução da atividade. Enquanto que a transição imediata possui o tempo zero. Sendo a representação de atividades que são concluidas instantaneamente, não possuindo um tempo de execução. A Figura 3 apresenta os elementos de uma Rede de Petri, juntamente com os elementos adicionais de uma SPN. Uma SPN é definida como uma 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$ (GERMAN, 2000), onde:

- $P = \{p_1, p_2, ..., p_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é um conjunto finito de transições;
- $I \in (N^n \to N)^{m \times n}$ é a matriz que representa a multiplicidade dos arcos de entrada (que podem ser dependentes de marcações);
- $O \in (N^n \to N)^{m \times n}$ é a matriz que representa a multiplicidade dos arcos de saída (que podem ser dependentes de marcações);
- $H \in (N^n \to N)^{m \times n}$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações).
- $\Pi \in N^m$ é um vetor vincula o nível de prioridade para cada transição;
- $G \in (N^n \to \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada à marcação do lugar a cada transição;
- $M_0 \in N^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, W, Markdep, Política, Concorrência)^m$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial), onde:
  - $Dist \in N^m \to f$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, onde o domínio de f é $[0, \infty)$ (esta distribuição pode ser dependente de marcação);
  - $W \in N^m \to R^+$ é uma função de peso (esta distribuição pode ser dependente de marcação);
  - $Markdep \in \{constant, enabdep\}$ é definido como a distribuição de probabilidade associada ao tempo de uma transiçãao constante (constant) ou dependente de marcação (enabdep);
  - $\circ = Política \in \{prd, prs\}$ é a política de memória adotada pela transição (prd - *preemptive repeat different*, corresponde a race enabling policy; prs - *preemptive resume*, corresponde a age memory policy).
  - $Concorrência \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica de *single server* e is representa a semântica de *infinity server*.

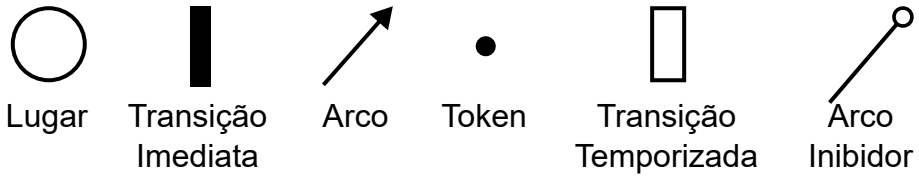

Figura 3 – Elementos de uma SPN

A Figura 4 ilustra um exemplo de SPN. Nela, modela-se o funcionamento de um sistema, no qual os lugares representam os estados do sistema (ligado ou desligado) e as transições representam as ações que alteram o estado do sistema (ligar e desligar). A parte (a) da figura mostra a representação do sistema quando está ligado (com um token no lugar "On"). Nesse estado, a única transição habilitada para ser disparada é a transição "Desligar". Após a ativação dessa transição, o modelo transita para o estado desligado (com um token no lugar "Off"), conforme ilustrado na parte (b) da figura. Em seguida, a transição "Ligar"torna-se habilitada, e sua ativação retorna o sistema ao estado de ligado. A Equação 2.1 apresenta a expressão usada para obter a disponibilidade nesse exemplo.

$$D = P\{(\#VM\_on = 1)\} \tag{2.1}$$

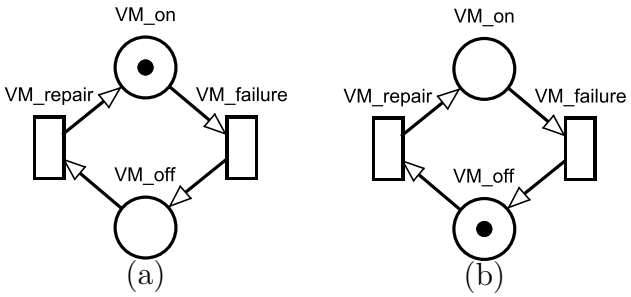

Figura 4 – Exemplo de SPN

2.5 Diagramas de Bloco de Confiabilidade

Os Diagramas de Bloco de Confiabilidade (Reliability Block Diagram - RBD) surgiram inicialmente como um modelo combinatório proposto como uma técnica para calcular a confiabilidade de um sistema. Esse modelo técnico foi posteriormente ampliado, com extensões que permitiram a utilização dos RBDs também no cálculo de outras métricas de dependabilidade, como disponibilidade e manutenibilidade (SAHNER; TRIVEDI; PULIAFITO, 1997).

Esses diagramas de blocos fornecem uma descrição gráfica dos componentes do sistema e das conexões entre eles. A estrutura do RBD estabelece a interação lógica entre os componentes, com o sistema sendo representado por subsistemas ou componentes

conectados de acordo com suas funções ou relacionamento de confiabilidade (XIE; DAI; POH, 2004).

Em um RBD, um componente físico é representado por um bloco, caso este esteja operacional no sistema. Para representar a falha de um componente, é necessário remover o bloco correspondente ao componente que entrou em falha. Com base nessa estrutura, é possível determinar o estado geral do sistema, o qual é considerado operacional se, e somente se, houver um conjunto de blocos ativos que permita a conexão entre a entrada e a saída do sistema.

Para determinar o estado do sistema, é necessário, primeiramente, analisar o estado de cada componente que pertence a esse sistema e, em seguida, identificar quais combinações de componentes ativos ou em falha são capazes de permitir que o sistema permaneça em operação. Caso haja uma insuficiência de blocos ativos suficientemente grande para interromper a conexão entre os pontos de entrada e saída, o sistema entrará em falha (DISTEFANO; SCARPA; PULIAFITO, 2006).

2.5.1 Tipos de composição

A seguir, serão apresentadas as formas como os componentes de um sistema podem estar conectados. Em um RBD, os componentes são interligados entre si por diferentes mecanismos de composição, podendo estar dispostos em série, paralelo, k-out-of-n, ponte ou qualquer combinação dessas estruturas (TRIVEDI et al., 1996).

A Figura 5 apresenta um modelo em RBD de um sistema em série, no qual o sistema só estará em funcionamento se todos os componentes estiverem ativos.

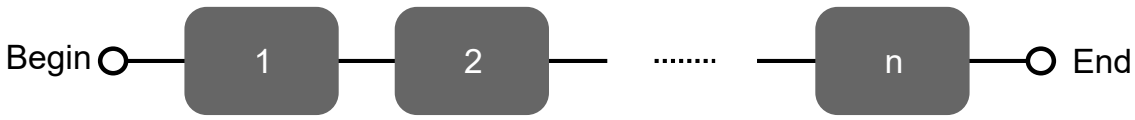

Figura 5 – RBD em série

A Figura 6 apresenta um modelo em RBD de um sistema em paralelo, no qual o sistema estará em funcionamento se, pelo menos, um dos componentes estiver ativo.

Um sistema em k-out-of-n estará em funcionamento se, e somente se, k ou mais de seus n componentes estiverem ativos. Os sistemas em série e paralelo são casos particulares de sistemas k-out-of-n. Sistemas em série correspondem ao caso n-out-of-n (onde todos os componentes do sistema devem estar funcionando), enquanto sistemas em paralelo representam o caso 1-out-of-n (onde pelo menos um componente do sistema deve estar funcionando).

A Figura 7 apresenta um modelo em RBD de um sistema 2-out-of-3. O modelo ilustrado na figura indica que, se pelo menos dois dos três componentes estiverem funcionando

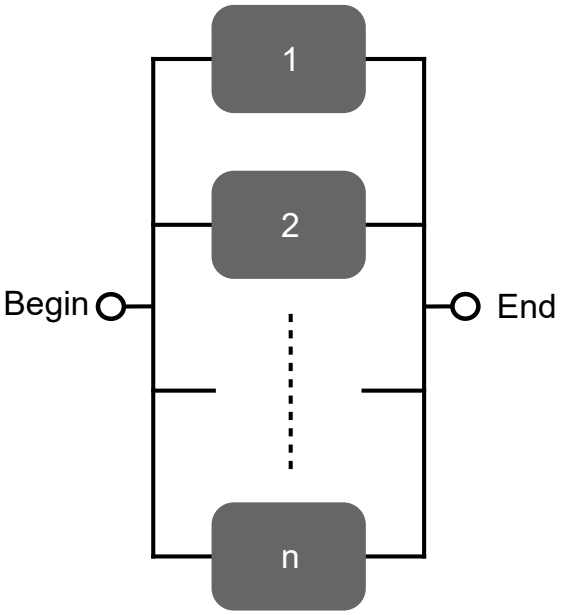

Figura 6 – RBD em paralelo

(por exemplo: 1 e 2, ou 1 e 3, ou 2 e 3), o sistema estará em funcionamento.

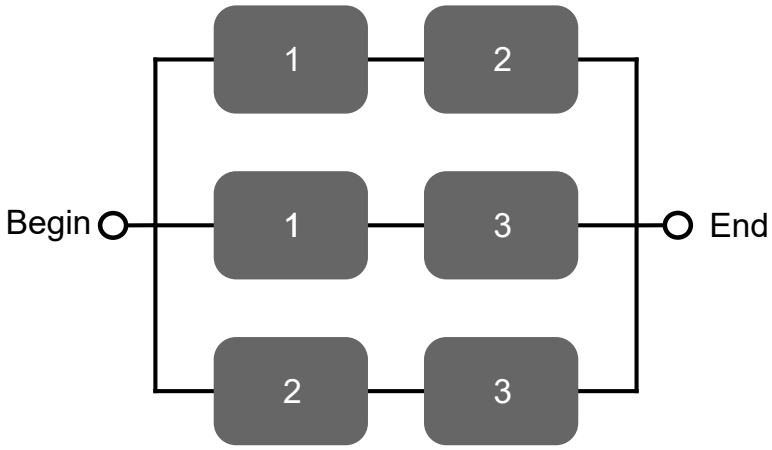

Figura 7 – RBD em k-out-of-n

A parte (a) da Figura 8 apresenta um modelo em RBD de um sistema em ponte. Nesse tipo de sistema, é considerado um modelo em série, no qual alguns componentes apresentam redundância em paralelo. Um sistema em ponte só se encontra operacional quando pelo menos certos componentes específicos estiverem funcionando (por exemplo: (1,3,5); (1,4); (2,3,4); (2,5)). Esse sistema pode ser reescrito como um modelo equivalente que apresenta todos os caminhos possíveis, como mostrado na parte (b) da Figura 8.

2.5.2 Funções de Métricas

Para avaliar as métricas de dependabilidade, como confiabilidade ou disponibilidade, o RBD utiliza fórmulas matemáticas internas para calcular os resultados. O cálculo das

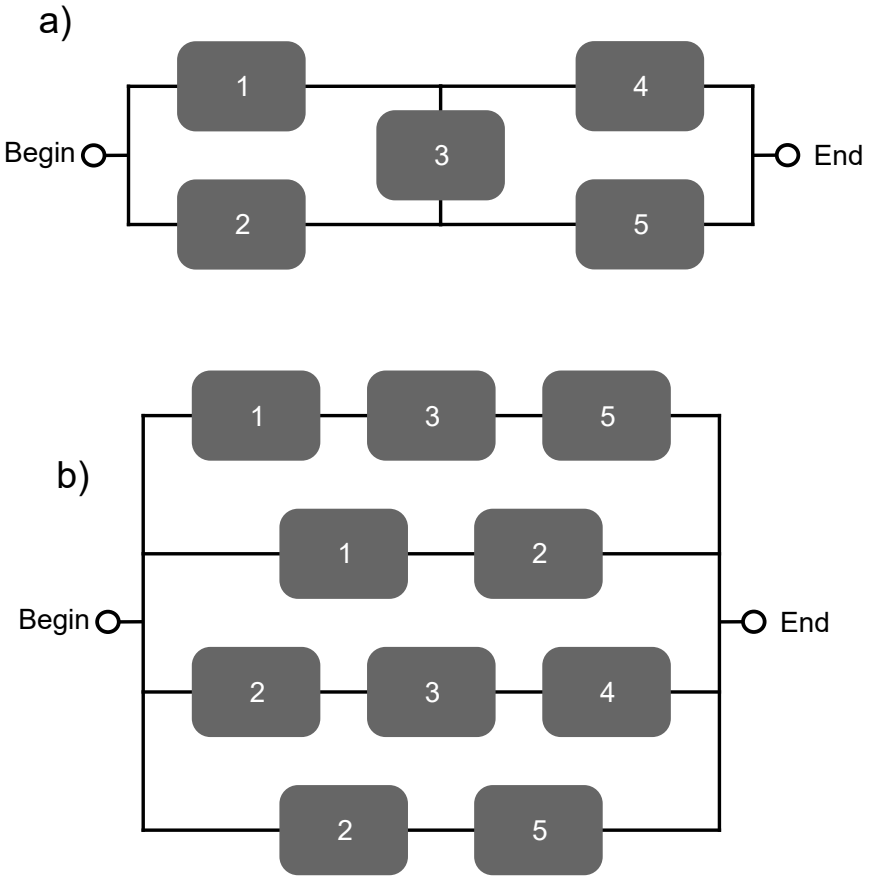

Figura 8 – RBD em ponte

métricas para blocos conectados em série (Rs), considerando n componentes, é obtido por meio da Equação 2.2:

$$Rs = \prod_{i=1}^{n} R_i \tag{2.2}$$

As métricas de blocos conectados em paralelo (Rp), considerando n componentes, são obtidas por meio do cálculo da Equação 2.3.

$$Rp = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{2.3}$$

Onde, em ambas as equações, R_i se refere à confiabilidade $R_i(t)$, à disponibilidade instantânea $A_i(t)$ ou à disponibilidade estacionária A_i do bloco i.

Normalmente, os sistemas são compostos por um arranjo que inclui partes em série e paralelo. Considerando o sistema apresentado na Figura 9, para calcular a disponibilidade do sistema, o RBD pode analisar o sistema parte por parte. Por exemplo, pode-se começar com o subsistema A (composto pelos componentes 2 e 3, que estão em paralelo), seguir para o subsistema em série B (analisando a região A e o componente 1), e depois para o subsistema C (composto pela região B e o componente 4, que estão em paralelo). Por fim,

o sistema inteiro pode ser analisado para calcular a disponibilidade total do sistema. A disponibilidade do sistema é calculada através da Equação 2.4:

$$\begin{aligned} Av_A &= 1 - (1 - Av_2) \times (1 - Av_3) \\ Av_B &= (Av_A) \times (Av_1) \\ Av_C &= 1 - (1 - Av_B) \times (1 - Av_4) \\ Av_S &= (Av_C) \times (Av_5) \end{aligned} \tag{2.4}$$

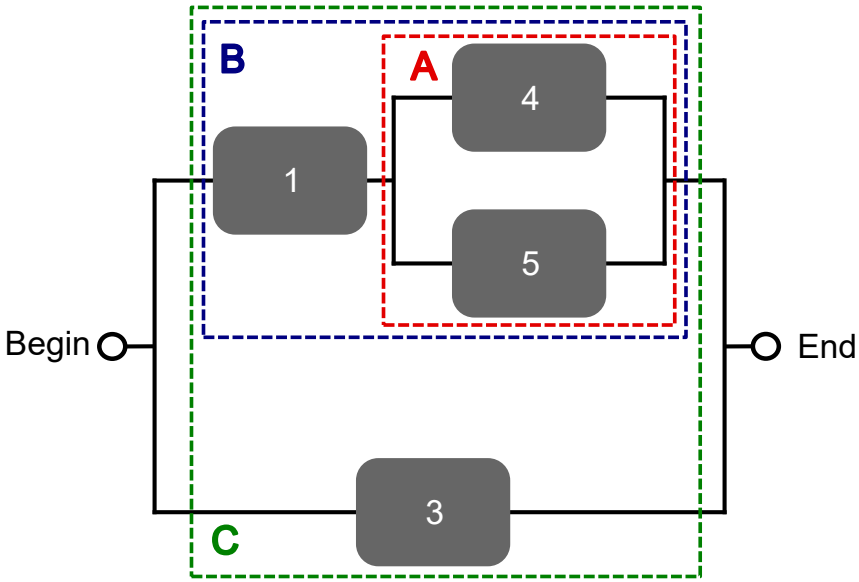

Figura 9 – RBD Combinando serie e paralelo

2.5.3 Níveis de Redundância

A redundância do sistema pode ser alcançada por meio de duas abordagens distintas: (i) redundância em baixo nível e (ii) redundância em alto nível. O primeiro caso envolve a utilização de um ou mais componentes em paralelo, enquanto a redundância em alto nível replica o sistema inteiro. Por exemplo, considere um sistema composto por dois componentes em série (1 e 2), no qual foi adotada redundância em baixo nível, conforme apresentado na Figura 10. De forma semelhante, a Figura 11 ilustra dois componentes em redundância de alto nível.

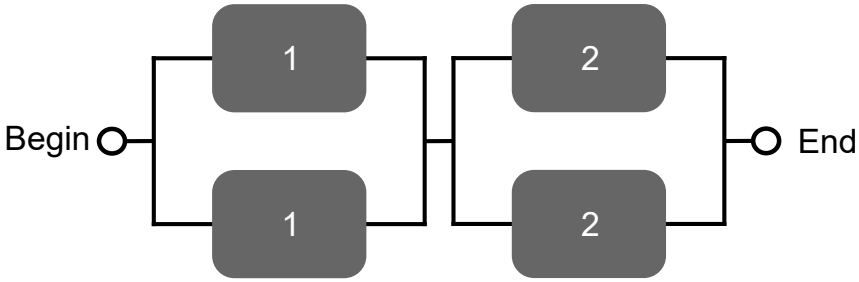

Figura 10 – RBD em redundância em baixo nível

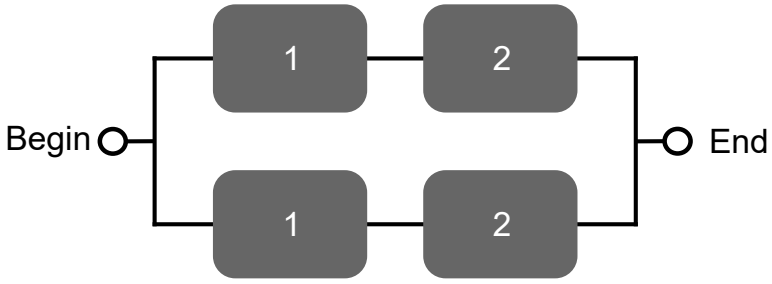

Figura 11 – RBD em redundância em alto nível

Uma questão interessante diz respeito a qual método de redundância oferece a maior confiabilidade. Não há dúvidas de que, em ambas as abordagens de redundância, caso ambos os componentes 1 e 2 falhem, o sistema também falhará. Contudo, é importante observar que, no sistema com redundância de alto nível, o sistema pode falhar caso um dos componentes 1 e um dos componentes 2 falhem. Esse comportamento não ocorre no caso da redundância em baixo nível. Portanto, sistemas com redundância em baixo nível possuem maior confiabilidade do que os sistemas com redundância em alto nível.

2.6 Avaliação de Desempenho

A Avaliação de Desempenho é o processo de coleta sistemática de dados, com base em critérios estabelecidos previamente, com o objetivo de apoiar a análise fundamentada em evidências (ROGERS; BADHAM, 1994) sobre o desempenho do que foi observado. Para se identificar a capacidade do sistema e saber o que fazer em momentos de sobrecarga do sistema, se faz necessário o conhecimento sobre as principais métricas de desempenho (BARROS et al., 2018). Dentre as métricas de desempenho as principais são: vazão, tempo de resposta, uso de memória e CPU, taxa de leitura e escrina em HD.

O processo de avaliação de desempenho pode incluir estatística, medição, modelagem, simulação, experimentos, etc. Segundo (CALLOU et al., 2011), a avaliação de desempenho pode ser dividida em medição e modelagem, como pode ser visto na Figura 12. Medição é o forma mais direta para avaliar o desempenho. A partir dela se pode conseguir resultados mais precisos, pois adquire os dados ao medir o ambiente real do sistema. Apesar de fornecerem dados exatos quanto ao desempenho do sistema, as técnicas de medição não podem ser utilizadas se o sistema ainda estiver sendo construído (TORRES et al., 2016).

A modelagem é classificada em analítica e simulação. Na modelagem analítica é feita uma abstração do sistema, gerando um modelo contendo as características comportamentais essenciais do sistema. Já na simulação o objetivo é realizar a execução do modelo reproduzir o comportamento característico do sistema que ele representa. Apesar dos resultados da simulação do modelo serem menos precisos, que aqueles fornecidos pela medição, estimativas podem ser calculadas. Sendo necessário comparar se os dados trazidos pelo modelo estão

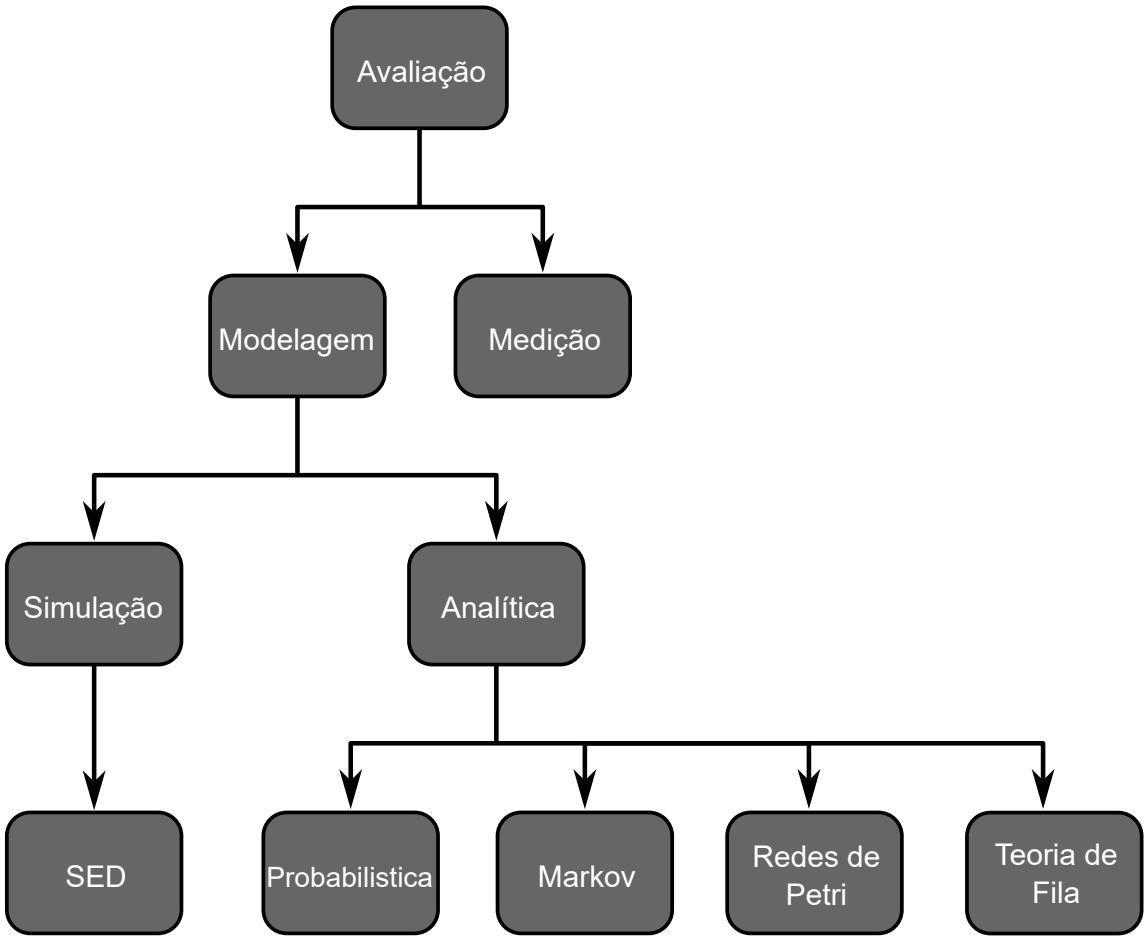

Figura 12 – Técnicas Avaliação de Desempenho

dentro dos intervalos de confiança dos valores medidos no sistema real, assim validando que o modelo representa o sistema.

2.7 Analíse de Sensibilidade

O termo Análise de Sensibilidade ainda não possui uma definição unificada na literatura. Segundo (KLEIJNEN, 1995), a análise de sensibilidade pode ser descrita como uma investigação sistemática dos resultados de simulações quando expostos a valores extremos nas entradas do modelo ou a mudanças significativas em sua estrutura. Já (FRANK; ESLAMI, 1980) define a análise de sensibilidade como um conjunto de técnicas empregadas para identificar quais componentes têm maior influência sobre uma métrica específica do sistema.

Os autores em (CRICK; HILL, 1987) classificam as técnicas utilizadas na análise de sensibilidade em dois tipos principais:

- Parâmetros importantes: São aqueles cuja incerteza contribui de forma significativa para a variação dos resultados do modelo.

- Parâmetros sensíveis: Refere-se aos parâmetros que possuem uma influência predominante nos resultados da avaliação.

A literatura descreve diversas técnicas para a avaliação de sensibilidade, sendo que cada uma delas pode gerar resultados distintos. Durante o processo de análise, o analista classifica os parâmetros de entrada de acordo com sua sensibilidade em relação à métrica a ser avaliada. Identificar quais parâmetros são mais significativos é uma tarefa fundamental. No entanto, a diferenciação na classificação dos parâmetros de menor importância entre as diferentes técnicas de análise de sensibilidade não se revela uma preocupação prática relevante, conforme mencionado por (HAMBY, 1994).

Para realizar uma análise de sensibilidade de maneira clara e objetiva, segundo (IOOSS; LEMAÎTRE, 2015), deve-se seguir os seguintes passos:

- Identificar e priorizar as variáveis de entrada mais influentes.
- Determinar as variáveis de entrada não influentes para fixá-las em valores nominais.
- Mapear o comportamento da métrica em relação aos valores de entrada, especificando um domínio de entrada.
- Calibrar algumas variáveis de entrada utilizando informações já disponíveis ou restrições do sistema.

O Projeto de Experimentos (Design of Experiments – DoE) é uma técnica de análise de sensibilidade amplamente reconhecida na literatura e utilizada em vários estudos. Ele pode ser definido de diversas formas, mas, essencialmente, é uma metodologia voltada para o planejamento de experimentos com o objetivo de fornecer resultados adequados para uma análise estatística rigorosa, permitindo, assim, alcançar conclusões objetivas e confiáveis (MONTGOMERY, 2017).

De forma mais detalhada, DoE é descrito como um conjunto de técnicas estatísticas que possibilitam uma compreensão aprofundada sobre o produto ou processo sob análise, oferecendo insights sobre como as variáveis ou fatores de entrada afetam os resultados ou saídas do sistema em questão (KLEIJNEN, 1995). Além disso, DoE pode ser visto como uma sequência de experimentos nos quais o pesquisador ajusta variáveis ou fatores de entrada para observar e identificar quais mudanças nesses fatores resultam em variações nas saídas (MONTGOMERY; RUNGER, 2010).

A técnica DoE é formada por um conjunto de elementos (MASON; GUNST; HESS, 2003):

- fatores – se referem a um variável controlável que influência o resultado.

Tabela 1 – Exemplo Design Fatorial

Caso	A	B
t_1	A_{baixo}	B_{baixo}
t_2	A_{baixo}	B_{alto}
t_3	A_{alto}	B_{baixo}
t_4	A_{alto}	B_{alto}

- níveis – os diversos valores atribuídos aos fatores.
- efeitos – são as variações nas medições dos resultados.

No Design of Experiments (DoE), um dos aspectos mais cruciais é a análise dos efeitos resultantes de diferentes combinações de níveis de fatores, de forma a identificar como essas interações impactam o resultado final do experimento. O design fatorial é uma das abordagens mais comuns e eficientes para lidar com experimentos que envolvem dois ou mais fatores.

No contexto de um design fatorial, os fatores são tratados como fatores cruzados. Isso significa que, ao combinar os fatores em diferentes níveis, o analista é capaz de observar como as interações entre os fatores influenciam os resultados. Essa abordagem permite uma análise abrangente e detalhada das relações entre os fatores, ajudando a identificar os efeitos de cada fator individualmente, bem como as interações entre eles.

O efeito em um experimento de DoE refere-se à mudança nos resultados (ou saídas) do sistema, que ocorre quando há variação nos níveis de um fator. A principal vantagem do design fatorial é sua capacidade de avaliar não só o impacto de cada fator isoladamente, mas também como os fatores combinados afetam o desempenho do sistema. Isso é fundamental, pois as interações entre fatores podem muitas vezes gerar efeitos inesperados ou não intuitivos, que não seriam percebidos em uma análise unidimensional.

Ao usar o design fatorial para considerar as interações entre os fatores, é possível evitar conclusões equivocadas que poderiam ser geradas se apenas os efeitos isolados dos fatores fossem analisados. Essa abordagem amplia significativamente a precisão e a validade das conclusões tiradas dos experimentos (MONTGOMERY, 2017).

Se um efeito está diretamente associado aos principais fatores utilizados em um experimento, esse efeito é considerado um efeito principal daquele experimento. A Tabela 1 apresenta um exemplo de design fatorial com dois fatores (A e B) nos níveis alto e baixo. O efeito principal é calculado como a diferença entre a média dos níveis mais altos e baixos de cada fator, conforme ilustrado nas Equações 2.5 e 2.6.

$$A = \frac{t_1 + t_2}{2} - \frac{t_3 + t_4}{2} \tag{2.5}$$

$$B = \frac{t_1 + t_3}{2} - \frac{t_2 + t_4}{2} \tag{2.6}$$

2.8 Disponibilidade

A disponibilidade pode ser descrita como a probabilidade de que o sistema analisado esteja operacional durante um determinado período de tempo (MACIEL; TRIVEDI; KIM, 2010). A disponibilidade é obtida através da divisão do tempo de funcionamento pela soma do tempo de funcionamento com o tempo de falha. A Equação 2.7 traz esse cálculo. Sendo E[Uptime] o tempo de funcionamento do sistema e E[Downtime] o tempo em que o sistema está em falha.

$$A = \frac{E[Uptime]}{E[Uptime] + E[Downtime]} \tag{2.7}$$

Outra forma possível de se calcular a disponibilidade é a partir do somatório dos tempos de falha e dos tempos de reparo dos dispositivos que compõem o sistema. Sendo utilizados para o cálculo da disponibilidade conforme a Equação 2.8, onde MTTF (Mean Time to Failure) representa o tempo médio para ocorrer falhas no sistema e MTTR (Mean Time to Repair) representa o tempo médio para reparo do sistema.

$$D = \frac{MTTF}{MTTF + MTTR} \tag{2.8}$$

Devido ao fato de ambientes de nuvem demandam alta disponibilidade, os valores obtidos dessa métrica tendem a resultados acima de 99%. Entretanto, esses valores acabam sendo muito próximos entre si, o que acaba por dificultar a compreensão de quão bom um resultado pode ser em relação aos outros. Para resolver esse problema é comum que alguns trabalhos, assim como (ROSENDO et al., 2019), trabalharem com a disponibilidade convertida em número de noves. A Equação 2.9 apresenta o cálculo dessa converção, onde D representa a disponibilidade préviamente calculada. Nela são quantificados os algarismos consecutivos iguais a 9, que se encontram após a vírgula e assim trazendo uma melhor apresentação do resultado.

$$D_{9's} = -log_{10}(1 - D) \tag{2.9}$$

Existem diferentes formas de se aumentar a disponibilidade, as quais incluem a adição de redundâncias ao sistema. Sendo os dois tipos clássicos de redundância *hot standby* e *cold standby*.

2.8.1 Hot Standby

Na técnica de redundância *hot standby* os equipamentos redundantes que estão neste modo de operação sempre permanescem ativos e em sincronia com o equipamento principal. Participando do processo da mesma maneira que o principal, e em alguns casos compartilhando a carga. Logo podem assumir a carga dos equipamentos principais, caso estes entrem em falha e vice-versa, sem um atraso de tempo de ativação (JOHNSON, 1988). A redundância do tipo *hot standby* pode manter o sistema ativo em caso de falhas. Entretando, devido ao fato que os equipamentos neste modo estão sempre expostos a carga de trabalho, assim podendo falhar ao mesmo tempo em que o equipamento principal entre em falha, o que reduz a confiabilidade. A Figura 13 ilustra o funcionamento do *hot standby*, a falha e reparo de ambas as VMs são independentes entre si. A Equação 2.10 apresenta a expressão usada para obter a disponibilidade nesse exemplo. A Figura 14 apresenta uma modelagem equivalente das duas VMs em *hot standby* só que usando o formalismo RBD. Neste tipo de modelo, componentes em paralelo representam que estão redundantes entre si, sendo essa a maneira mais prática de modelar uma redundância em *hot standby*.

$$D = P\{(\#VM\_principal\_on = 1)OR(\#VM\_redundante\_on = 1)\} \tag{2.10}$$

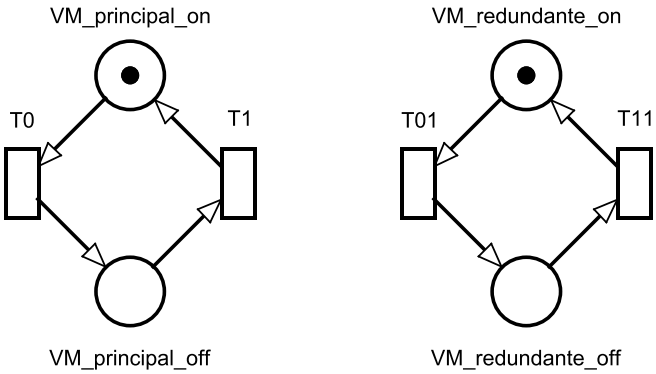

Figura 13 – Hot Standby

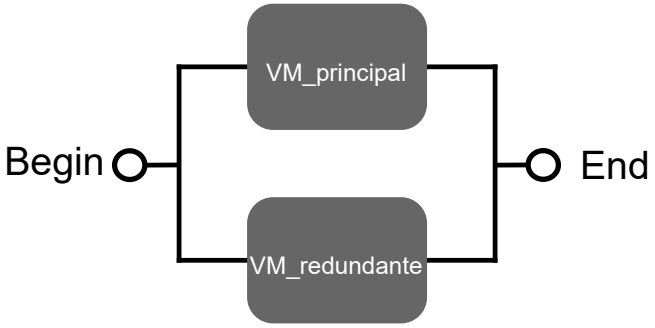

Figura 14 – Hot Standby em RBD

2.8.2 Cold Standby

Na redundância *cold standby* os componentes redundantes que estão neste modo não iniciam ativos como os componentes principais. Eles permanescem inativos esperando até que os componentes principais falhem para que sejam ativados (JOHNSON, 1988). A redundância do tipo *cold standby* neste momento ocorre a ativação dos componentes redundantes em um certo período de tempo que é chamado de Mean Time To Activate (MTTA). Por estarem normalmente inativos, os componentes redundantes acabam apresentando uma taxa de falha reduzida, o que provê uma maior confiabilidade do sistema. A Figura 15 ilustra o funcionamento do *cold standby*, e A Tabela 2 apresenta as expressões de guarda usadas nesse modelo. A expressão de guarda presente na transição MTTA garante que a VM_redundante apenas seja ativada quando não houver token no lugar VM_principal_on, ou seja a VM principal está em falha. Enquanto a expressão de guarda presente na transição imediata T3 se ativa quando houver token no lugar VM_principal_on, ou seja a VM principal está funcionando. A Equação 2.10 apresenta a expressão usada para obter a disponibilidade nesse exemplo. Diferente da redudância em *hot standby*, o RBD não permite a criação de modelos que representem a ativação um componente quando outro entra em falha.

Tabela 2 – Expressões de Guarda (Modelo Cold Standby)

Transição	Expressão
MTTA	(#VM_principal_off>0)
T3	(#VM_principal_on>0)

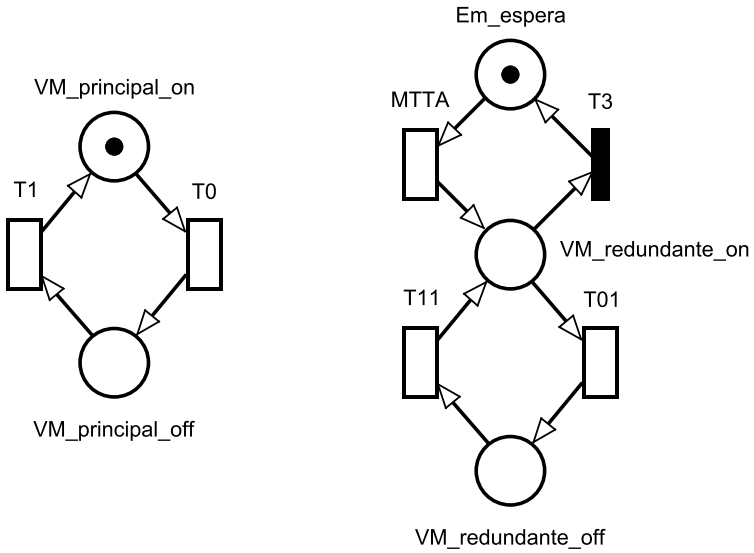

Figura 15 – Cold Standby

2.9 Modelagem Hierárquica

À medida que um sistema cresce em número de componentes, sua representação torna-se progressivamente mais complexa. Em modelos formais, como Redes de Petri Estocásticas (SPN) e Diagramas de Blocos de Confiabilidade (RBD), é comum a abstração de determinados componentes, a fim de obter uma descrição mais simplificada do sistema. No entanto, determinadas interconexões entre componentes não podem ser omitidas ou agregadas sem comprometer a fidelidade da análise, uma vez que sua ausência pode alterar significativamente os resultados obtidos na avaliação do modelo.

Em cenários nos quais a simplificação compromete a representatividade do sistema como um todo, a modelagem hierárquica surge como uma abordagem eficaz para gerenciar a complexidade por meio da modularidade. Nessa abordagem, o sistema é decomposto em subsistemas menores, os quais são modelados individualmente. Posteriormente, cada submodelo é abstraído como um componente e integrado à estrutura global, possibilitando uma representação mais compreensível e organizada do sistema completo (CUNHA et al., 2021).

Além de promover a modularização e facilitar a modelagem de sistemas complexos, a modelagem hierárquica também pode ser empregada como estratégia para superar limitações inerentes aos formalismos utilizados. Em particular, ela permite a modelagem de determinadas partes do sistema com formalismos alternativos mais adequados a aspectos específicos, integrando posteriormente os resultados desses submodelos ao modelo principal.

Neste trabalho, a modelagem hierárquica é empregada com o propósito de integrar a modelagem realizada por meio de Diagramas de Blocos de Confiabilidade (RBD)—cuja limitação reside na incapacidade de representar interdependências entre componentes—à modelagem de mecanismos de redundância do tipo cold standby. A Figura 16 ilustra um exemplo da aplicação da modelagem hierárquica adotada neste trabalho.

A parte (a) da Figura 16 apresenta a modelagem de um sistema composto por três componentes utilizando RBD, no qual o componente Comp2 possui uma estrutura redundante. Contudo, o formalismo RBD não possui suporte para representar adequadamente o comportamento de componentes em *cold standby*. Para lidar com essa limitação, o subsistema responsável por essa funcionalidade é modelado em um formalismo mais expressivo.

A parte (b) da Figura 16 apresenta esse subsistema modelado por meio de uma SPN, contendo dois componentes em configuração *cold standby*. O modelo SPN é avaliado de forma independente, e os resultados obtidos são utilizados para substituir ou parametrizar a representação do componente Comp2 no modelo RBD original, completando assim a integração hierárquica entre os dois formalismos.

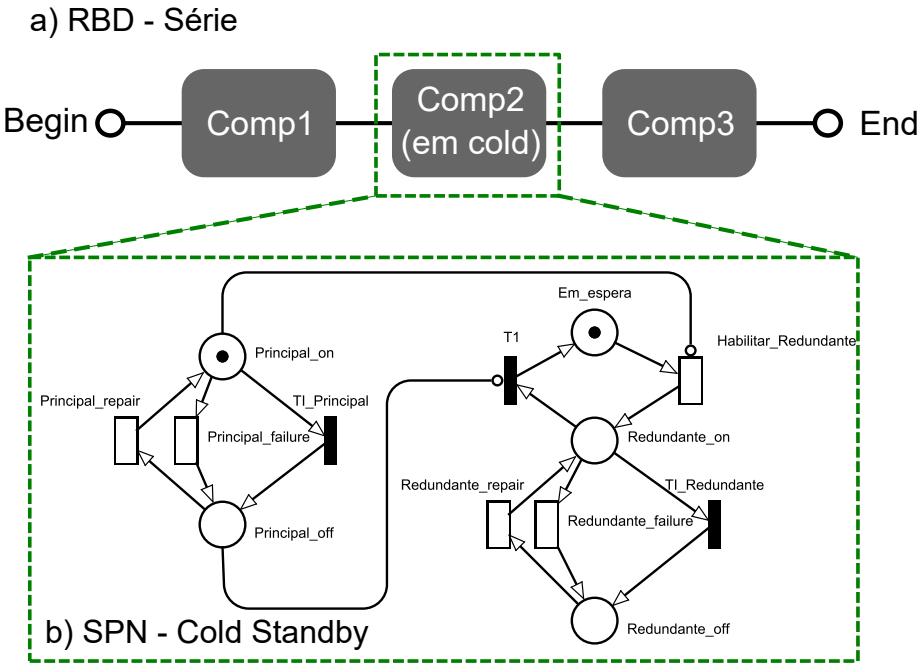

Figura 16 – Exemplo de Modelagem Hierárquica

3 Trabalhos Relacionados

Este capítulo apresenta trabalhos relacionados identificados na literatura sobre a avaliação de desempenho e a disponibilidade de ambientes de computação em nuvem, fornecendo uma análise crítica das principais abordagens, métricas e metodologias adotadas nesse campo.

Os trabalhos a seguir foram identificados por meio de mecanismos de busca especializados em artigos científicos, como o Google Scholar. Para a pesquisa, foram empregadas algumas palavras-chave em inglês como: "cloud computing", "availability and performance evaluation", que correspondem, respectivamente, aos termos: computação em nuvem, avaliação de disponibilidade e desempenho. Além das palavras-chave, foi estabelecido um critério temporal, restringindo-se aos trabalhos publicados nos cinco anos anteriores à conclusão deste estudo. Dentre os artigos encontrados, foram selecionados aqueles com maior número de citações e que apresentavam maior afinidade com o tema deste trabalho.

Os autores em (ARAÚJO et al., 2021) utilizam modelos formais baseados em SPN e RBD para avaliar a disponibilidade e confiabilidade de uma Computação em Nuvem Veicular (VCC) com múltiplas Unidades de Estrada (RSUs). Duas análises de sensibilidade foram realizadas com o objetivo de identificar quais componentes do modelo têm maior impacto. Por fim, os autores propõem uma extensão do modelo original, a qual introduz um grande número de redundâncias, resultando em melhores resultados em comparação ao modelo base.

Os autores em (PEREIRA et al., 2021) propõem modelos analíticos de disponibilidade para a avaliação de ambientes de borda e névoa. Ao analisar um ambiente real, os autores obtiveram dados que permitiram definir um intervalo de confiança, o qual serviu como base para a validação dos modelos propostos. Através de experimentos, os autores demonstraram que, ao utilizar o modelo analítico proposto para planejar a infraestrutura, é possível aumentar a disponibilidade do sistema.

Em (SHOYARI et al., 2021), os autores propõem uma modelagem hierárquica que combina RBD com CTMC para avaliar uma nuvem privada baseada no OpenStack em diferentes cenários. Para a definição dos parâmetros do modelo, foram coletados dados a partir de um ambiente real do OpenStack. Os resultados dos experimentos realizados demonstram um impacto positivo na disponibilidade e redução do downtime, embora com um aumento nos custos. Devido à natureza inversamente proporcional entre disponibilidade e custo, a decisão sobre a distribuição desses parâmetros para diferentes objetivos fica a cargo do administrador da nuvem, sendo o modelo proposto uma ferramenta para auxiliar nesse processo decisório.

Os autores em (PEREIRA et al., 2022) propõem um modelo utilizando árvore de falhas (*fault tree*) e cadeia de Markov para a avaliação da disponibilidade de um ambiente *edge-fog-cloud*. Com o modelo proposto, os autores avaliaram diversos cenários, comprovando que a adição de redundâncias contribui para a melhoria da disponibilidade do sistema.

(MACIEL et al., 2022) apresentam uma pesquisa sobre a disponibilidade e confiabilidade de arquiteturas computacionais baseadas em nuvem, neblina (*fog*) e borda (*edge*). Entre os focos deste trabalho, os autores abordam as diferenças entre *edge* e *fog*, além de realizar uma análise do estado da arte dessas arquiteturas computacionais. Como resultado da pesquisa, os autores identificaram que, até o momento em que a pesquisa foi conduzida, o ano de 2019 foi o período com o maior número de publicações nesta área, sendo a China o país com o maior volume de trabalhos publicados.

Os autores em (CLEMENTE et al., 2022) propõem a avaliação de um sistema hospedado em uma nuvem privada. Para isso, foram criados modelos hierárquicos para representar o ambiente em análise. Através da realização de uma análise de sensibilidade, os autores foram capazes de identificar quais componentes e parâmetros do sistema têm maior impacto na disponibilidade. A partir desses resultados, os autores demonstraram, por meio de experimentos, que alterações na arquitetura do sistema, com a utilização de redundâncias, proporcionam melhores resultados.

É proposto pelos autores em (SANTOS et al., 2024), um modelo baseado em SPN que incorpora a integração de um mecanismo de defesa por movimentação do alvo (MTD) baseado em eventos. Este modelo abrange tanto a detecção da probabilidade de invasão quanto a capacidade de identificar ameaças potenciais. Por meio de diferentes políticas de migração de VMs, os autores buscam identificar as estratégias mais eficientes em condições específicas. Em seus experimentos, observaram que a confiabilidade das políticas de detecção de eventos é benéfica quando o sistema apresenta uma precisão de detecção superior a 50%. Por outro lado, estratégias baseadas em tempo demonstram ser mais vantajosas quando a precisão de detecção é inferior a 50%.

(MANCAŞ, 2019) propõe uma avaliação da performance de ambientes de nuvem pública e privada, com o objetivo de recomendar o tipo de nuvem mais adequado para as especificidades de diferentes aplicações. Foram realizados testes em diversos virtualizadores para determinar o mais eficiente para uma nuvem privada, sendo o VMware aquele que demonstrou os melhores resultados. Na comparação entre os ambientes de nuvem privada e pública, os resultados indicam que a nuvem pública apresenta uma performance inferior. Com base nesses resultados, foi recomendado o uso de uma nuvem privada quando o foco for melhor desempenho e segurança. Por outro lado, se a prioridade for custo e escalabilidade, a nuvem pública é a mais indicada.

Os autores em (PINHEIRO et al., 2019) propõem uma abordagem para avaliar o

desempenho e a utilização de máquinas virtuais (VMs) em sistemas elásticos hospedados em uma nuvem pública. Para isso, é apresentado um modelo baseado em SPN para representar o sistema, juntamente com um modelo de custo para prever o uso das VMs. A abordagem adotada possibilita o planejamento de arquiteturas com base no tempo médio de resposta, além de permitir a análise de diferentes configurações do sistema. Um estudo de caso foi conduzido para avaliar a estratégia proposta, a qual se mostrou viável, sendo capaz de minimizar o tempo médio de resposta e reduzir os custos.

Os autores em (PAPADOPOULOS et al., 2019) propõem uma metodologia para a avaliação de desempenho em ambientes de computação em nuvem, com o objetivo de definir diretrizes que permitam que as estratégias de avaliação de desempenho sejam replicadas e verificadas de forma consistente. Como estudo de caso, a metodologia foi utilizada em cenários reais para demonstrar sua aplicabilidade. Os autores comprovam, através de princípios estatísticos, que a aplicação dessa metodologia resulta em uma melhoria na precisão e na confiabilidade da avaliação de desempenho desses sistemas.

Em (MATOS et al., 2020), os autores propõem uma metodologia para detectar gargalos em sistemas de computação em nuvem utilizando modelos hierárquicos. Por meio da combinação hierárquica de modelos baseados em SPN e Cadeia de Markov de tempo contínuo (CTMC), foi possível realizar uma análise de sensibilidade e classificar os parâmetros do sistema de acordo com seu impacto nas métricas de interesse. Através de um estudo de caso que avaliou serviços web, os autores concluíram que o ranqueamento dos parâmetros permite a identificação eficiente de gargalos.

Em (SANTOS; VALE; ALENCAR, 2020), é apresentada uma abordagem voltada para a avaliação de desempenho de três diferentes servidores de arquivos (OwnCloud, NextCloud e Pydio) em uma nuvem privada para uso pessoal. Através de experimentos, utilizando diferentes configurações para esses serviços e aplicando distintas cargas de estresse, foram obtidas conclusões sobre o desempenho de cada solução. Embora as três soluções apresentem um desempenho aceitável para uso pessoal, o NextCloud demonstrou o melhor desempenho para uploads de arquivos, enquanto o OwnCloud obteve melhores resultados na transmissão de vídeo.

Em (NGUYEN et al., 2021), os autores propõem um modelo para a quantificação da performance de servidores de nuvem em sistemas médicos. Diferentes técnicas de balanceamento de carga foram avaliadas, tanto com quanto sem o uso do mecanismo de failover, que é um mecanismo de redundância que garante a continuidade dos serviços. Os resultados indicaram que o uso do failover contribui para a eliminação da perda de requisições, e que o balanceamento de carga é um fator crucial para a melhoria da performance do sistema.

Os autores em (FÉ et al., 2022) apresentam um modelo baseado em SPN para a análise de mecanismos de autoescalonamento em infraestruturas de computação em nuvem.

O modelo proposto permite prever o comportamento desses mecanismos, fornecendo uma abordagem quantitativa para a avaliação de desempenho. Para validar a eficácia do modelo, os autores utilizaram um sistema de transmissão de vídeos como estudo de caso, considerando cargas de trabalho homogêneas e heterogêneas. Os resultados das simulações indicaram uma taxa de confiança de 95%, demonstrando a precisão da abordagem na representação dos cenários analisados. Além disso, o modelo proposto oferece suporte à tomada de decisão para projetistas de infraestrutura em nuvem, permitindo uma avaliação detalhada da relação entre desempenho e custo em sistemas de autoescalonamento.

Em (WASEEM et al., 2021), os autores realizam uma análise quantitativa e avaliação de desempenho de estratégias de replicação de dados baseadas em objetivos-alvo. O foco do estudo é identificar, nas estratégias de replicação orientadas a objetivos, quais são os objetivos-alvo mais frequentemente abordados e classificá-los em três categorias, da mais intensamente abordada à menos abordada. Os autores identificaram que, em cerca de 80% dos casos, as estratégias testadas tinham como objetivo primário métricas como disponibilidade, confiabilidade e desempenho.

Em (RODRIGUES et al., 2021), os autores adotam modelos analíticos para avaliar a disponibilidade e o desempenho de sistemas de hospitais inteligentes, com o objetivo de evitar gastos antecipados com equipamentos reais. Para representar a arquitetura do sistema, foram propostos dois modelos em SPN, um voltado para a avaliação da disponibilidade e outro para o desempenho. Nos experimentos realizados, os autores identificaram que a taxa de chegada de clientes impacta diretamente no desempenho do sistema, e que a presença de redundância no sistema resulta em maior disponibilidade.

Em (ANDRADE et al., 2021), é apresentada uma abordagem baseada em Redes de Petri Estocásticas e Determinísticas para a avaliação de desempenho de ambientes de Internet das Coisas (IoT) que adotam dispositivos de Fog. Através dos modelos propostos, os usuários são capazes de analisar as compensações entre diferentes configurações desses ambientes Fog-Cloud IoT. No estudo de caso realizado, os autores demonstraram que o uso de um dispositivo de Fog melhora a disponibilidade e apresenta melhores resultados do que um ambiente de nuvem, especialmente quando o nível de requisições não se aproxima da capacidade máxima de limite.

Os autores em (MELO et al., 2022) apresentam a avaliação de um ambiente privado que hospeda uma aplicação baseada em blockchain, utilizando métricas como disponibilidade, tempo de inatividade (*downtime*), latência e vazão. Para isso, foi proposto um modelo baseado em SPN para a avaliação combinada de disponibilidade e desempenho. Ao analisarem as métricas de desempenho, os autores observaram um aumento no consumo de recursos, o que poderia indicar um problema relacionado ao envelhecimento do software. Considerando esse aumento no consumo dentro do modelo combinado, os autores identificaram uma perda superior a 1% na disponibilidade.

Em (TORQUATO; MACIEL; VIEIRA, 2022), os autores propõem um conjunto de Redes de Recompensa Estocásticas (SNRs) com o objetivo de avaliar a modelagem combinada de disponibilidade e desempenho no processo de migração de VMs como forma de aliviar sistemas sobrecarregados com excesso de carga de trabalho. A partir de diferentes cenários, que consideram diversas situações de carga excessiva, os autores desenvolvem um modelo capaz de abordar as incertezas associadas a essas cargas de trabalho. Os resultados obtidos indicam que o modelo proposto oferece soluções específicas para aliviar a carga do sistema em cada cenário, com o objetivo de maximizar tanto a disponibilidade quanto o desempenho do sistema.

Os autores em (SOUSA et al., 2024) propõem a avaliação da disponibilidade e do desempenho de ambientes de big data em nuvens privadas. A abordagem adotada faz uso de modelos estocásticos e combinatórios para a análise dos ambientes estudados. Nesse contexto, um modelo em SPN foi utilizado para avaliar o desempenho, enquanto um modelo em RBD foi empregado para avaliar a disponibilidade. Nos experimentos realizados, os autores identificaram que a carga de trabalho é o fator com maior impacto nas aplicações de *big data* em nuvens privadas.

A Tabela 3 apresenta uma comparação entre este trabalho e estudos similares encontrados na literatura, levando em consideração as métricas utilizadas, a abordagem de modelagem adotada e o tipo de ambiente de nuvem avaliado.

Quanto às métricas, este trabalho apresenta um conjunto abrangente de indicadores, incluindo disponibilidade, *uptime*, *downtime*, vazão e tempo de resposta. Embora diversos dos estudos citados, como (MACIEL et al., 2022) e (SHOYARI et al., 2021), analisem a disponibilidade, e vários trabalhos semelhantes citados, como (NGUYEN et al., 2021) e (PINHEIRO et al., 2019), também avaliem o desempenho, poucos combinam ambos os aspectos, como (ANDRADE et al., 2021). Este trabalho se distingue por adotar uma abordagem que integra a modelagem de disponibilidade e desempenho, de maneira que a disponibilidade impacte diretamente no desempenho do sistema analisado.

No que se refere às técnicas de modelagem, este trabalho emprega SPN e RBD, o que possibilita tanto a análise estocástica do comportamento do sistema quanto a avaliação da confiabilidade estrutural da arquitetura. Abordagens similares podem ser encontradas em estudos como (ARAÚJO et al., 2021) e (SOUSA et al., 2024), embora aplicadas a cenários distintos, como computação veicular e *big data*, respectivamente.

No que diz respeito ao tipo de nuvem utilizada, este trabalho realiza a análise de um serviço hospedado em uma nuvem privada. Esse enfoque segue a linha de pesquisas como (MELO et al., 2022) e (SANTOS; VALE; ALENCAR, 2020), que também investigam nuvens privadas. No entanto, diferencia-se de estudos que analisam tanto nuvens públicas quanto privadas, como (MANCAŞ, 2019) e (PAPADOPOULOS et al., 2019).

Tabela 3 – Trabalhos Relacionados

Artigos	Métricas	Modelagem	Nuvem
(ARAÚJO et al., 2021)	Disponibilidade, Confiabilidade	SPN, RBD	sim
(PEREIRA et al., 2021)	Disponibilidade	CTMC	sim
(SHOYARI et al., 2021)	Disponibilidade, Downtime, Custo	RBD, CTMC	privada
(PEREIRA et al., 2022)	Disponibilidade	Arvore de Falha, CTMC	sim
(MACIEL et al., 2022)	Disponibilidade, Confiabilidade	-	sim
(CLEMENTE et al., 2022)	Disponibilidade	SPN	privada
(SANTOS et al., 2024)	Downtime, Taxa de Migração de VMs, Probabilidade de Ataque	SPN	sim
(MANCAŞ, 2019)	Vazão, Dhrystone, Whetstone	-	publica, privada
(PINHEIRO et al., 2019)	Vazão, Tempo de Resposta, Tempo de Espera	SPN	public
(PAPADOPOULOS et al., 2019)	Tempo de Resposta	-	publica, privada
(MATOS et al., 2020)	Tempo de Resposta, Tamanho da Fila, Uso de VMs	SPN, CTMC	privada
(SANTOS; VALE; ALENCAR, 2020)	Uso de CPU, Uso de disco, Uso de memória	-	privada
(NGUYEN et al., 2021)	Vazão, Tempo de Resposta	SRN	sim
(FÉ et al., 2022)	Vazão, Tempo de Resposta, Custo	SPN	privada
(WASEEM et al., 2021)	Disponibilidade, Confiabilidade, Tolerância a falhas, Escalabilidade, Balanço de carga, Elasticidade, Desempenho, Consistência, Custo	-	sim
(RODRIGUES et al., 2021)	Disponibilidade, Downtime, Tempo de Resposta	SPN	sim
(ANDRADE et al., 2021)	Disponibilidade, Vazão, Tempo de Resposta	SPN	sim
(MELO et al., 2022)	Disponibilidade, Downtime, Vazão, Latência	SPN	privada
(TORQUATO; MACIEL; VIEIRA, 2022)	Disponibilidade, Confiabilidade, Vazão	SRN	sim
(SOUSA et al., 2024)	Disponibilidade, Tempo de Execução, Uso de memória, Uso de CPU	SPN, RBD	privada
Este trabalho	Disponibilidade, Uptime, Downtime, Vazão, Tempo de Resposta	SPN, RBD	privada

Dessa forma, este estudo se distingue pela combinação de um conjunto diversificado de métricas, pela utilização complementar de SPN e RBD, e pelo foco na avaliação de nuvens privadas, oferecendo uma abordagem mais abrangente para a análise de desempenho e disponibilidade em ambientes de computação em nuvem.

4 Metodologia

Este capítulo descreve a metodologia adotada na execução desta pesquisa, cuja finalidade é apresentar a análise da disponibilidade e desempenho de sistemas hospedados em nuvens privadas. Para atingir os objetivos propostos neste trabalho, apresentados no Capítulo 1, foram utilizados técnicas, ferramentas e processos para a concepção e experimentação dos modelos propostos a serem apresentados no Capítulo 5.

A metodologia adotada tem como foco realizar a integração entre a medição de um sistema real, a criação de modelos baseados nesse sistema e a experimentação desses modelos sob a perspectiva de métricas de disponibilidade e desempenho. A qual resultou na criação de modelos para a avaliação de desempenho e disponibilidade do serviço um serviço hospedado em nuvem privada. Neste trabalho foi utilizado o serviço Nextcloud hospedado na nuvem privada Cloudstack.

A seguir, serão apresentados os passos seguidos pela metodologia. Criados com o intuito de propor modelos para reprentar o sistema em análise. Por fim, é detalhado o ambiente experimental, montado em laboratório, destinado à realização dos experimentos que fornecerão os dados de entrada para os estudos de caso apresentados no Capítulo 6.

4.1 Metodologia Proposta

A metodologia adotada segue uma sequência de etapas, podendo ser adotada por usuários que possuam um mínimo de conhecimento sobre a modelagem de sistemas a partir de formalismos como SPN e RBD. A Figura 17 apresenta um fluxograma detalhado que ilustra graficamente a sequência de etapas que compõem a metodogia proposta.

As etapas são as seguintes: compreender o sistema, estabelecer métricas, coletar dados, gerar modelos e realizar experimentos. A seguir, são apresentados a descrição e os objetivos de cada uma dessas etapas.

4.1.1 Compreender o Sistema

Esta primeira etapa consiste na compreensão do sistema a ser analisado, conforme detalhado na Seção 4.2. O objetivo desta etapa é entender a composição do sistema, abrangendo não apenas a infraestrutura física (como hosts, switches, etc.), mas também a parte lógica (como o gerenciador da nuvem, máquinas virtuais, etc.).

Um ambiente real foi configurado para dar suporte a compreensão do sistema estudado, bem como auxiliar nas etapas seguintes. Esse ambiente é descrito na Seção

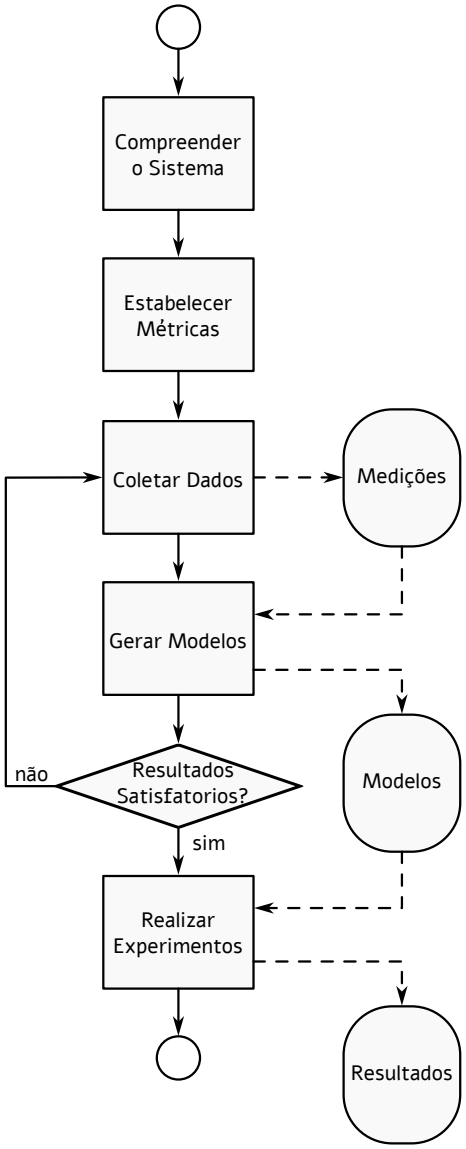

Figura 17 – Metodologia Proposta

4.2 e foi projetado para servir como um servidor em um ambiente controlado, no qual seria instalado um gerenciador de nuvem privada, e então um serviço hospedado nessa nuvem. Nesse trabalho, o Apache Cloudstack foi instalado como gerenciador da nuvem, e internamente a nuvem foi hospedado o serviço do Nextcloud.

4.1.2 Estabelecer Métricas

Após compreender o sistema e o serviço que foi hospedado nele, a segunda etapa envolve a definição dos objetivos deste trabalho. A partir disso, é realizado de um estudo para identificar as métricas que podem ser extraídas do sistema. O propósito desta fase é estabelecer quais métricas são relevantes para análise.

Nesta etapa também são definidos os métodos de medição e coleta de dados, para cada métrica escolhida, que serão usados na etapa seguinte. Para este trabalho, foram

escolhidas algumas métricas para serem utilizadas, entre elas estão:

- A disponibilidade que permite avaliar a continuidade do serviço, ou seja, a capacidade do sistema de permanecer operacional e acessível, garantindo seu funcionamento ininterrupto. Por meio dessa métrica, também é possível determinar os valores de downtime (tempo de inatividade) e uptime (tempo em atividade), o que facilita a identificação de perdas financeiras e danos à reputação causados pela indisponibilidade do serviço.
- A vazão que de sua medição indica a quantidade de dados ou requisições processadas por unidade de tempo, auxiliando na identificação de gargalos, no planejamento da escalabilidade e na otimização do desempenho do sistema.
- O tempo de resposta, cujo monitoramento possibilita avaliar a velocidade com que o serviço atende às requisições, permitindo a detecção de problemas de desempenho e a necessidade de otimizações, fatores que impactam diretamente a experiência do usuário.

4.1.3 Coletar Dados

A terceria etapa tem como o objetivo principal realizar a coleta de dados, como os tempos de execução de atividade e os limites de usuários que o sistema suporta, que serão utilizados posteriormente nos modelos propostos. Para tal é necessário realizar um estimulo ao sistema para que se possa aferir o seu comportamento.

Como não é possível encontrar usuários reais para utilizar o sistema, faz-se necessário o uso de simulação. A partir disso um plano de teste pode ser gerado, incluindo uma pequena base de dados para representar os clientes que acessariam o sistema. Para realizar a simulação do acesso de usuários é necessária uma ferramenta que gere uma carga de trabalho, neste trabalho foi empregada a ferramenta JMeter (The Apache Software Foundation, 2024). A partir da carga gera são feitas medições relativas às métricas definidas na etapa anterior. Além disso, nesta etapa, define-se o conjunto de atividades a ser executado nos experimentos, servindo como base para os modelos criados na próxima etapa. Esse conjunto de atividades consiste num procedimento simples que um usuário faria no sistema. Onde o usuário acessaria o serviço, realizaria o login no sistema, iria até a página de arquivos, faria um upload de um arquivo, e por fim, realizaria o logout.

4.1.4 Gerar Modelos

Na quarta etapa, os conhecimentos adquiridos nas etapas anteriores são empregados para gerar modelos formais em RBD e SPN, os quais representarão o sistema. Esses modelos serão detalhados no Capítulo 5. Caso os modelos não forneçam resultados dentro

do esperado, significa que as Etapas 3 (coletar dados) e 4 (gerar modelos) precisam ser executadas novamente. Assim devem ser refeitas as medições e revistos os modelos.

Esta etapa também tem como foco identificar qual formalismo usar e em quais cituações usar. O RBD é projetado para avaliar métricas de dependabilidade, como disponibilidade e confiabilidade, não servindo para modelar o desempenho de um sistema. Nesse caso entra o SPN que permite a modelagem de uma gama maior de sistemas, incluindo disponibilidade e desempenho. Outro uso do SPN é a modelagem de componentes em *cold stand-by*, que não podem ser representados em RBD, e este tipo de redundância será usada em alguns dos modelos a serem criados.

4.1.5 Realizar Experimentos

Esta ultima etapa consiste em utilizar os modelos gerados na etapa anterior para realizar experimentos. O objetivo desta etapa é testar os modelos e utilizar os resultados obtidos como dados para os estudos de caso, os quais são explicados no Capítulo 6.

Outra parte importante desta etapa é atingir os objetivos definidos na Rtapa 2. O que inclui o uso de abordagens, como o projeto de experimentos e o índice de importância da disponibilidade, para identificar os componentes de maior impacto no sistema.

4.2 Ambiente de Experimentos

Esta seção descreve a arquitetura do ambiente configurado para a realização dos experimentos. A Figura 18 ilustra essa arquitetura, composta por um servidor que integra o poder computacional de duas máquinas físicas Dell (Host1 e Host2), ambas com as seguintes especificações: processador Intel Core i5-10400F de 10ª geração, 8 GB de memória RAM, 500 GB de armazenamento e sistema operacional CentOS 7. O Host1 foi configurado como host principal, hospedando o controlador do CloudStack, responsável pelo gerenciamento de recursos do sistema, e suas VMs de sistema (SSVM e CPVM).

Uma máquina física HP, com as seguintes configurações: processador AMD A8-550B de 3,2 GHz com 4 núcleos, 8 GB de memória RAM, 500 GB de armazenamento e sistema operacional Windows 10, foi utilizada para representar o cliente que acessa os serviços hospedados no servidor. Na máquina cliente, foi instalado o Apache JMeter, uma ferramenta para geração de carga, responsável por enviar requisições ao servidor. Um switch interconecta os hosts do servidor, o cliente e a Internet.

Foram criadas três VMs no Host2 para simular um serviço em nuvem, todas com a mesma configuração (processador de 1 core de 0,5 GHz, 512 MB de memória RAM e 20 GB de armazenamento). O Nextcloud, um servidor de arquivos, foi instalado em duas dessas VMs, enquanto a terceira VM recebeu o Nginx, configurado como balanceador

de carga. Esses componentes estão representados na Figura 18, em que VM1 e VM2 correspondem às máquinas virtuais redundantes que contêm o serviço em nuvem, e LB refere-se à máquina virtual que executa o balanceamento de carga. O acesso ao serviço é gerido pelo Nginx, configurado para distribuir o tráfego de maneira proporcional entre as duas VMs que hospedam o serviço.

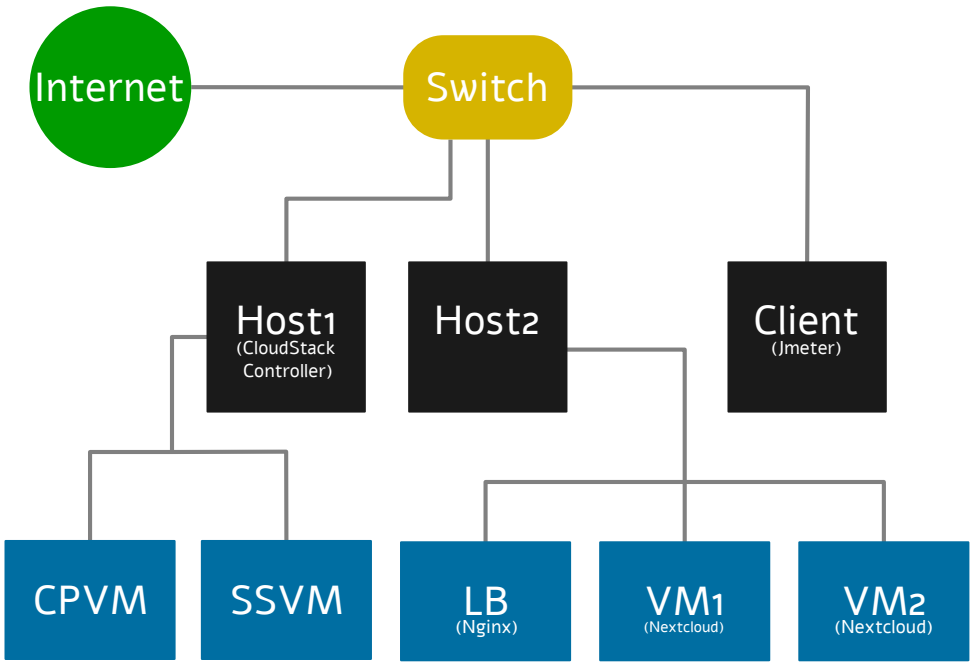

Figura 18 – Arquitetura do Ambiente de Testes

O objetivo principal deste ambiente experimental é avaliar o desempenho de um serviço em nuvem com redundância, configurado no modo *hot stand-by*. Para isso, foi desenvolvido um plano de teste que incluiu a criação de uma base de dados contendo um número determinado de usuários, cuja função é gerar a carga de trabalho necessária para os testes realizados no ambiente experimental. Com o intuito de simular o processo de acesso de um usuário no cenário real, foram delineadas cinco etapas de atividade, as quais são descritas a seguir:

- Acessar o serviço – O cliente acessa o endereço web do serviço Nextcloud.
- Fazer login – O cliente utiliza seu nome de usuário e senha para acessar as funcionalidades do serviço.
- Acessar os arquivos – O cliente navega até o menu de arquivos e acessa sua página de arquivos.
- Fazer upload – O cliente, na página de arquivos, seleciona a opção para enviar um arquivo e realiza o upload de um arquivo para o sistema.
- Fazer logout – O cliente seleciona a opção de sair do serviço para encerrar seu acesso.

5 Modelo

Este capítulo apresenta o modelo desenvolvido para representar o ambiente descrito no Capítulo 4. Além disso, neste Capítulo serão abordadas as métricas de interesse e os parâmetros associados ao modelo.

A Figura 19 ilustra o modelo SPN proposto, que representa o ambiente mostrado na Figura 18. Este modelo é composto por duas partes:

- Modelo de Disponibilidade (representado no retângulo superior da Figura 19 (a): Esta parte do modelo aborda os componentes do sistema, representando suas falhas e redundâncias, de forma a avaliar a disponibilidade dos serviços e a resiliência do sistema diante de falhas. É uma parte essencial para compreender como a infraestrutura lida com falhas e eventuais períodos de inatividade.
- Modelo de Desempenho (representado no retângulo inferior da Figura 19 (b): Este modelo descreve as atividades que o serviço realiza no ambiente e como essas atividades impactam o desempenho do sistema. Por exemplo, são computados o tempo de resposta e a vazão (*throughput*), que são métricas essenciais para garantir a performance do serviço sob diferentes condições de carga.

As próximas seções exploraram com mais detalhes cada parte do modelo proposto, abordando a dinâmica dos componentes no Modelo de Disponibilidade e as métricas relevantes no Modelo de Desempenho. Além disso, também são apresentadas possíveis melhorias para o modelo, considerando a necessidade de ajustes conforme o ambiente evolui ou novas condições sejam introduzidas.

5.1 Modelo de disponibilidade

O modelo de disponibilidade, ilustrado na parte (a) da Figura 19, foi desenvolvido em SPN e tem como objetivo calcular a disponibilidade da arquitetura que compõe o servidor integrado ao ambiente experimental apresentado no Capítulo 4.

Esse modelo é composto por vários componentes interrelacionados, a fim de representar o ambiente real anteriormente descrito. A representação de cada componente é feita de modo a demonstrar o comportamento do ambiente de nuvem em termos de falhas e reparos. A seguir, é detalhada a estrutura do modelo de disponibilidade e como ele é representado dentro do modelo geral em SPN.

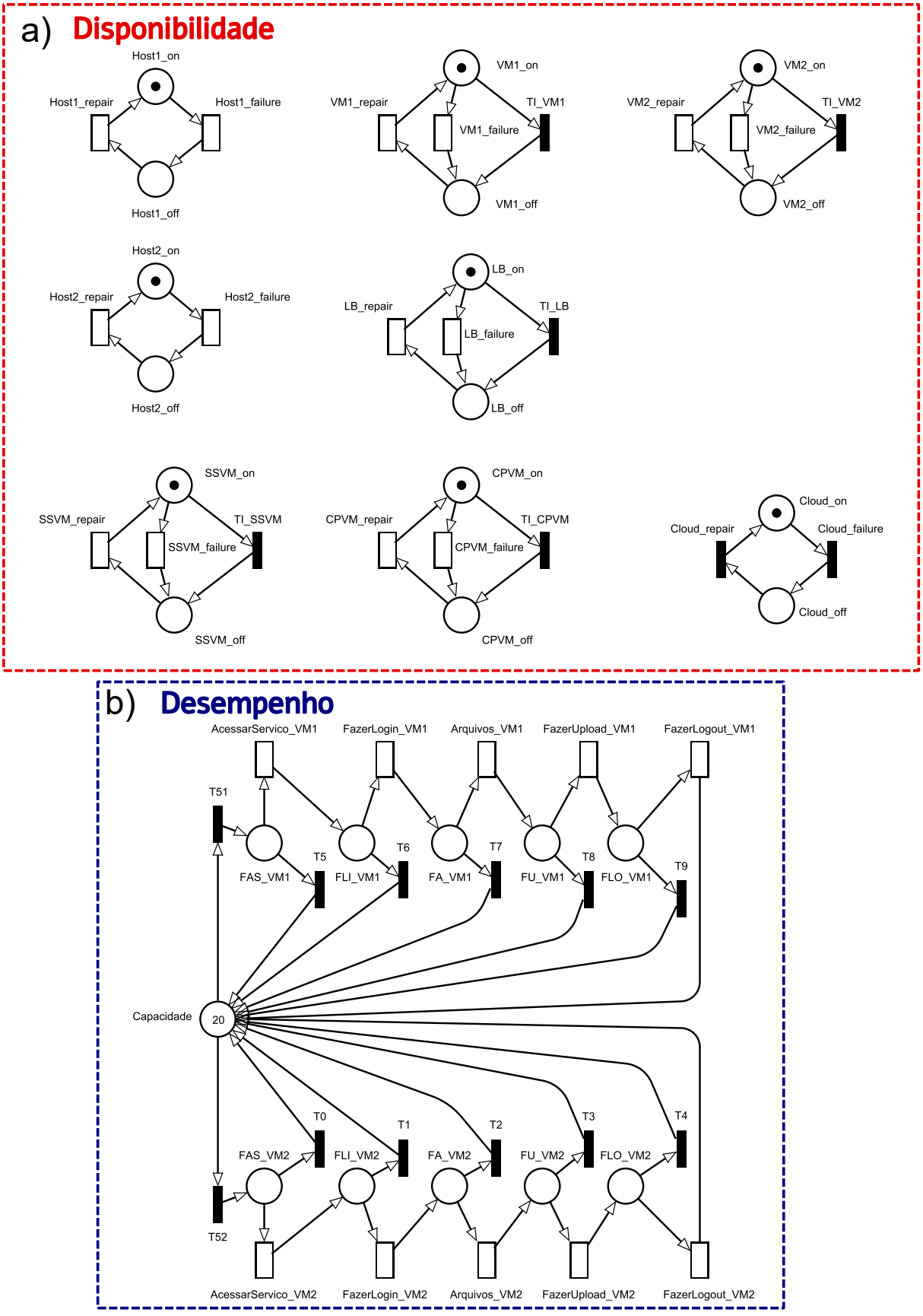

Figura 19 – Modelo Proposto

O modelo SPN é composto por dois hosts (Host1 e Host2), duas VMs de sistema (SSVM e CPVM), duas VMs de serviço (VM1 e VM2) e a VM do balanceador de carga (LB). Cada componente é representado como um sub modelo de disponibilidade, no qual um lugar denominado *_on indica que o componente está operacional, enquanto um lugar *_off sinaliza que o componente falhou. Onde o * está sendo utilizado para representar os componentes (Host1, Host2, SSVM, CPVM, VM1, VM2 e LB). Esses lugares estão conectados a transições que representam os processos de falha e reparo, respectivamente, com o tempo decorrido para a ocorrência de cada processo.

A Tabela 4 apresenta os valores adotados nas transições temporizadas *_failure e *_repair, que represetam o tempo médio de falha (MTTF) e o tempo médio de reparo (MTTR). Esses valores foram obtidos a partir de (MACIEL, 2016) e são representados em horas.

Tabela 4 – Valores Transições (Modelo Disponibilidade)

Componente	MTTF(h)	MTTR(h)
Host1, Host2	1259,0	0,768
SSVM, CPVM	619,56	0,05
VM1, VM2, LB	619,56	0,84

Como as VMs de sistema, as VMs de serviço e do balanceador de carga estão necessariamente hospedados em um dos hosts, a disponibilidade dessas VMs é impactada pelos hosts. Assim, os submodelos de disponibilidade das VMs incluem uma transição imediata, representando esse impacto. A cada uma dessas transições imediatas é associada uma expressão de guarda. A Tabela 5 apresenta essas expressões. Por exemplo, associado a Transição TI_VM1 existe a expressão de guarda Host2_off>0, a qual representa a habilitação dessa transição caso o Host2 venha a falhar. O disparo da Transição TI_VM1 faz com que a VM1 saia do estado funcionando (um token no lugar VM1_on) para o estado em falha (um token no lugar VM1_off).

Adicionalmente, a Tabela 5 inclui diferentes expressões de guarda que representam distintas formas de distribuição das VMs entre os hosts. Mais detalhes sobre essas distribuições serão fornecidos no Capítulo 6. Cada uma dessas distribuições se refere a uma maneira diferente de alocar as VMs com o serviço e o balanceador de carga entre os dois hosts. Onde na Distribuição 1 essas VMs se encontram todas no Host1, na Distribuição 2 uma das VMs com o serviço é alocada no Host2 e na Distribuição 3 todas as VMs com o serviço estão alocadas no Host2. Deve-se observar que, para as Distribuições 2 e 3, existem múltiplas entradas na tabela, uma vez que as VMs estão distribuídas entre os dois hosts, enquanto na Distribuição 1 as VMs estão alocadas exclusivamente no Host1.

O sistema é considerado operacional quando o lugar Cloud_on contém um token, e é considerado em falha quando o lugar Cloud_off possui um token. As transições imediatas

Tabela 5 – Expressões de Guarda (Modelo Disponibilidade)

Distribuição.	Transição	Expressão
1	TI_SSVM,TI_CPVM,TI_VM1,TI_VM2,TI_VM3,TI_LB,TI_LB2	(#Host1_off>0)
2	TI_SSVM,TI_CPVM,TI_VM2,TI_VM3,TI_LB,TI_LB2	(#Host1_off>0)
2	TI_VM1	(#Host2_off>0)
3	TI_SSVM,TI_CPVM,TI_LB,TI_LB2	(#Host1_off>0)
3	TI_VM1,TI_VM2,TI_VM3	(#Host2_off>0)

Tabela 6 – Expressões de Guarda (Componente Cloud)

Transição	Expressões
Cloud_failure	((#SSVM_off>0) OR (#CPVM_off>0)) OR ((#LB_off>0) OR ((#VM1_off>0) AND (#VM2_off>0)))
Cloud_repair	((#SSVM_on>0) AND (#CPVM_on>0)) AND ((#LB_on>0) AND ((#VM1_on>0) OR (#VM2_on>0)))

associadas a esses lugares possuem expressões de guarda, as quais estão detalhadas na Tabela 6. Dessa forma, a transição Cloud_failure pode ser acionada se qualquer uma das VMs do sistema, o balanceador de carga, ou ambas as VMs de serviço falharem. Por outro lado, a transição Cloud_repair pode ser acionada se as VMs do sistema, o balanceador de carga e, pelo menos, uma das VMs de serviço estiverem operacionais.

Na Seção 5.4 são apresentadas algumas possíveis extensões para o modelo em SPN. Em cada uma dessas extensões é considerado um cenário onde foi levado em consideração uma modificação na arquitetura apresentada na Figura 18. No Cenário com 3VMs é considerada uma VM adicional com o serviço do Nextcloud. Por isso as expressões de guarda do Componente Cloud possuem um acréscimo dessa VM extra, assim o sistema entrará em falha se uma das 3 VMs falhar e retornará a funcionar se houver pelo menos uma dessas 3 VMs funcionando. Enquanto que no Cenário com 2LBs é considerdo um balanceador de carga adicional. Assim as expressões de guarda Componente Cloud

5.2 Modelo de desempenho

O modelo de desempenho em SPN apresentado na parte (b) da Figura 19 representa o funcionamento do sistema levando em conta os procedimentos realizado no serviço hospedado no ambiente de nuvem apresentado no Capítulo 4.

Esse modelo é baseado no modelo proposto em (SILVA et al., 2023), com modificações para considerar a presença de duas VMs e um balanceador de carga. O modelo é estruturado em torno do lugar Capacidade, que representa o número máximo de usuários simultâneos que o sistema pode atender. A partir deste lugar, são definidas duas transições imediatas (T51 e T52), com peso de 0,5 cada, as quais simbolizam a distribuição de carga realizada pelo balanceador de carga entre as VMs de serviço disponíveis.

O sistema é iniciado com o disparo das transições imediatas T51 e T52, as quais se

ativam caso existam clientes a serem atendidos (token presente no lugar Capacidade). O disparo da transição T51 inicia o acesso ao serviço prestado pela VM1, enquanto o disparo da transição T52 inicia o acesso ao serviço fornecido pela VM2. O acionamento dessas transições consome um token do lugar Capacidade e gera um token no lugar FAS_VMn, indicando que um cliente chegou ao sistema e foi direcionado para a VMn, sendo n o número da VM acessada.

Os lugares FAS_VMn, FLI_VMn, FA_VMn, FU_VMn e FLO_VMn representam as filas de solicitações. ONde as transições temporizadas que partem desses lugares representam as ações de acessar a página do serviço, realizar o login no sistema, acessar os arquivos, enviar arquivos e realizar o logout respectivamente. Essas ações ocorrem com um tempo que é associado a transição e descreve o tempo necessário para realizar a ação. O disparo dessas transições consome todos os tokens do lugar anterior e gera um token no lugar subsequente, conforme a semântica de serviço adotada.

A Tabela 7 apresenta os valores dos tempos associados às transições temporizadas, os quais foram obtidos por meio de medições realizadas em um sistema real de testes. Para isso, utilizou-se uma ferramenta capaz de simular o acesso a um serviço *Web*, como o JMeter, que gerou uma carga de trabalho representando múltiplos usuários interagindo com o serviço. A partir dessa mesma ferramenta, foram coletados dados relativos à duração de cada atividade e ao número total de usuários simultâneos que o sistema consegue suportar sem ocorrência de perdas de pacotes.

Nos experimentos, o acesso dos clientes e a utilização do sistema por estes ocorre de maneira simultânea, ou seja, todos os clientes acessam o sistema ao mesmo tempo, sendo atendidos individualmente e simultaneamente. Nesse contexto, o modelo deve considerar que o número máximo de clientes que o sistema pode atender simultaneamente estará, de fato, acessando o serviço, o que exige uma semântica para as transições temporizadas que reflita esse conceito. A semântica denominada infinity server atende ao objetivo proposto, ao assumir que o sistema dispõe de elevado poder computacional, sendo capaz de executar uma quantidade de tarefas paralelas proporcional a esse poder.

Um aspecto de grande importância no modelo proposto (Figura 19), mais especificamente no modelo de desempenho, é o impacto da disponibilidade do sistema sobre seu desempenho. Esse impacto é representado pelas transições imediatas que partem dos lugares FAS_VMn, FLI_VMn, FA_VMn, FU_VMn e FLO_VMn, as quais simulam a interrupção das atividades do serviço quando ocorre uma falha na respectiva VM associada ao serviço.

As transições T0 a T4 são utilizadas para interromper as atividades do sistema assim que o serviço fornecido pela VM2 deixar de funcionar. De maneira análoga, as transições T5 a T9 são responsáveis por coletar todas as atividades que estavam sendo executadas quando a VM1 falhou. Para representar a relação entre o modelo de desempenho

Tabela 7 – Valores Transições (Modelo Desempenho)

Atividade	Tempo(h)
AcessarServico	0,00073
FazerLogin	0,00258
Arquivos	0,00064
Upload	0,00091
FazerLogoff	0,00078

Tabela 8 – Expressões de Guarda (Modelo Desempenho)

Transição	Expressão
T51,T5 a T9	((#Cloud_off>0)OR(#VM1_off>0))
T52,T0 a T4	((#Cloud_off>0)OR(#VM2_off>0))

e o modelo de disponibilidade, foram incluídas expressões de guarda associadas a essas transições imediatas, conforme detalhado na Tabela 8.

5.3 Métricas

As métricas de interesse que podem ser calculadas por meio dos modelos SPN apresentados anteriormente são disponibilidade, vazão e tempo de resposta. A métrica disponibilidade é calculada através do modelo de disponibilidade mostrado na Figura 19(a). As métricas de vazão e tempo de resposta são extraídas do modelo de desempenho mostrado na Figura 19(b).

A fórmula utilizada para calcular a disponibilidade pode ser vista na Equação 5.1 é executada sobre o Componente Cloud. Assim a disponibilidade (D) é calculada através da probabilidade de haver um token no Lugar Cloud_on. Para se calcular a vazão é utilizada a Equação 5.2. A vazão (V) é calculada pela soma das esperanças de haver um token nos Lugares FLO (fila de clientes esperando para deslogar) dividido pelo tempo que leva para executar essa ação. Para calcular o tempo de resposta (TR) é utilizado o cálculo apresentado na Equação 5.3. Neste cálculo é feito o somatório das esperanças de haver algum token em alguma etapa do acesso do cliente (Lugares FAS, FLI, FA, FU, FLO) de ambas as VMs, divido pela vazão. Nessas equações, (E{#F*}) representa o número esperado de tokens no respectivo lugar.

$$D = P\{\#Cloud\_on = 1\} \tag{5.1}$$

$$V = ((E\{\#FLO\_VM1\}) * (\frac{1}{2.805})) + ((E\{\#FLO\_VM2\}) * (\frac{1}{2.805})) \tag{5.2}$$

$$TR = \frac{\begin{array}{c}((\text{E}\{\#\text{FAS\_VM1}\})+(\text{E}\{\#\text{FLI\_VM1}\})+(\text{E}\{\#\text{FA\_VM1}\})+(\text{E}\{\#\text{FU\_VM1}\})+(\text{E}\{\#\text{FLO\_VM1}\})+ \\ (\text{E}\{\#\text{FAS\_VM2}\})+(\text{E}\{\#\text{FLI\_VM2}\})+(\text{E}\{\#\text{FA\_VM2}\})+(\text{E}\{\#\text{FU\_VM2}\})+(\text{E}\{\#\text{FLO\_VM2}\}))\end{array}}{(((E\{\#FLO\_VM1\}) * (\frac{1}{2.805})) + ((E\{\#FLO\_VM2\}) * (\frac{1}{2.805})))} \tag{5.3}$$

5.4 Extensões do Modelo

O modelo apresentado anteriormente na Figura 19 foi estendido para acomodar outros cenários, com o objetivo de aumentar a disponibilidade do sistema e aprimorar seu desempenho.

5.4.1 Cenário 3VMs

A primeira extensão do modelo surge de um cenário que tem como objetivo aprimorar o desempenho do serviço hospedado na nuvem. Para isso, uma VM adicional com o serviço (VM3) foi introduzida. A Figura 20 aprenta como ficaria a arquitetura do sistema ao considerar esse novo componente. A VM3 se encontra em redundância *hot standby* em relação as demais VM com o serviço.

A Figura 21 ilustra o modelo SPN que representa esse cenário, onde usa de base o modelo proposto (Figura 19) com adição da VM extra com o serviço. Vale ressaltar que o acréscimo dessa VM altera tanto o modelo de disponibilidade, com a adição de um submodelo de disponibilidade para representar a VM3. Quanto o modelo de desempenho, que agora recebe uma representação do acesso a VM3, que se incicia com o disparo da transição imediata T53. Além disso, o peso das transições imediatas que partem do Lugar Capacidade tem seus pesos alterados para 0,33.

No Cenário 3VMs é considerado uma VM adicional com o serviço. No modelo de desempenho 3VMs permanescem as transições imediatas T0 a T9, e elas funcionam da mesma maneira que no modelo proposto. Entretanto, no modelo estendido apresentado na Figura 21, as transições T10 a T14 operam de forma semelhante, mas em relação à VM3. Para representar a relação entre o modelo de desempenho e o modelo de disponibilidade, foram incluídas expressões de guarda associadas a essas transições imediatas, conforme detalhado na Tabela 9.

Neste cenário, além das transições imediatas T51 e T52, existe também a transição imediata T53. Essa transição é responsavel por iniciar o acesso ao serviço prestado pela VM3. E quando dispara gera um token no Lugar FAS_VM3. O resto da sequência acontece da mesma maneira como nas demais VMs com o serviço.

A equação utilizada para calcular a disponibilidade do modelo 3VMs é mesma utilizada para o modelo proposto e pode ser vista na Equação 5.1 é executada sobre

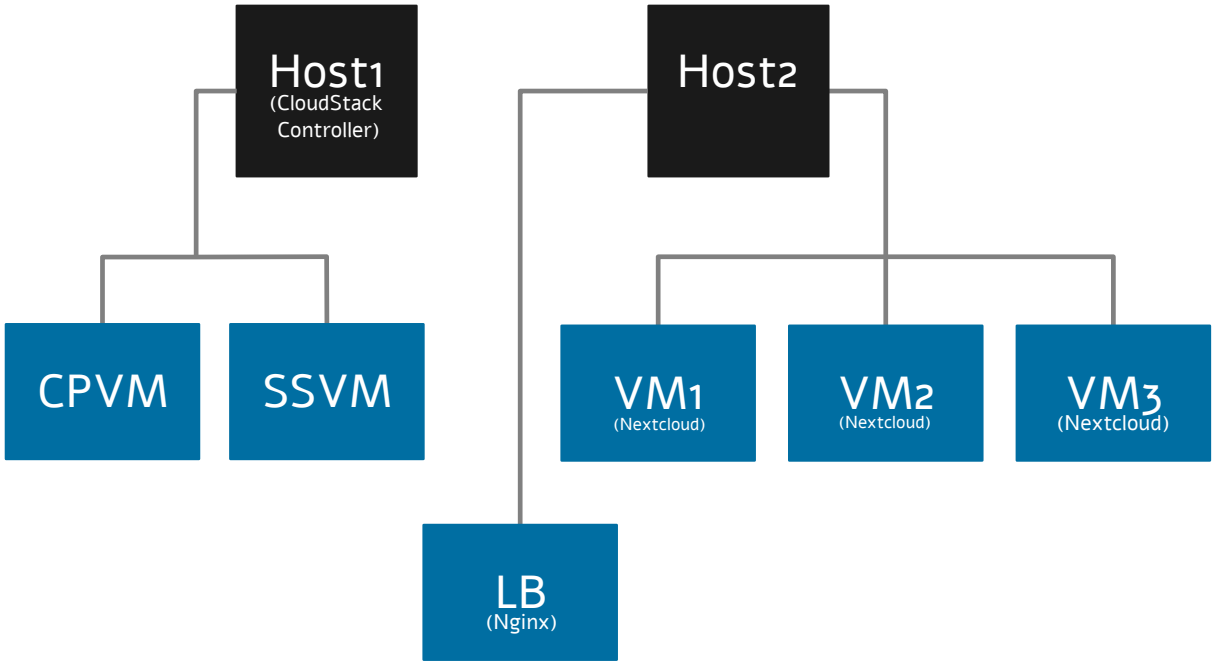

Figura 20 – Ambiente possuindo 3 VMs com serviço

Tabela 9 – Expressões de Guarda 3VMs (Modelo Desempenho)

Transição	Expressão
T51,T5 a T9	((#Cloud_off>0)OR(#VM1_off>0))
T52,T0 a T4	((#Cloud_off>0)OR(#VM2_off>0))
T53,T10 a T14	((#Cloud_off>0)OR(#VM3_off>0))

Tabela 10 – Expressões de Guarda 3VMs (Componente Cloud)

Cenário	Transição	Expressões
3VMs	Cloud_failure	((#SSVM_off>0) OR (#CPVM_off>0)) OR ((#LB_off>0) OR ((#VM1_off>0) AND (#VM2_off>0) AND (#VM3_off>0)))
3VMs	Cloud_repair	((#SSVM_on>0) AND (#CPVM_on>0)) AND ((#LB_on>0) AND ((#VM1_on>0) OR (#VM2_on>0) OR (#VM3_on>0)))

o Componente Cloud. A diferença é que neste modelo extendido, a presença da VM3 modifica as expressões de guarda do Componente Cloud, como pode ser visto na Tabela 10. Onde, considerando apenas as VMs com o serviço, o sistema falhará se as três VMs com o serviço falhar, ao invés de apenas duas. E o sistema retornará se pelo menos uma das três estiver funcionando, ao invés de pelo menos uma das duas.

Para se calcular a vazão do modelo 3VMs é utilizada a Equação 5.4. A vazão (V(3VMs)) é calculada de maneira similar ao modelo proposto, mas soma das esperanças de haver um token nos Lugares FLO (fila de clientes esperando para deslogar) das VMs dividido pelo tempo que leva para executar essa ação leva em consideração três lugares FLO. Para calcular o tempo de resposta (TR(3VMs)) é utilizado o cálculo apresentado na Equação 5.5. Assim como no cálculo feito para o modelo proposto, o somatório das esperanças de haver algum token em alguma etapa do acesso do cliente (Lugares FAS, FLI,

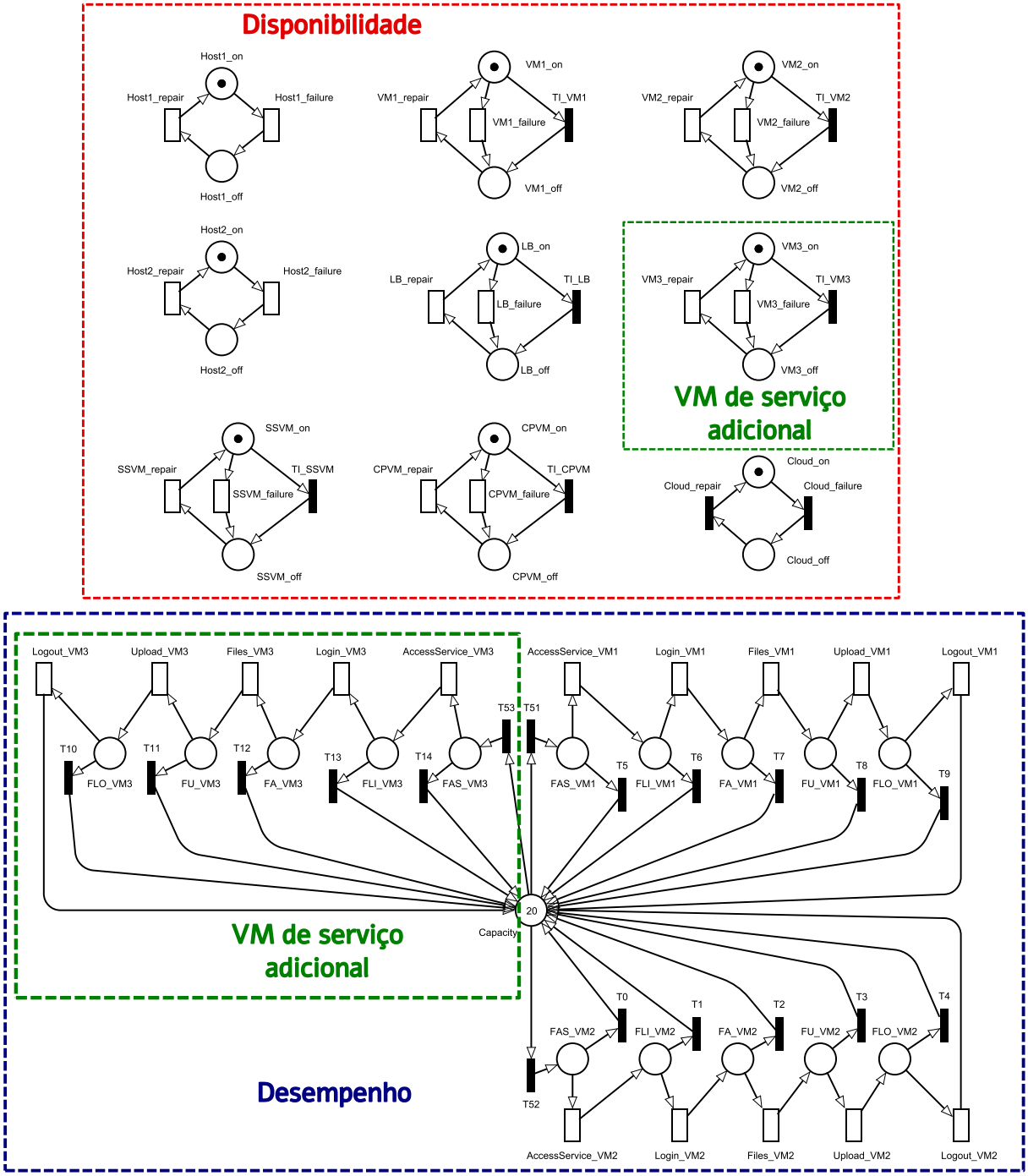

Figura 21 – Modelo possuindo 3 VMs com serviço

FA, FU, FLO) de ambas as VMs leva em consideração as três VMs, divido pela vazão do medelo 3VMs.

$$V(3VMs) = ((\mathrm{E}\{\#\mathrm{FLO\_VM1}\})*(\tfrac{1}{2.805}))+((\mathrm{E}\{\#\mathrm{FLO\_VM2}\})*(\tfrac{1}{2.805}))+((\mathrm{E}\{\#\mathrm{FLO\_VM3}\})*(\tfrac{1}{2.805})) \tag{5.4}$$

$$TR(3VMs) = \frac{\begin{array}{c}((E\{\#FAS\_VM1\}) + (E\{\#FLI\_VM1\}) + (E\{\#FA\_VM1\}) + (E\{\#FU\_VM1\}) + (E\{\#FLO\_VM1\}) + \\ (E\{\#FAS\_VM2\}) + (E\{\#FLI\_VM2\}) + (E\{\#FA\_VM2\}) + (E\{\#FU\_VM2\}) + (E\{\#FLO\_VM2\}) + \\ (E\{\#FAS\_VM3\}) + (E\{\#FLI\_VM3\}) + (E\{\#FA\_VM3\}) + (E\{\#FU\_VM3\}) + (E\{\#FLO\_VM3\}))\end{array}}{(((E\{\#FLO\_VM1\}) * (1/2.805)) + ((E\{\#FLO\_VM2\}) * (1/2.805)) + ((E\{\#FLO\_VM3\}) * (1/2.805)))} \quad (5.5)$$

5.4.2 Cenário 2LBs

A segunda extensão do modelo se origina de um cenário que busca melhorar a disponibilidade do sistema. Para tal é considerada a introdução de uma redundância para um componente crítico do sistema, identificado no experimento e1 do Estudo de Caso 6.3, o balanceador de carga. A Figura 22 apresenta como ficaria a arquitetura do sistema ao considerar esse novo componente. Visto não existe a necessidade de dois balanceadores de carga ativos simultaneamente, o LB2 se encontra em *cold standby*.

A Figura 23 apresenta o modelo SPN que representa esse cenário, sendo baseado no modelo proposto (Figura 19) com adição do balanceador de carga redundante. Neste novo modelo existe a inclusão de um submodelo de disponibilidade para representar o LB2. Como apenas um balanceador de carga estará ativo por vez, não existe diferença entre o modelo de desempenho para este cenário e o modelo proposto.

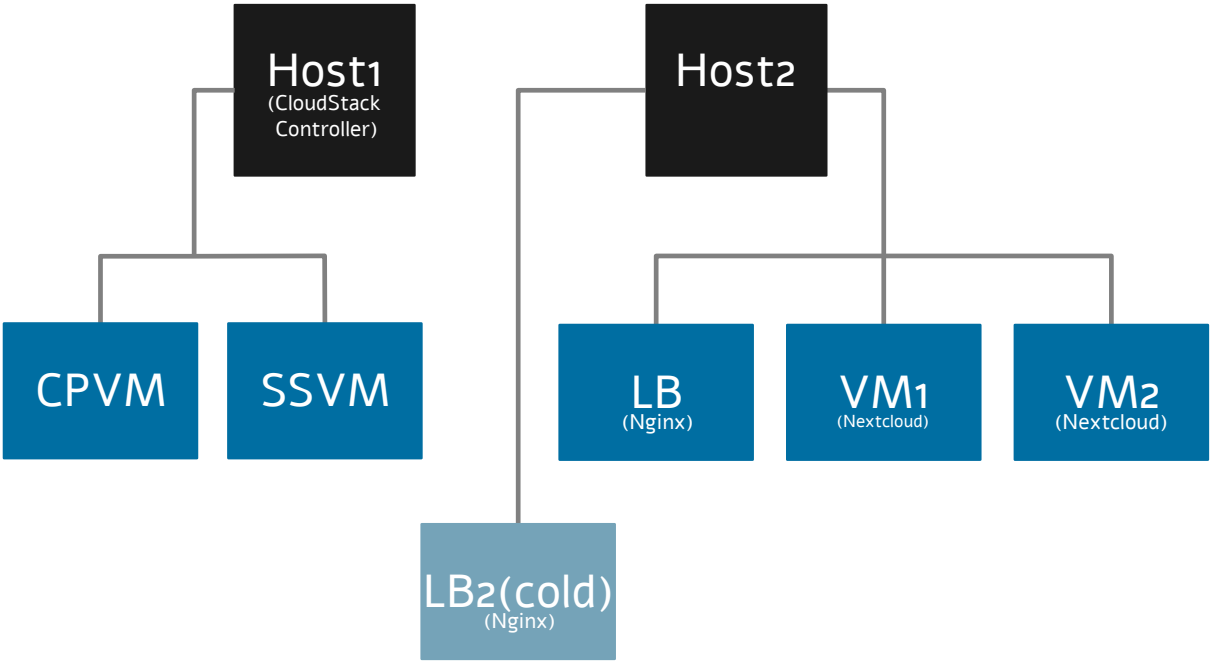

Figura 22 – Ambiente possuindo 2 VMs com balanceador de carga

A equação utilizada para calcular a disponibilidade do modelo 2LBs é mesma utilizada para o modelo proposto e pode ser vista na Equação 5.1 sendo executada sobre o Componente Cloud. A diferença é que nesta extensão do modelo, a presença do LB2 modifica as expressões de guarda do Componente Cloud, como pode ser visto na Tabela 11.

Tabela 11 – Expressões de Guarda 2LBs (Componente Cloud)

Cenário	Transição	Expressões
2LBs	Cloud_failure	((#SSVM_off>0) OR (#CPVM_off>0)) OR (((#LB_off>0) AND (#LB_off>0)) OR ((#VM1_off>0) AND (#VM2_off>0)))
2LBs	Cloud_repair	((#SSVM_on>0) AND (#CPVM_on>0)) AND (((#LB_on>0) OR (#LB_on>0)) AND ((#VM1_on>0) OR (#VM2_on>0)))

Onde, considerando apenas os balanceadores, o sistema falhará se ambos os balanceadores falhar, ao invés de apenas um. E o sistema retornará se pelo menos um dos balanceadores estiver funcionando, ao invés de pelo menos um.

Como o Cenário 2LBs possui um modelo de desempenho igual ao modelo proposto, as métricas de vazão e tempo de resposta são iguais as do modelo proposto. Sendo a Equação 5.2 usada para obter a vazão do sistema, e a Equação 5.3 para obter o tempo de resposta.

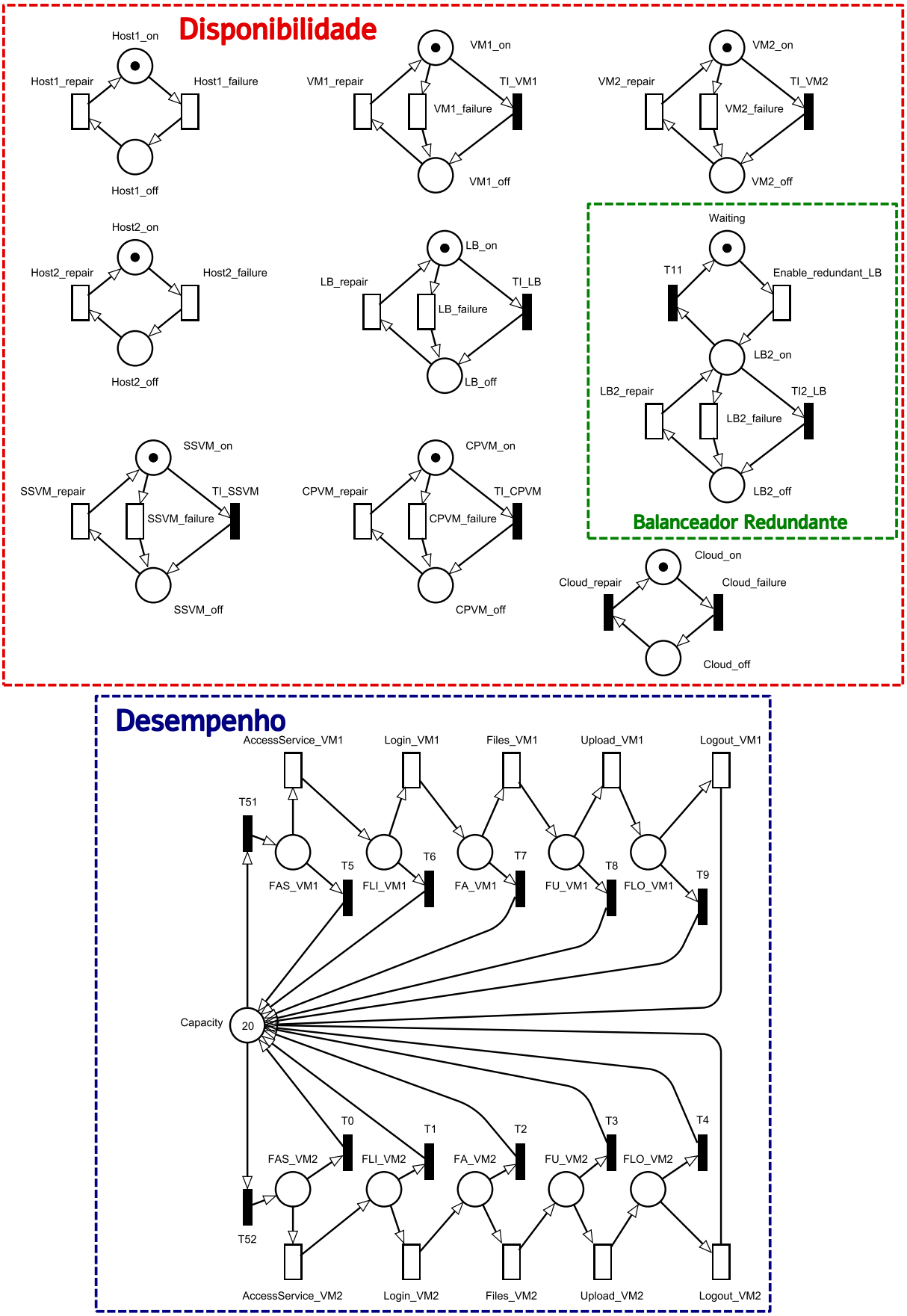

Figura 23 – Modelo possuindo 2 VMs com balanceador de carga

6 Estudo de caso

Este capítulo apresenta os estudos de caso realizados a partir da análise dos modelos gerados para representar o ambiente de experimentos descrito na Seção 4.2. O foco do estudo é avaliar o desempenho e a disponibilidade do serviço Nextcloud hospedado na nuvem privada Apache CloudStack. Através de diferentes cenários e abordagens, os estudos buscam entender como as variações no sistema, como a inclusão de redundâncias, afetam essas métricas cruciais, como disponibilidade, vazão e tempo de resposta.

O primeiro estudo de caso busca analisar as diferenças, em termos de disponibilidade e desempenho, de diferentes formas de distribuição das VMs hospedadas em um servidor com dois hosts. O segundo estudo de caso avalia o sistema sob a perspectiva de um projeto de experimentos (DoE) baseado em design fatorial. O terceiro estudo de caso foca na avaliação da disponibilidade, utilizando a técnica do índice de importância da disponibilidade para identificar os componentes de maior impacto no sistema. O quarto estudo de caso considera diferentes tempos de chegada de usuários ao sistema e analisa o impacto disso no desempenho geral do sistema.

6.1 Estudo de Caso 1

Este estudo de caso tem como objetivo avaliar a disponibilidade e o desempenho do sistema apresentado na Figura 18 sob a perspectiva de diferentes cenários. Para isso, diferentes cenários foram elaborados, sendo que cada cenário corresponde a uma versão distinta desse ambiente.

O primeiro cenário, denominado Cenário Básico, corresponde à versão do sistema real, sem alterações, cuja representação em SPN é apresentada pelo modelo proposto na Figura 19. O segundo cenário, denominado Cenário 3VMs, considera uma melhoria para o sistema, ao adicionar uma VM contendo o serviço do Nextcloud. A Figura 21 apresenta o modelo correspondente a este cenário. O terceiro cenário, denominado Cenário 2LBs, inclui uma redundância em cold standby para o balanceador de carga, com o modelo SPN correspondente mostrado na Figura 23.

Esses cenários foram avaliados por meio de diferentes distribuições das VMs no sistema. Cada distribuição possui uma configuração única para a alocação das VMs de serviço e do balanceador de carga entre os hosts.

- Distribuição 1: as VMs de serviço e o balanceador de carga são hospedados no Host1.
- Distribuição 2: uma das VMs com o serviço é hospedada no Host2, enquanto as

demais permanecem no Host1.

- Distribuição 3: apenas o balanceador de carga permanece no Host1, enquanto as VMs de serviço são alocadas no Host2.

Como cada distribuição de VMs entre os hosts gera uma variação no sistema apresentado em cada cenário, é necessário que cada distribuição seja representada por um modelo específico. As diferenças entre os modelos de distribuições diferentes dentro de um mesmo cenário correspondem às expressões de guarda inseridas nas transições imediatas dos submodelos de disponibilidade das VMs, conforme mostrado na Tabela 5.

Essas expressões de guarda nos modelos de disponibilidade começam com a condição #Host1_off>0 na Distribuição 1, quando as VMs estão alocadas no Host1. À medida que as VMs migram para o Host2 nas distribuições subsequentes, a expressão de guarda é alterada para #Host2_off>0, refletindo a mudança na alocação das VMs entre os hosts.

As métricas analisadas neste estudo são: disponibilidade, vazão e tempo de resposta. Com base nos resultados obtidos, o objetivo é identificar a melhor forma de distribuição das VMs. Um "melhor resultado"é definido como aquele que apresenta maior disponibilidade, maior vazão e menor tempo de resposta. Os resultados de disponibilidade são expressos em número de noves (9s), que são calculados por 9s = -log_{10}(1 - A), onde A representa a disponibilidade do sistema. Quanto maior o número de noves, melhor a disponibilidade do serviço.

6.1.1 Resultados

A Tabela 12 apresenta os resultados obtidos a partir da avaliação do modelo no Cenário Básico (Figura 19). Neste cenário, a melhor performance, considerando as métricas de disponibilidade, vazão e tempo de resposta, foi observada na Distribuição 2.

Tabela 12 – Resultados Cenário Básico.

Distribuição	Disponibilidade	Vazão	Tempo de Resposta
1	2,237	0,917	21,687
2	2,252	0,923	21,541
3	2,086	0,911	21,784

Obs: Disponibilidade em 9s, Vazão em req/s, Tempo de Resposta em s.

A Tabela 13 apresenta os resultados obtidos a partir da avaliação do Cenário 3VMs, que contempla uma VM adicional em configuração de redundância hot standby para o serviço de nuvem. Neste cenário, a Distribuição 1 apresentou a maior disponibilidade, o melhor vazão e o menor tempo de resposta.

A Tabela 14 apresenta os resultados obtidos a partir da avaliação do Cenário 2LBs, o qual considera um segundo balanceador de carga configurado em redundância

Tabela 13 – Resultados Cenário 3VMs.

Distribuição	Disponibilidade	Vazão	Tempo de Resposta
1	2,357	0,935	21,312
2	2,215	0,925	21,502
3	2,187	0,925	21,483

Obs: Disponibilidade em 9s, Vazão em req/s, Tempo de Resposta em s.

cold standby. Neste cenário, a maior disponibilidade foi alcançada na Distribuição 2. Em contrapartida, os resultados mais relevantes em termos de vazão foram obtidos na Distribuição 1.

Tabela 14 – Resultados Cenário 2LBs.

Distribuição	Disponibilidade	Vazão	Tempo de Resposta
1	2,284	0,942	21,122
2	2,367	0,940	21,187
3	2,301	0,935	21,290

Obs: Disponibilidade em 9s, Vazão em req/s, Tempo de Resposta em s.

As Figuras 24, 25 e 26 apresentam uma comparação entre os três cenários para as métricas de disponibilidade, vazão e tempo de resposta, considerando cada uma das três distribuições. Os valores apresentados nessas figuras correspondem à variação percentual dos resultados obtidos nos cenários analisados, em relação ao Cenário Básico.

A Figura 24 ilustra a comparação entre os três cenários utilizando a Distribuição 1, na qual as VMs com o serviço e o balanceador de carga estão hospedados no Host1. Nesse cenário, o Cenário 3VMs obteve o melhor resultado em termos de disponibilidade, com um aumento superior a 5% em comparação com o Cenário Básico. Por outro lado, o Cenário 2LBs apresentou os melhores resultados em termos de vazão e tempo de resposta, com um aumento de 1,9% na vazão e uma redução de 2,6% no tempo de resposta, quando comparado ao Cenário Básico.

A Figura 25 apresenta a comparação dos três cenários utilizando a Distribuição 2, na qual uma das VMs com o serviço é hospedada no Host2, ao contrário da distribuição anterior. Neste caso, o Cenário 2LBs obteve os melhores resultados entre os três cenários, alcançando um aumento de cerca de 5% na disponibilidade, um aumento de 1,8% na vazão e uma redução de 2,6% no tempo de resposta, em relação ao Cenário Básico.

A Figura 26 também mostra a comparação entre os cenários, mas utilizando a Distribuição 3, na qual ambas as VMs de serviço estão hospedadas no Host2. Semelhante à distribuição anterior, o Cenário 2LBs obteve os melhores resultados, apresentando um aumento superior a 10% na disponibilidade, um aumento aproximado de 2,7% na vazão e uma redução de 2,7% no tempo de resposta, em comparação com o Cenário Básico.

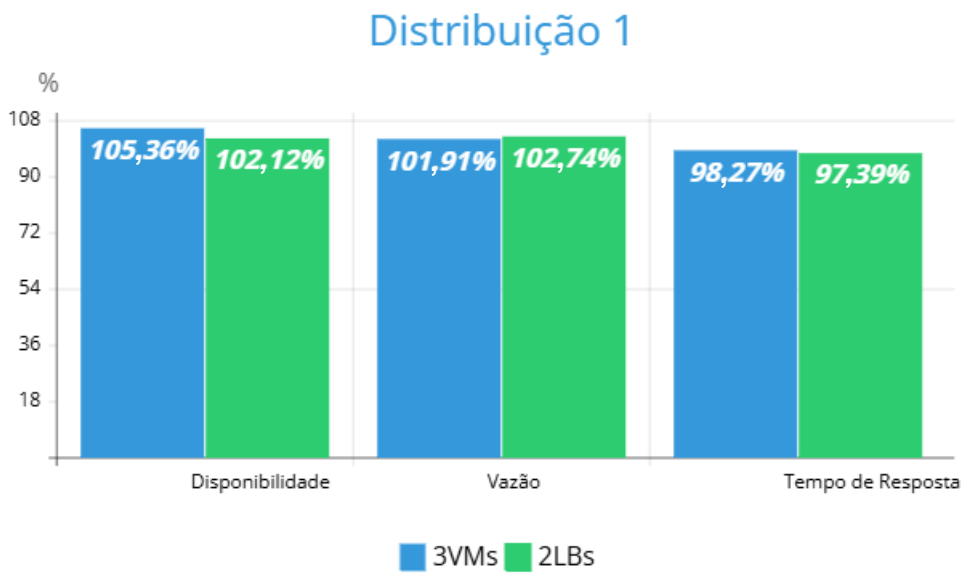

Figura 24 – Comparação Distribuição 1

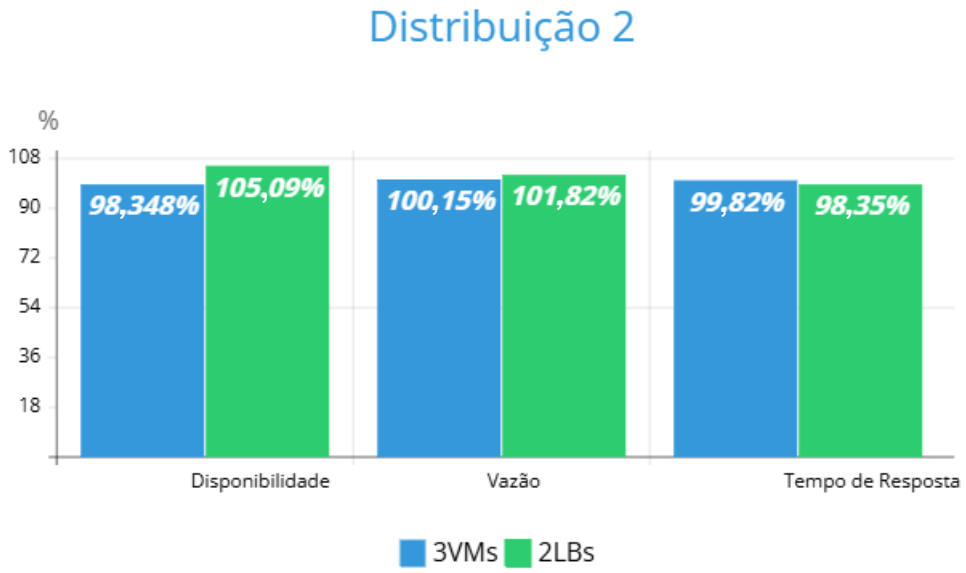

Figura 25 – Comparação Distribuição 2

A análise comparativa mostra que a adição de uma VM com o serviço de nuvem (Figura 21) melhora os resultados das métricas do sistema em relação ao Cenário básico. Por outro lado, a incorporação de uma redundância para o balanceador de carga (Figura 23) resulta em melhorias em todos os resultados das três métricas.

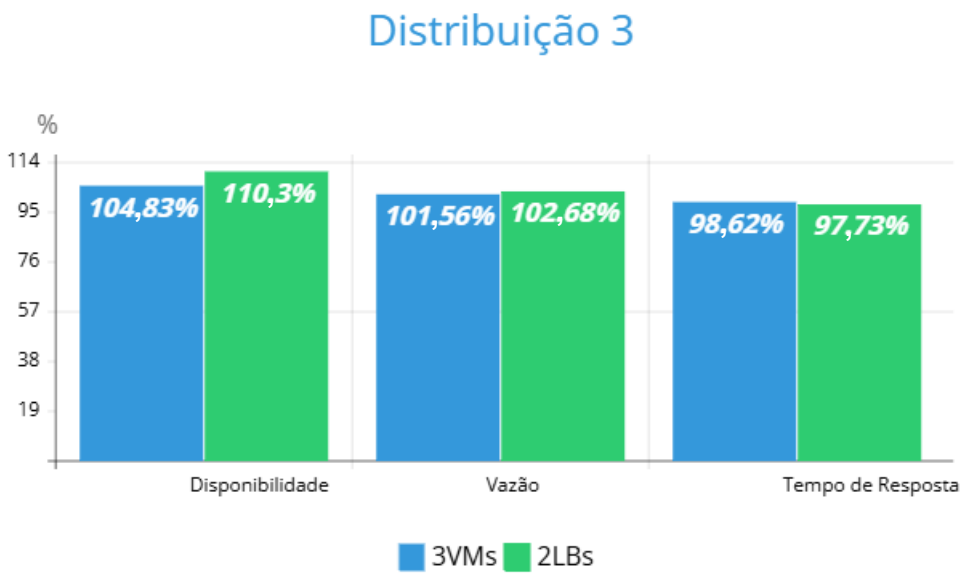

Figura 26 – Comparação Distribuição 3

6.2 Estudo de Caso 2

Este estudo de caso tem como objetivo avaliar a disponibilidade e o desempenho do sistema apresentado na Figura 18, modelado em SPN na Figura 19, utilizando uma abordagem fundamentada no Design de Experimentos (DoE – Design of Experiments). Para atingir esse objetivo, optou-se pela aplicação do design fatorial de dois níveis (2^k), que contempla a análise de dois níveis distintos para cada variável, sendo um nível elevado e outro reduzido.

O objetivo principal deste estudo é analisar a influência da falha de cada componente na disponibilidade e no desempenho do sistema. Para tanto, os MTTFs dos componentes do sistema foram definidos como os fatores a serem manipulados nos experimentos. O nível mais baixo de cada fator corresponde aos valores padrão de tempo de falha adotados para os componentes, conforme especificado na Tabela 4. O nível elevado foi definido como uma melhoria de 50% em relação a esses valores, ou seja, um aumento de 50% nos tempos médios até falhar.

A Tabela 15 apresenta os fatores adotados, sendo que a coluna "Nível 1 (baixo)"contém os valores padrão (MTTFs) dos fatores observados no sistema, enquanto a coluna "Nível 2 (alto)"reflete um acréscimo de 50% sobre os valores do Nível 1. Para isolar o impacto das falhas dos componentes, os tempos de reparo (MTTRs) foram mantidos constantes em todos os tratamentos experimentais, seguindo os valores apresentados na Tabela 4.

Para cada tratamento, que corresponde a uma combinação específica dos níveis dos fatores, é gerada uma versão distinta do modelo apresentado na Figura 19. Cada versão do modelo contém os valores dos MTTFs configurados conforme a combinação de níveis do

Tabela 15 – Fatores e Níveis

Fator	Nível 1 (h)	Nível 2 (h)
MTTF Host 1	1259	1888,5
MTTF Host 2	1259	1888,5
MTTF Balanceador	619,56	929,34
MTTF VMs de serviço	619,56	929,34

respectivo tratamento, conforme detalhado na Tabela 16. Considerando que a VM2 é uma redundância idêntica à VM1, ambas serão tratadas como um único fator (VMs de serviço), com seus MTTFs sendo ajustados conjuntamente durante os tratamentos experimentais.

O Tratamento 1 representa o sistema conforme sua configuração atual e serve como referência para a comparação com os resultados obtidos nos demais tratamentos. Como era esperado, o aumento nos MTTFs resulta em uma melhoria nas métricas avaliadas. O Tratamento 16, no qual todos os fatores estão configurados no nível 2, obteve os melhores resultados. A Tabela 16 apresenta os resultados de disponibilidade, vazão e tempo de resposta alcançados em cada um dos tratamentos.

Tabela 16 – Resultados DoE

	Fatores				Resultados		
Caso	ServiceVMs	LB	Host2	Host1	Disp	Vazão	T_Resp
1	619,56	619,56	1259	1259	2,4271	0,921	21,644
2	929,34	619,56	1259	1259	2,4750	0,92999	21,4468
3	619,56	929,34	1259	1259	2,5017	0,92835	21,4782
4	929,34	929,34	1259	1259	2,5086	0,93399	21,3487
5	619,56	619,56	1888,5	1259	2,5258	0,92592	21,5378
6	929,34	619,56	1888,5	1259	2,5214	0,93919	21,2314
7	619,56	929,34	1888,5	1259	2,5834	0,92637	21,5338
8	929,34	929,34	1888,5	1259	2,5952	0,93569	21,3216
9	619,56	619,56	1259	1888,5	2,4295	0,9303	21,421
10	929,34	619,56	1259	1888,5	2,4437	0,93026	21,423
11	619,56	929,34	1259	1888,5	2,5200	0,93351	21,361
12	929,34	929,34	1259	1888,5	2,5719	0,93954	21,231
13	619,56	619,56	1888,5	1888,5	2,5719	0,9305	21,4373
14	929,34	619,56	1888,5	1888,5	2,5361	0,93852	21,2493
15	619,56	929,34	1888,5	1888,5	2,6655	0,93452	21,3573
16	929,34	929,34	1888,5	1888,5	2,7011	0,93795	21,2828

Obs: ServiceVMs, LB, Host2 e Host1 em h, Disponibilidade em 9s, Vazão em req/s, Tempo de Resposta em s.

Após ordenar os efeitos obtidos por meio do design fatorial de forma decrescente, é possível identificar qual componente ou combinação de componentes exerce o maior impacto positivo quando há uma alteração no valor de seu MTTF. As Tabelas 17, 18 e 19 apresentam a classificação dos efeitos, incluindo os quatro mais impactantes.

A Tabela 17 apresenta a classificação dos efeitos conforme a métrica de dispo-

nibilidade. Nesta classificação, o MTTF do balanceador de carga (LB) demonstrou o maior impacto, com um valor de 0,000685, seguido pelo Host2, com um efeito de 0,000592. Assim, o balanceador de carga configura-se como o componente cuja falha provoca o maior impacto na disponibilidade do sistema.

Tabela 17 – Classificação de efeitos (Disponibilidade)

Fator/Iteração	Efeito
LB_MTTF	0,000685
Host2_MTTF	0,000592
Serv_VMs_MTTF	0,000215
Host2_MTTF*Serv_VMs_MTTF	0,000172

A Tabela 18 exibe a classificação dos efeitos segundo a métrica de vazão. De acordo com essa classificação, o MTTF do Host1 foi identificado como o mais significativo, apresentando um efeito de 0,006832, seguido pelo MTTF das VMs de serviço, com um efeito de 0,004325. Esses resultados indicam que a falha do Host1, que hospeda tanto o controlador na nuvem quanto as VMs de sistema, tem o maior impacto na vazão do sistema.

Tabela 18 – Classificação de efeitos (Vazão)

Fator/Iteração	Efeito
Host1_MTTF	0,006832
Serv_VMs_MTTF	0,004325
Host2_MTTF	0,00303
LB_MTTF	0,002715

A Tabela 19 apresenta a classificação dos efeitos conforme a métrica de tempo de resposta. Nessa classificação, observou-se que a maior influência está relacionada à combinação dos componentes Host2 e LB, com um efeito de 0,06945, enquanto o segundo maior impacto foi atribuído à combinação dos componentes Host1 e VMs de serviço, com um valor de 0,05685. Isso significa que a falha dos componentes Host2 e LB tem o maior impacto no tempo de resposta do serviço.

Tabela 19 – Classificação de efeitos (Tempo de Resposta)

Fator/Iteração	Efeito
Host2_MTTF*LB_MTTF	0,06945
Host1_MTTF*Serv_VMs_MTTF	0,05685
Host1_MTTF*Host2_MTTF*LB_MTTF	0,034
Host1_MTTF*Host2_MTTF*LB_MTTF*Serv_VMs_MTTF	0,02737

Ao cruzar as informações das tabelas que contêm a classificação dos efeitos apresentadas anteriormente, é possível determinar a importância de cada fator (componente) utilizado nos experimentos. Para as métricas de disponibilidade e tempo de resposta, o

balanceador de carga (LB) se destaca como o componente mais relevante, seguido pelo Host2. Dessa forma, ao utilizar versões desses componentes com maior MTTF, é possível alcançar melhores resultados. Por outro lado, para a métrica de vazão, a utilização de versões, com maior MTTF, do Host1 e das VMs com o serviço gera um impacto positivo mais significativo, apresentando uma vazão maior.

6.3 Estudo de Caso 3

Este estudo tem como objetivo analisar o sistema de forma abrangente e identificar o nível de impacto de cada componente na disponibilidade total do sistema. Para alcançar tal objetivo é imprescindível a utilização de um formalismo que se especialize na avaliação da relevância de cada componente para o sistema, com base em métricas de dependabilidade, como a disponibilidade. O RBD possibilita a criação de modelos adequados a essa finalidade, razão pela qual foi selecionado para o presente estudo.

Como os experimentos a serem conduzidos neste estudo demandam o uso do RBD, é necessário que o sistema seja representado por meio desse formalismo. Assim, o primeiro passo consiste na modelagem do sistema a ser analisado utilizando esse formalismo. O sistema em questão refere-se ao ambiente de experimentos ilustrado na Figura 18. Em seguida, foi realizada a avaliação desse modelo com base no índice de importância da disponibilidade (ID), que tem como objetivo identificar os componentes mais críticos do sistema. Dessa forma, o índice ID vai identificar aqueles componentes cujas falhas terão maior impacto sobre a disponibilidade do sistema.

Este estudo tem como foco identificar quais equipamentos possuem maior impacto na disponibilidade do sistema. Após a identificação do componente com o maior índice ID, é apropriado considerar a implementação de redundância para o mesmo. Dessa forma, os experimentos a seguir seguirão a seguinte sequência: (i) identificar o componente mais crítico do experimento, (ii) propor uma nova arquitetura com a redundância adequada ao componente identificado anteriormente, (iii) criar um novo modelo para representar a nova arquitetura (em RBD) e, por fim, (iv) avaliar o novo modelo para quantificar os impactos dos componentes levando em consideração a nova arquitetura. Esse processo será repetido até que se consiga obter a disponibilidade desejada ou até se ter as redundâncias máximas possíveis. Vale destacar que uma modelagem hierárquica poderá ser utilizada levando em consideração SPN e RBD quando forem mais indicados. Por exemplo, SPN é indicado para representar a redundância do sistema quando se tem dependência entre os equipamentos, ou seja, a falha de um dispositivo ativa um outro.

Dessa forma, esse estudo demonstra os benefícios obtidos na disponibilidade do sistema a partir da adição de redundância aos componentes mais críticos. Para isso, o presente estudo de caso apresentará a seguir uma série de experimentos. Cada um

desses experimentos terá a arquitetura representada a partir de modelos em RBD que vão representar as variações do ambiente ilustrado na Figura 18. Esses modelos farão uso dos valores de MTTF e MTTR apresentados na Tabela 4. Esses dados foram obtidos a partir de (MACIEL, 2016).

O experimento *e0* parte da arquitetura base apresentada na Figura 27 (a). Essa arquitetura considera o ambiente (Figura 18) com uma única máquina virtual (VM) fornecendo o serviço. A Tabela 20 apresenta o resultado do cálculo dos índice ID para o experimento *e0*. Esse cálculo foi realizado por meio da avaliação do modelo RBD apresentado na Figura 27 (b). De acordo com o resultado obtido, o componente VM1 foi avaliado como o dispositivo que mais impacta a disponibilidade do sitema. Sendo assim, a VM1 passa a ser um componente candidato a receber uma redundância a fim de se melhorar a disponibilidade na arquitetura que será proposta.

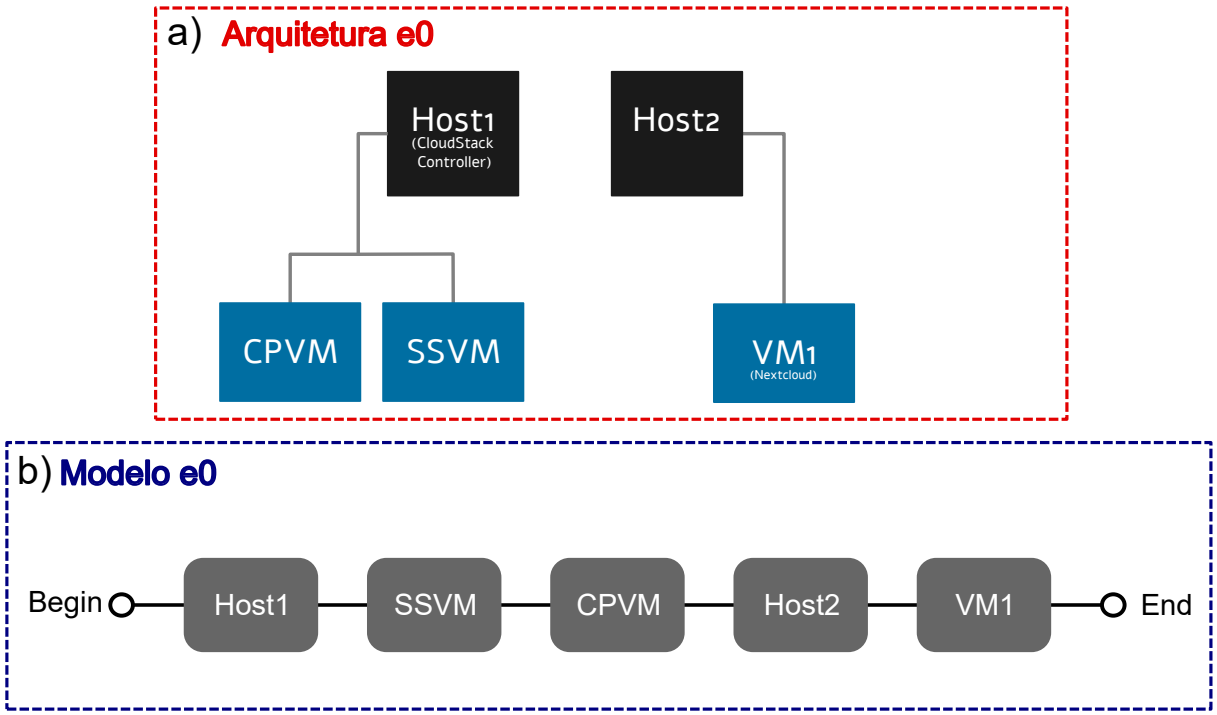

Figura 27 – Experimento 0 (e0)

Tabela 20 – Resultados do índice de ID do Experimento *e0*.

Equipamento	Índice ID	Número de 9s
Host1	0,997876155614076	2,67287730718104
SSVM	0,997348296266178	2,57647499985761
CPVM	0,997348296266178	2,57647499985761
Host2	0,997876155614076	2,67287730718106
VM1	**0,99861991091741**	**2,86009287966512**

Seguindo a recomendação de redundância indicadada pelo resultado anterior, o experimento *e1* considera uma nova arquitetura levando em consideração uma nova VM

(VM2) redundante. A Figura 28 (a) apresenta essa nova arquitetura *e1* que adiciona redundância em *Hot Standby* para a VM1 com o serviço na VM2. A fim de fazer um balanceamento de carga entre as duas VMs com o mesmo serviço ofertado, foi adicionado também um balanceador de carga (LB), cuja finalidade é distribuir de forma equilibrada as requisições ao serviço entre ambas as VMs (VM1 e VM2).

O modelo correspondente adotado em RBD para se computar o índice ID e também a disponibilidade dessa nova arquitetura é mostrado na Figura 28 (b). A Tabela 21 apresenta os resultados obtidos da avalição do índice ID para o experimento *e1*. A partir desses resultados é possível observar que o componente LB obteve o valor mais alto. Assim, o LB é identificado como o componente mais crítico, e a sua falha tem o maior impacto na disponibilidade do sistema. Sendo assim, ele é o componente indicado para ser replicado.

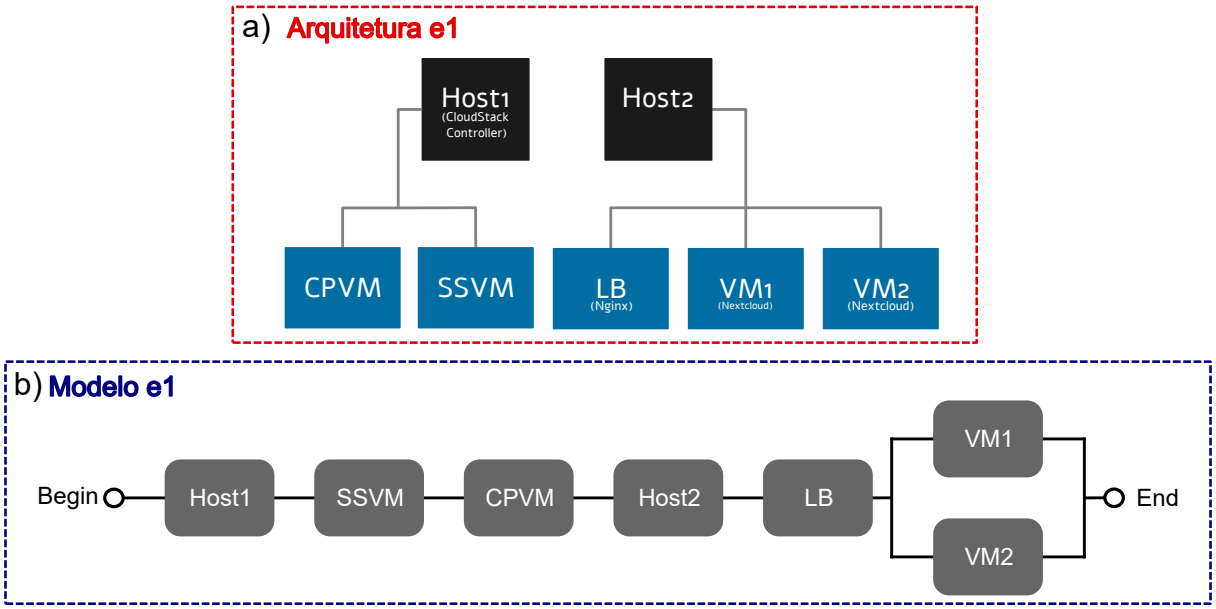

Figura 28 – Experimento 1 (e1)

Tabela 21 – Resultados do índice ID do experimento *e1*.

Equipamento	Índice ID	Número de 9s
Host1	0,997874326285834	2,67250339786188
SSVM	0,99734646790562	2,57617565522587
CPVM	0,99734646790562	2,57617565522587
Host2	0,997874326285835	2,6725033978619
LB	**0,998618080225701**	**2,85951716867713**
VM1	0,00135026589947473	0,000586809292749772
VM2	0,00135026589947473	0,000586809292749772

O experimento *e2* apresenta uma nova arquitetura conforme mostrada na Figura 29 (a). Essa arquitetura *e2* inclui uma redundância para o balanceador de carga, mas como não faz sentido haver dois balanceadores de carga ativos simultaneamente, o LB redundante estará configurado em *Cold Standby*. Assim, o redundante será ativado apenas

Tabela 22 – Resultados do índice ID para o Experimento *e2*.

Equipamento	Índice ID	Número de 9s
Host1	0,999150648154377	3,07091236516451
SSVM	0,998622114621815	2,86078690847493
CPVM	0,998622114621815	2,86078690847493
Host2	**0,999150648154378**	**3,07091236516457**
LB_cold	0,998618080225701	2,85951716867713
VM1	0,00135199294450472	0,000587560353656783
VM2	0,00135199294450472	0,000587560353656783

quando o balanceador de carga principal vier a falhar. Devido às limitações do RBD, não é possível modelar esse comportamento desejado com um componente em *Cold Standby*. Dessa forma, um modelo que leva em consideração esse LB e a sua redundância será avaliado externamente ao modelo em RBD apresentado na Figura 29 (b). Dessa forma, foi proposto um modelo em SPN que inclui o LB e sua redundância em *Cold Standby* conforme mostrado na Figura 29 (c). Esse modelo usa como parâmetros de entrada os tempos de falha e reparo presentes na Tabela 4 e a transição enable_redundant_LB usa 0,5h. A disponibilidade desse modelo em SPN é calculada e o seu resultado é aplicado no modelo RBD, especificamente no bloco correspondente ao componente LB, agora denominado LB_cold. Esse bloco LB_cold terá os seus valores substituídos pelos valores obtidos a partir da avaliação do modelo SPN. A disponibilidade desse modelo SPN é computada pela expressão P{(#LB_on=1)OR(#LB_cold_on=1)}.

A Tabela 22 apresenta o impacto de cada componente na arquitetura e2. Observa-se que, com a inclusão da redundância em *Cold Standby* para o LB, este deixa de ser o componente mais relevante, com o Host2 assumindo o papel de componente mais crítico, apresentando um valor mais alto segundo o índice de importância da disponibilidade. Assim, a falha do Host2 passa a ter o maior impacto na disponibilidade do sistema.

O Experimento *e3* considera o componente mais crítico identificado no Experimento *e2* e propõe uma extensão para a arquitetura *e2*. A arquitetura *e3* é apresentada na Figura 30 (a), e seu modelo RBD correspondente é mostrado na Figura 30 (b). Essa arquitetura e o modelo correspondente levam em consideração uma redundância para o componente Host2 em *Hot Standby*. Nesse caso, o novo componente denominado Host2_redun assume a função do Host2 quando este vier a falhar.

A Tabela 23 apresenta os respectivos resultados do cálculo do índice de importância para a disponibilidade para o Experimento *e3*. Neste resultado, o Host1 assume o papel de componente mais crítico, apresentando o valor mais alto. Assim, a falha do Host1 passa a ter o maior impacto na disponibilidade do sistema.

O Experimento *e4* propõe a implementação de uma redundância para o Host1. A Figura 31 (a) ilustra a arquitetura *e4* que inclui uma redundância para o componente

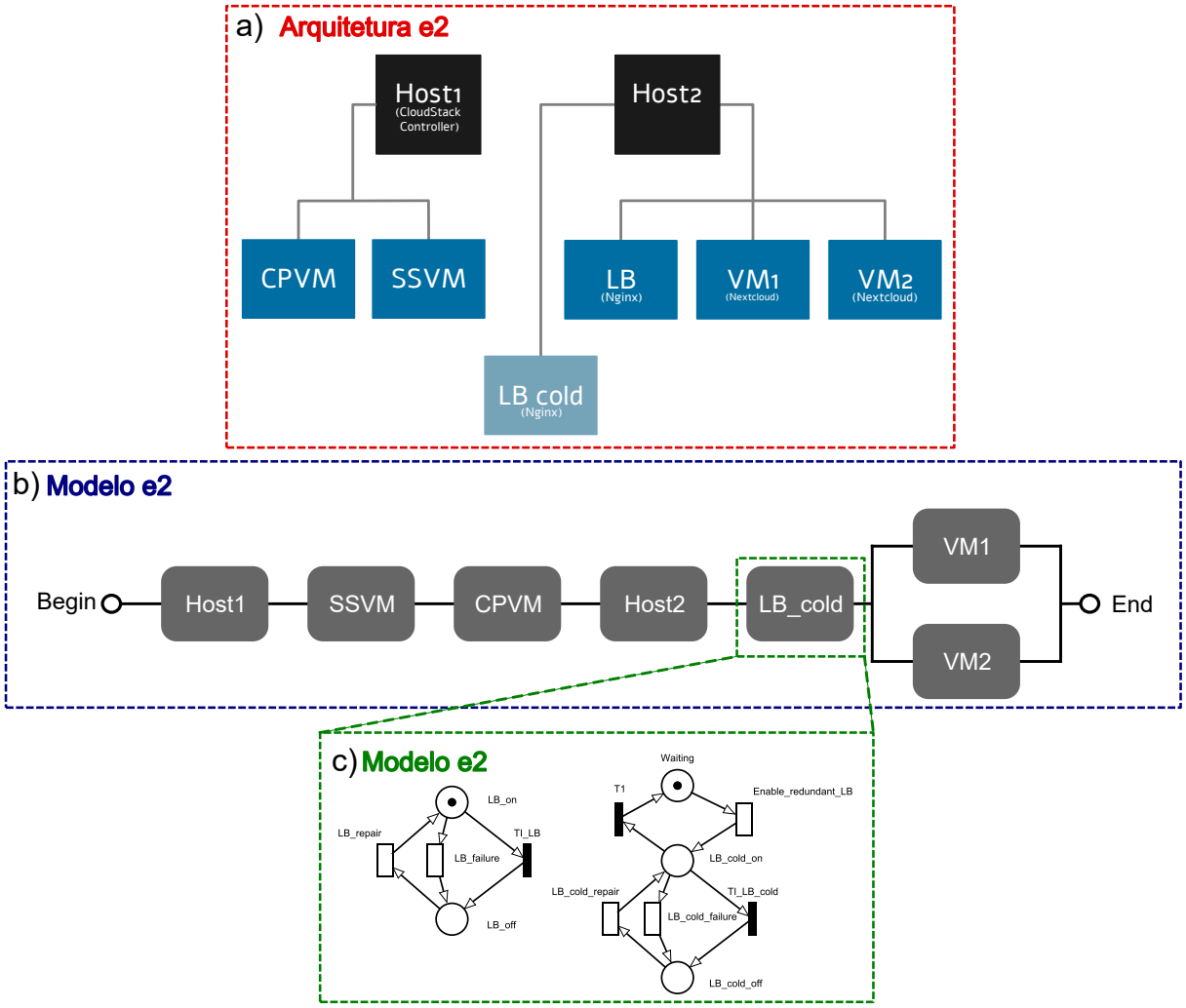

Figura 29 – Experimento 2 (e2)

Tabela 23 – Resultados do índice ID do Experimento *e3*.

Equipamento	Índice ID	Número de 9s
Host1	**0,999759766418838**	**3,61936628463453**
SSVM	0,999230910673175	3,11402321560257
CPVM	0,999230910673175	3,11402321560257
LB_cold	0,999226873817547	3,11174961879759
VM1	0,00135281716815661	0,00058791879419745
VM2	0,00135281716815661	0,00058791879419745
Host2	6,09E-04	0,000264617300881208
Host2_redun	6,09E-04	0,000264617300881208

Host1. Esse novo componente denominado Host1_redun é configurado em *Hot Standby* em relação ao componente Host1. A Figura 31 (b) apresenta o modelo correspondente e a Tabela 24 apresenta os resultados obtidos para os índices ID nesse sistema. A partir desses resultados, é possível perceber que as VMs de sistema, SSVM e CPVM, assumem juntas o papel de componentes com maior impacto, de acordo com o índice de importância da disponibilidade. No entanto, por serem parte do gerenciador da nuvem, essas VMs não

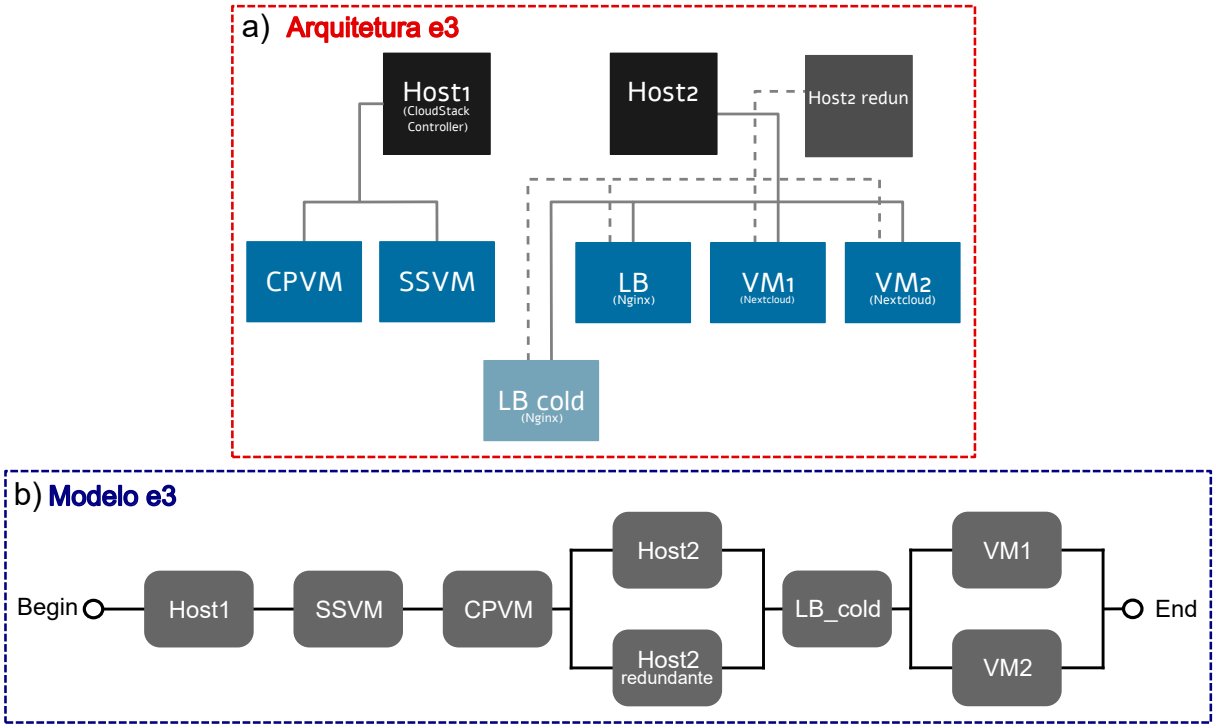

Figura 30 – Experimento 3 (e3)

estão sob controle total do projetista. Assim, o próximo componente com o ID mais alto irá se tornar o novo componente mais indicado a ser replicado. Nesse caso, esse equipamento é o LB, o qual possui o maior valor após as VMs de sistema. Como o LB já tem redundância, optamos por parar a proposição de novas arquiteturas.

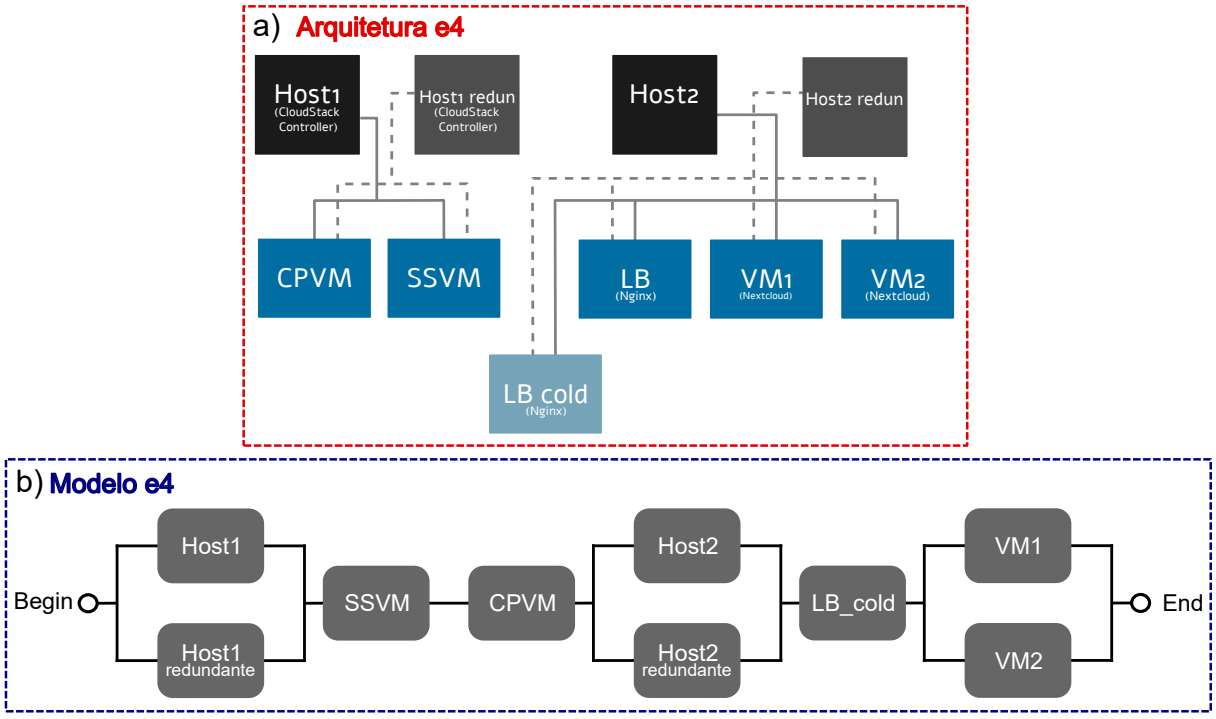

Figura 31 – Experimento 4 (e4)

Tabela 24 – Resultados do índice ID do Experimento *e4*.

Equipamento	Índice ID	Número de 9s
SSVM	0,999840077868561	3,79609143058219
CPVM	0,999840077868561	3,79609143058219
LB_cold	**0,999836038551921**	**3,78525825472449**
VM1	0,00135364189428477	0,000588277453552426
VM2	0,00135364189428477	0,000588277453552426
Host2	6,09E-04	0,000264778670316464
Host2_redun	6,09E-04	0,000264778670316464
Host1	6,09E-04	0,000264778670316416
Host1_redun	6,09E-04	0,000264778670316416

Resumidamente, os experimentos realizados visam analisar e identificar o componente mais crítico em cada versão do sistema. Em cada experimento subsequente, aplica-se uma redundância ao componente identificado como mais crítico, realiza-se a análise e identifica-se o novo componente mais crítico, que serve de base para o experimento seguinte. Abaixo, segue um resumo dos resultados encontrados nos experimentos:

- No *experimento 0*, é avaliado o índice de importância da disponibilidade do sistema em sua configuração mais simples, utilizando apenas uma VM com o serviço. Nesse experimento, o componente VM1 é identificado como o mais crítico.
- No *experimento 1*, é avaliado o índice de importância da disponibilidade do sistema em uma versão expandida, que inclui uma VM adicional com serviço e um balanceador de carga (LB). O componente LB é identificado como o mais crítico.
- No *experimento 2*, é aplicada uma redundância em Cold Standby para o LB, o componente mais crítico no experimento anterior, resultando na identificação do Host2 como o novo componente mais crítico.
- No *experimento 3*, é aplicada uma redundância em *Hot Standby* para o Host2, componente mais crítico no experimento anterior, levando à identificação do Host1 como o novo componente mais crítico.
- No *experimento 4*, é aplicada uma redundância em *Hot Standby* para o Host1, o componente mais crítico no experimento anterior, resultando na identificação das VMs de sistema como SSVM e CPVM como novos componentes mais críticos. No entanto, como estas VMs não estão sob o controle do projetista, o LB é considerado o novo componente mais crítico, por ser o próximo na lista.

A Tabela 25 apresenta os valores obtidos a partir das avaliações dos experimentos anteriores levando em consideração as métricas relativas à disponibilidade, incluindo o *downtime*, *uptime* e a disponibilidade em número de noves. É possível observar que, a partir

Tabela 25 – Resultados das Métricas dos Experimentos.

Experimento	Disponibilidade	Número de 9s	Uptime (horas)	Downtime (horas)
e0	0,9972678	2,5634898	8741,86294	23,94983
e1	0,9972659	2,5631993	8741,84691	23,96585
e2	0,9985415	2,8361024	8753,02809	12,78467
e3	0,9991503	3,0707225	8758,36425	7,44851
e4	0,9997594	3,6186951	8763,70367	2,10909

do experimento e1, os valores obtidos para essas métricas aumentam progressivamente. Isso ocorre porque, ao adicionar redundância aos componentes, a confiabilidade de cada sistema aumenta, elevando a disponibilidade geral.

Entretanto, é possível observar que houve uma redução de 0,0002% na disponibilidade ao estender o Experimento *e0* para o *e1*. Esse decréscimo é explicado pelo fato de que a adição de uma nova VM ao sistema implica na adição de um novo componente em série (LB) ao sistema que impacta negativamente a disponibilidade geral. Embora essa melhoria não tenha levado a um aumento imediato na disponibilidade, ela possibilitou ganhos em outras métricas, como aquelas relacionadas ao desempenho, as quais serão abordadas em trabalhos futuros.

A Figura 32 ilustra as melhorias na disponibilidade alcançadas com a aplicação das redundâncias nos experimentos. Tomando o experimento *e1* como valor de referência, a Figura 32 apresenta a porcentagem de melhora dos demais experimentos em relação ao valor atingido pelo sistema em sua configuração original. No experimento *e2* foi alcançada uma melhoria de aproximadamente 12,8% na disponibilidade. No experimento *e3*, a melhoria foi de cerca de 18,9% em relação ao e1. Por fim, o experimento *e4* obteve uma melhoria de 25% quando comparado ao e1.

6.4 Estudo de Caso 4

Este estudo tem como objetivo avaliar o ambiente experimental descrito na Seção 4.2 sob a perspectiva da chegada de clientes (usuários) a esse sistema. Para isso, é imprescindível que exista um modelo formal que represente o sistema e forneça uma abstração adequada para a dinâmica de chegada dos clientes.

A Figura 19 apresenta um modelo em SPN que representa o ambiente de experimentos, bem como o serviço configurado nele. No entanto, o modelo de desempenho apresentado na parte (b) dessa figura não inclui uma representação da chegada de clientes, assumindo sempre o número máximo de clientes presente no sistema. Portanto, é necessário desenvolver um novo modelo que atenda ao objetivo deste estudo, incorporando a dinâmica de chegada dos clientes.

A Figura 33 apresenta um novo modelo em SPN, desenvolvido para representar o

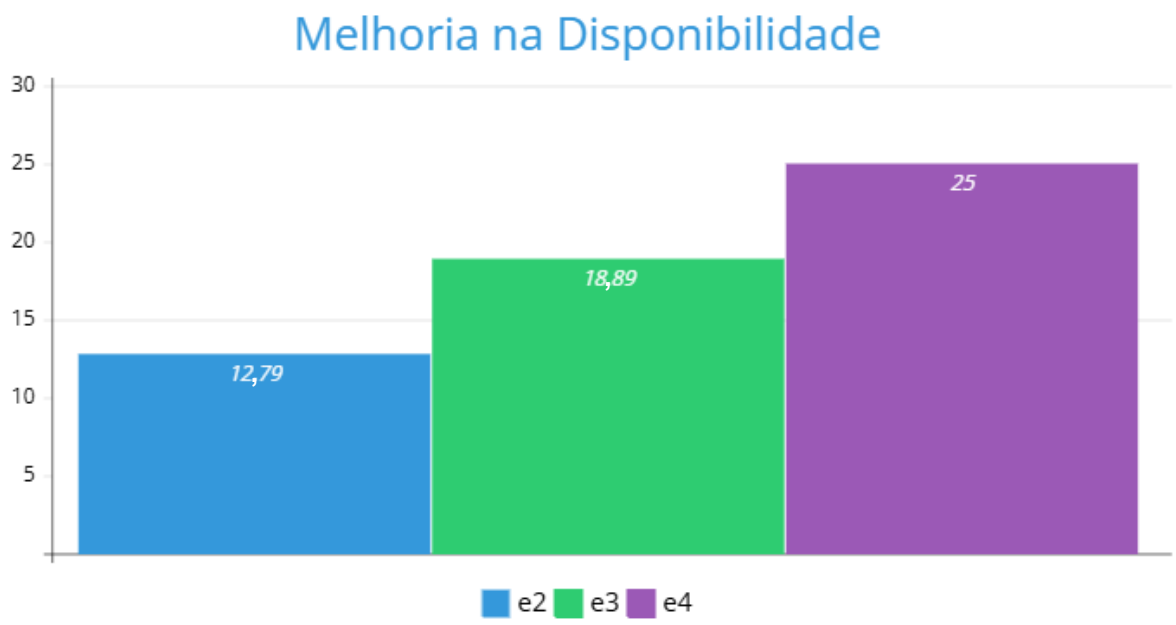

Figura 32 – Porcentagem Melhoria em relação a e1

sistema a ser avaliado. Esse modelo é uma adaptação do modelo de desempenho apresentado na parte (b) da Figura 19, com a adição da Transição Chegada_clientes, que representa o tempo médio necessário para que um usuário chegue ao serviço. Além disso, foi incorporado o Lugar F_clientes, que simboliza a fila de clientes aguardando para acessar o serviço. A Equação 6.1 apresenta a função que calcula a esperança de haver tokens no lugar F_clientes, ou seja, quanto um cliente esperaria para acessar o sistema em cada experimento.

Neste estudo de caso, serão atribuídos à Transição Chegada_clientes diferentes tempos médios de chegada, conforme apresentados na Tabela 26. O objetivo dessas variações é avaliar o impacto na fila de espera para acessar o serviço, permitindo verificar como as mudanças nos tempos de chegada influenciam o desempenho do sistema. Das linhas e1 a e6 o valor do tempo de chegada é reduzido pela metade em relação ao anterior, o e2 possui metade do tempo de chegada de e1, e3 tem a metada do tempo de e2 e assim por diante.

$$D = E\{\#F\_clientes\} \tag{6.1}$$

A linha e2 apresenta um tempo de chegada de aproximadamente 20,3 segundos, o qual é equivalente ao tempo de atendimento de um cliente pelo serviço, medido num ambiente real. A linha e1, por sua vez, representa um aumento de 100% em relação a linha e2, implicando que o cliente leva o dobro do tempo para chegar ao sistema.

A linha e3 introduz uma redução de 50% no tempo de chegada da linha e2, o que significa que um cliente leva metade do tempo para chegar e acessar o sistema. Já a linha e4 considera uma redução de 75% no tempo de chegada em relação ao valor da linha

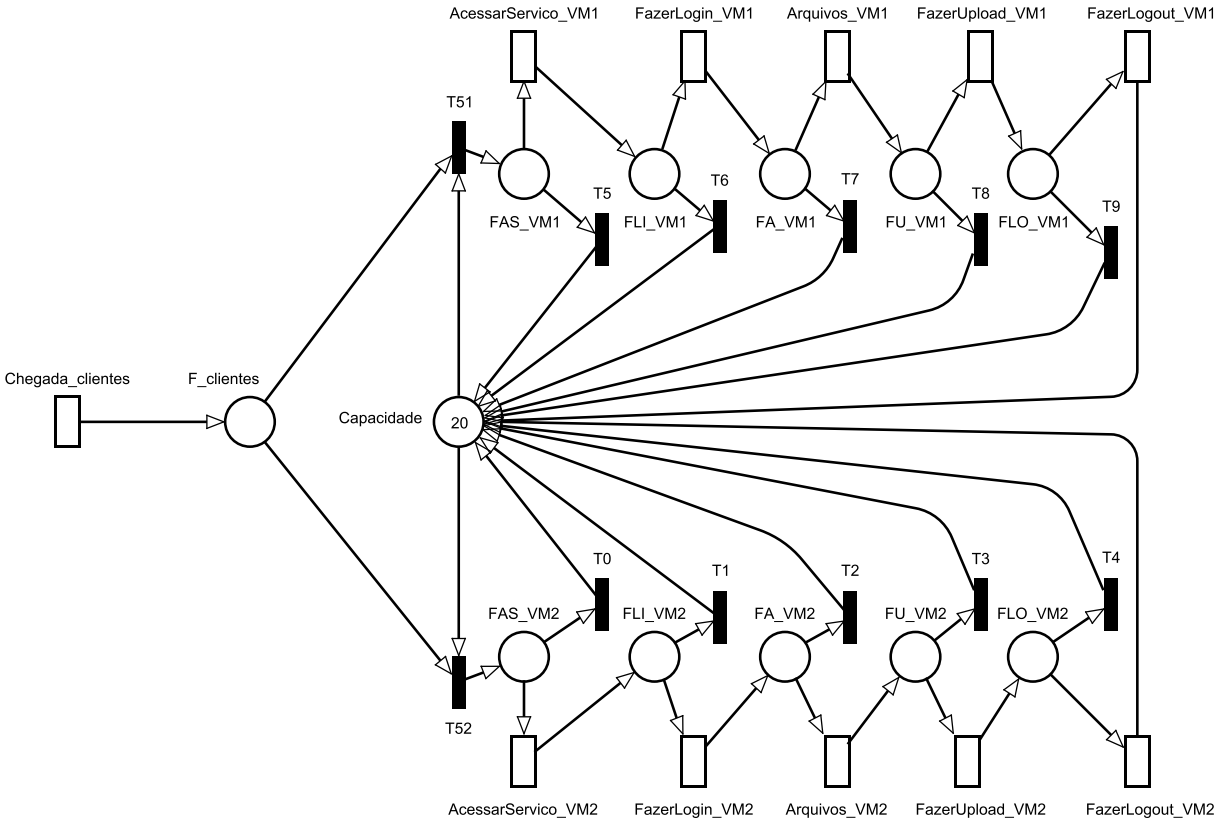

Figura 33 – Modelo com chegada de clientes

Tabela 26 – Resultados usando diferentes tempos de chegada

Experimento	Tempo de Chegada	Vazão	Tempo de Resposta	Fila
e1	40,582	0,02166	21,0214	0,000099
e2	20,291	0,04068	21,9128	0,000114
e3	10,1455	0,08347	21,4178	0,00037
e4	5,07275	0,16625	21,4211	0,000645
e5	2,53637	0,34028	21,3217	0,000717
e6	1,26818	0,67371	21,5161	0,418466
e7	1,15	0,73523	21,4593	1,125856

Obs: Tempo de Chegada em s, Vazão em req/s, Tempo de Resposta em s.

e2, fazendo com que o cliente leve apenas um quarto do tempo para chegar ao sistema e utilizá-lo.

A linha e5 considera uma redução de 50% no tempo de chegada em relação ao tempo da linha e4, neste experimento um novo cliente chega a uma taxa de 12,5% do tempo em que o cliente passa sendo atendido pelo sistema. Já a linha e6 corresponde a uma redução de 50% no tempo de chegada em relação ao tempo da linha e5, reduzindo o tempo de chegada para 6,25% do tempo de atendimento. Para finalizar, a linha e7 considera o tempo de 1,15 segundo.

Conforme ilustrado na Tabela 26, observa-se que, à medida que o tempo de chegada aumenta, a vazão é negativamente impactada. Por exemplo, a linha e1 apresenta

um tempo de espera duas vezes maior em relação à linha e2, o que resulta em uma vazão significativamente inferior na linha e1, atingindo aproximadamente metade do valor observado na linha e2. Em contrapartida, a fila de espera sofre uma redução, embora de forma menos expressiva, uma vez que sua variação é menos sensível. Essa diminuição nos valores é atribuída à presença reduzida de clientes no sistema, decorrente do aumento no tempo de chegada.

Analisando a Tabela 26 sob a perspectiva oposta, quando o tempo de chegada é reduzido, observa-se um impacto positivo na vazão, resultando em valores mais elevados. Por exemplo, a linha e3 apresenta um tempo de espera metade do registrado na linha e2, o que favorece a vazão da linha e3, que atinge aproximadamente o dobro da vazão observada na linha e2. Por outro lado, a fila de espera é negativamente afetada, com seus valores também sendo aumentados. Ambas as métricas apresentam elevações à medida que o tempo de chegada dos clientes ao sistema diminui.

A Figura 34 ilustra a relação entre os valores de vazão e tempo de chegada apresentados na Tabela 26. Como evidenciado na figura, a vazão exibe uma relação inversamente proporcional ao tempo de chegada. Em outras palavras, à medida que o tempo de chegada diminui, a vazão aumenta, resultando em melhores desempenhos para essa métrica. Em cada novo experimento, o valor do tempo de chegada é reduzido pela metade em comparação com o experimento anterior, o que implica um aumento correspondente na vazão, com os valores chegando a aproximadamente o dobro daqueles observados no experimento anterior. A exceção ocorre no experimento 7 (e7), que não segue a redução pela metade do tempo do experimento anterior, conforme visualizado na Figura 34, ponto no qual o padrão de aumento da vazão deixa de ser observado.

A Figura 35 ilustra a relação entre os valores da fila de espera e o tempo de chegada, conforme apresentado na Tabela 26. É possível observar que, assim como a vazão, a fila de espera também apresenta uma relação inversamente proporcional ao tempo de chegada. No entanto, ao contrário da vazão, a fila de espera tende a apresentar resultados mais desfavoráveis à medida que o tempo de chegada é reduzido, ou seja, maiores filas de espera. Como pode ser observado na Figura 35, inicialmente, a fila não sofre variações significativas até o experimento e6, quando ocorre um aumento substancial em seu valor. A partir do experimento 7 (e7), conforme ilustrado na mesma figura, começa a ser observada a presença de clientes em espera.

À medida que o tempo de chegada aumenta, observa-se uma diminuição tanto na fila de espera quanto na vazão do sistema. Por outro lado, à medida que o tempo de chegada é reduzido, tanto a fila de espera quanto a vazão do sistema aumentam. A partir de um tempo de chegada de aproximadamente 1,15 segundos, conforme evidenciado pelo experimento 7, o sistema descrito na Seção 4.2 começa a apresentar uma fila de espera superior a um. Em outras palavras, a partir deste ponto, clientes não conseguirão acessar

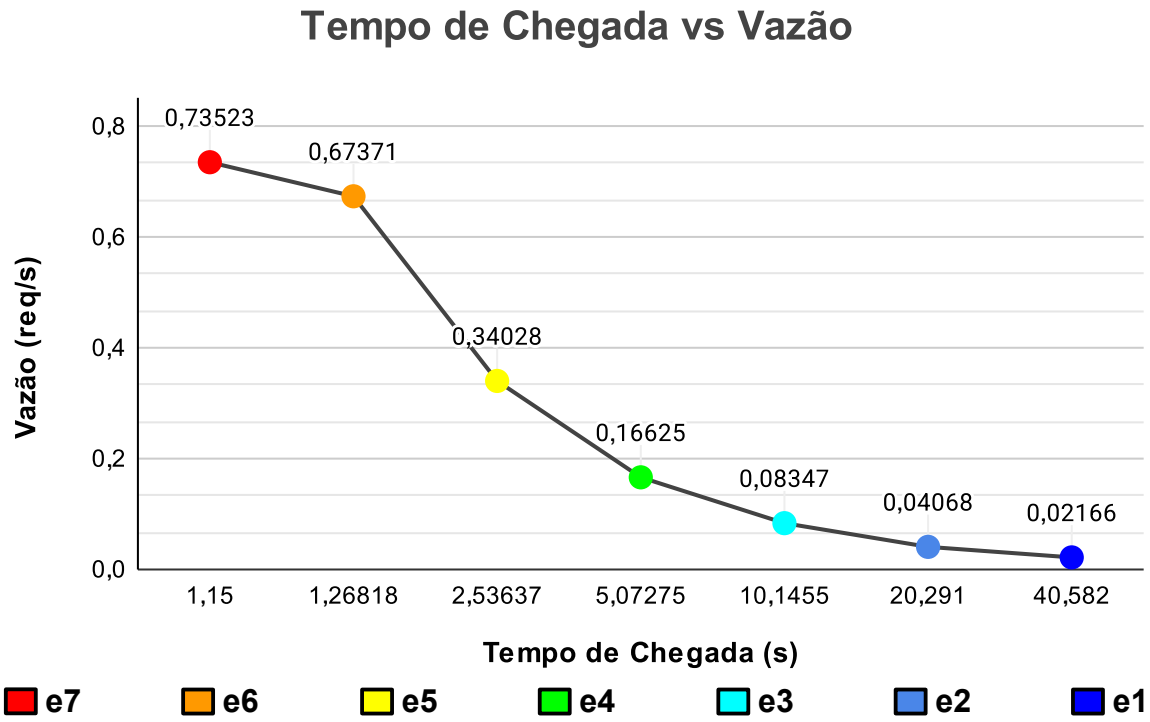

Figura 34 – Tempo de Chegada vs Vazão

o sistema imediatamente ao chegarem. Esse fenômeno ocorre devido ao aumento na taxa de chegada de clientes, o que intensifica a concorrência pelo acesso ao sistema, resultando em tempos de espera mais elevados para os usuários.

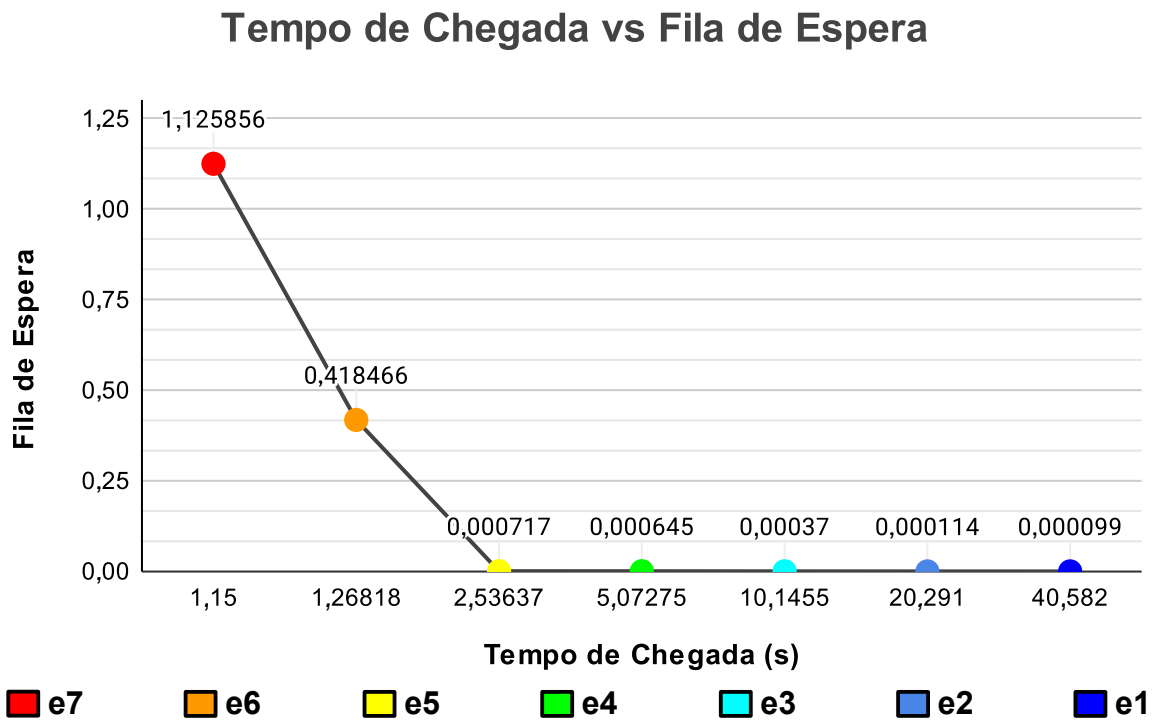

Figura 35 – Tempo de Chegada vs Fila

7 Conclusões

Este trabalho propôs modelos para a avaliação de desempenho e disponibilidade de um serviço hospedado em nuvem privada. Sendo o serviço Nextcloud escolhido para realizar experimentos, onde foi hospedado em um ambiente de nuvem privada Apache CloudStack. A abordagem adotada utiliza SPN e RBD para avaliar a arquitetura proposta.

O processo seguiu uma sequência lógica, que começou com a compreensão do sistema, passando pela definição de métricas e coleta de dados, e culminando na construção de modelos e na realização de experimentos. A partir dessa base, foram realizados estudos de caso sobre os modelos propostos.

O primeiro estudo analisou o impacto da disponibilidade em um modelo de desempenho, considerando diferentes formas de distribuição das VMs entre os hosts. Foi proposta uma extensão para o modelo, resultando em melhores resultados com a distribuição das VMs no Host2. O segundo estudo utilizou o projeto de experimentos (DoE) para avaliar a relevância de um componente, considerando uma melhoria em seu tempo médio de falha (MTTR), identificando que o balanceador de carga tem o maior impacto tanto na disponibilidade quanto no desempenho do sistema. O terceiro estudo utilizou o índice de importância da disponibilidade para identificar os componentes mais críticos e propor redundâncias, resultando em melhorias no desempenho e na disponibilidade do sistema. O quarto estudo analisou o comportamento do sistema com a inclusão da chegada de clientes, concluindo que, quanto menor o tempo de chegada dos clientes, maior o tempo de espera, mas também maior a vazão.

7.1 Contribuições

As principais contribuições presentes neste trabalho são:

- Modelo proposto em SPN que combina um modelo de disponibilidade impactando no modelo de desempenho.
- Proposição da extenção do modelo SPN acrescentando uma instância adicional com o serviço para melhorar o desempenho.
- Proposição da extenção do modelo SPN com o acréscimo de uma redundância no balanceador de carga com a finalidade de aumentar a disponibilidade.
- Modelo proposto em RBD para análise da disponibilidade do sistema segundo o índice de importância da disponibilidade.

- Proposição de extenções do modelo em RBD que consideram redundâncias para os componentes de maior impacto na disponibilidade.
- Proposição da extenção do modelo SPN considerando a chegada de clientes ao sistema.
- A metodologia proposta que traz uma abordagem que faz uso de um ambiente real para extrair dados a serem usados em modelos.

7.2 Trabalhos futuros

A seguir nesta seção são apresentadas as possíveis abordagens para trabalhos futuros:

- Este trabalho utilizou o Apache Cloudstack como ambiente de nuvem privada. Assim, pode-se comparar o desempenho deste com outros ambientes de nuvem privada, como Openstack e Eucalyptus.
- Utilização de uma máquina física projetada especificamente para hospedar um servidor, e comparar o desempenho contra um servidor que combina o poder computacional de máquinas desktop como o utilizado neste trabalho.
- Adicionar a análise do consumo energético como métrica adicional para avaliar as melhores configurações de distribuição de VMs.
- Analisar nuvens híbridas, pois será possível aumentar a disponibilidade e o desempenho. Além disso, a parte pública pode ser utilizada somente quando a privada não estiver disponível.
- Analisar o custo como métrica a ser levada em consideração.

7.3 Publicações

A seguir são apresentados os artigos gerados a partir do presente trabalho:

- Stochastic Petri Net Models for Availability and Performance Evaluation of Nextcloud Service hosted in Apache Cloudstack. LADC 2024, status: publicado.
- Avaliação da Disponibilidade do Serviço Nextcloud Hospedado em Nuvem Privada. SBRC 2025, status: submetido.

Referências

ANDRADE, E. et al. Performance and availability trade-offs in fog–cloud iot environments. *Journal of Network and Systems Management*, Springer, v. 29, p. 1–27, 2021. Citado 3 vezes nas páginas 30, 31 e 32.

APACHE. *Apache Cloudstack*. 2012. Disponível em: <http://https://cloudstack.apache.org/>. Citado 2 vezes nas páginas 4 e 7.

ARAÚJO, G. et al. Vehicular cloud computing networks: Availability modelling and sensitivity analysis. *International Journal of Sensor Networks*, Inderscience Publishers (IEL), v. 36, n. 3, p. 125–138, 2021. Citado 3 vezes nas páginas 27, 31 e 32.

ARAUJO, J. et al. Availability evaluation of digital library cloud services. In: IEEE. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. [S.l.], 2014. p. 666–671. Citado na página 6.

BARROS, J. R. A. d. et al. Modelagem de desempenho do banco de dados cassandra. Universidade Federal Rural de Pernambuco, 2018. Citado na página 18.

BAUER, E.; ADAMS, R. *Reliability and availability of cloud computing*. [S.l.]: John Wiley & Sons, 2012. Citado na página 2.

CALLOU, G. et al. Energy consumption and execution time estimation of embedded system applications. *Microprocessors and Microsystems*, Elsevier, v. 35, n. 4, p. 426–440, 2011. Citado na página 18.

CLEMENTE, D. et al. Availability evaluation of system service hosted in private cloud computing through hierarchical modeling process. *The Journal of Supercomputing*, Springer, v. 78, n. 7, p. 9985–10024, 2022. Citado 2 vezes nas páginas 28 e 32.

CLOUD, H. The nist definition of cloud computing. *National institute of science and technology, special publication*, v. 800, n. 2011, p. 145, 2011. Citado 2 vezes nas páginas 6 e 7.

CRICK, M.; HILL, M. The role of sensitivity analysis in assessing uncertainty. In: *Uncertainty analysis for performance assessments of radioactive waste disposal systems*. [S.l.: s.n.], 1987. Citado na página 19.

CUNHA, A. et al. Modelagem hierárquica e heterogênea para avaliação de disponibilidade de aplicações big data na nuvem privada. In: SBC. *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance)*. [S.l.], 2021. p. 49–60. Citado na página 25.

DANTAS, J. et al. Estimating capacity-oriented availability in cloud systems. *International Journal of Computational Science and Engineering*, Inderscience Publishers (IEL), v. 22, n. 4, p. 466–476, 2020. Citado na página 2.

DESAI, T.; PRAJAPATI, J. A survey of various load balancing techniques and challenges in cloud computing. *International Journal of Scientific & Technology Research*, v. 2, n. 11, p. 158–161, 2013. Citado 2 vezes nas páginas 9 e 10.

DESEL, J.; REISIG, W. Place/transition petri nets. In: SPRINGER. *Advanced Course on Petri Nets*. [S.l.], 1996. p. 122–173. Citado na página 10.

DISTEFANO, S.; SCARPA, M.; PULIAFITO, A. Modeling distributed computing system reliability with drbd. In: IEEE. *2006 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06)*. [S.l.], 2006. p. 106–118. Citado na página 14.

F5, Inc. *NGINX - High Performance Load Balancer, Web Server, Reverse Proxy*. 2024. <https://www.nginx.com>. Accessed: 2025-05-28. Citado na página 4.

FÉ, I. et al. Performance-cost trade-off in auto-scaling mechanisms for cloud computing. *Sensors*, MDPI, v. 22, n. 3, p. 1221, 2022. Citado 2 vezes nas páginas 29 e 32.

FRANK, P. M.; ESLAMI, M. Introduction to system sensitivity theory. *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE, v. 10, n. 6, p. 337–338, 1980. Citado na página 19.

GERMAN, R. *Performance analysis of communication systems with non-Markovian stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 2000. Citado na página 12.

HAMBY, D. M. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment*, Springer, v. 32, p. 135–154, 1994. Citado na página 20.

IOOSS, B.; LEMAÎTRE, P. A review on global sensitivity analysis methods. *Uncertainty management in simulation-optimization of complex systems: algorithms and applications*, Springer, p. 101–122, 2015. Citado na página 20.

JOHNSON, B. W. *Design & analysis of fault tolerant digital systems*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1988. Citado 2 vezes nas páginas 23 e 24.

KLEIJNEN, J. P. Sensitivity analysis and optimization in simulation: design of experiments and case studies. In: *Proceedings of the 27th conference on Winter simulation*. [S.l.: s.n.], 1995. p. 133–140. Citado 2 vezes nas páginas 19 e 20.

MACIEL, P. et al. A survey on reliability and availability modeling of edge, fog, and cloud computing. *Journal of Reliable Intelligent Environments*, Springer, p. 1–19, 2022. Citado 3 vezes nas páginas 28, 31 e 32.

MACIEL, P.; TRIVEDI, K.; KIM, D. Dependability modeling in: Performance and dependability in service computing: Concepts, techniques and research directions. *Hershey: IGI Global, Pennsylvania, USA*, v. 13, p. 1380, 2010. Citado na página 22.

MACIEL, P. R. M. Modeling availability impact in cloud computing. *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*, Springer, p. 287–320, 2016. Citado 2 vezes nas páginas 40 e 58.

MANCAŞ, C. Performance analysis in private and public cloud infrastructures. In: IEEE. *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. [S.l.], 2019. p. 1–6. Citado 3 vezes nas páginas 28, 31 e 32.

MASON, R. L.; GUNST, R. F.; HESS, J. L. *Statistical design and analysis of experiments: with applications to engineering and science*. [S.l.]: John Wiley & Sons, 2003. Citado na página 20.

MATOS, R. et al. Bottleneck detection in cloud computing performance and dependability: Sensitivity rankings for hierarchical models. *Journal of Network and Systems Management*, Springer, v. 28, n. 4, p. 1839–1871, 2020. Citado 2 vezes nas páginas 29 e 32.

MELO, C. et al. Performance and availability evaluation of the blockchain platform hyperledger fabric. *The Journal of Supercomputing*, Springer, v. 78, n. 10, p. 12505–12527, 2022. Citado 3 vezes nas páginas 30, 31 e 32.

MONTGOMERY, D. C. *Design and analysis of experiments*. [S.l.]: John wiley & sons, 2017. Citado 2 vezes nas páginas 20 e 21.

MONTGOMERY, D. C.; RUNGER, G. C. *Applied statistics and probability for engineers*. [S.l.]: John wiley & sons, 2010. Citado na página 20.

MORE, N. S.; HIRAY, S. R. Load balancing and resource monitoring in cloud. In: *Proceedings of the CUBE international information technology conference*. [S.l.: s.n.], 2012. p. 552–556. Citado na página 8.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. Citado na página 10.

Nextcloud GmbH. *Nextcloud - A safe home for all your data*. 2024. <https://nextcloud.com>. Accessed: 2025-05-28. Citado na página 4.

NGUYEN, T. A. et al. Performability evaluation of load balancing and fail-over strategies for medical information systems with edge/fog computing using stochastic reward nets. *Sensors*, MDPI, v. 21, n. 18, p. 6253, 2021. Citado 3 vezes nas páginas 29, 31 e 32.

PAPADOPOULOS, A. V. et al. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Transactions on Software Engineering*, IEEE, v. 47, n. 8, p. 1528–1543, 2019. Citado 3 vezes nas páginas 29, 31 e 32.

PEREIRA, P. et al. Analytical models for availability evaluation of edge and fog computing nodes. *The Journal of Supercomputing*, Springer, v. 77, p. 9905–9933, 2021. Citado 2 vezes nas páginas 27 e 32.

PEREIRA, P. et al. Availability model for edge-fog-cloud continuum: an evaluation of an end-to-end infrastructure of intelligent traffic management service. *The Journal of Supercomputing*, Springer, p. 1–28, 2022. Citado 2 vezes nas páginas 28 e 32.

PETRI, C. A. Kommunikation mit automaten. 1962. Citado na página 10.

PINHEIRO, T. et al. Performance and resource consumption analysis of elastic systems on public clouds. In: IEEE. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. [S.l.], 2019. p. 2115–2120. Citado 3 vezes nas páginas 28, 31 e 32.

RODRIGUES, L. et al. Performance and availability evaluation of an smart hospital architecture. *Computing*, Springer, v. 103, p. 2401–2435, 2021. Citado 2 vezes nas páginas 30 e 32.

ROGERS, G.; BADHAM, L. Evaluation in the management cycle. *Improving educational management through research and consultancy. London: Paul Chapman Publishing*, 1994. Citado na página 18.

ROSENDO, D. et al. A methodology to assess the availability of next-generation data centers. *The Journal of Supercomputing*, Springer, v. 75, p. 6361–6385, 2019. Citado na página 22.

RUSCHEL, H.; ZANOTTO, M. S.; MOTA, W. d. C. Computação em nuvem. *Pontifícia Universidade Católica do Paraná, Curitiba, Brazil*, 2010. Citado na página 6.

SAHNER, R.; TRIVEDI, K. S.; PULIAFITO, A. Performance and reliability analysis of computer systems (an example-based approach using the sharpe software. *IEEE Transactions on Reliability*, IEEE, v. 46, n. 3, p. 441–441, 1997. Citado na página 13.

SANTOS, D. M. L. dos; VALE, K. M. A. C.; ALENCAR, F. M. R. de. Avaliação de desempenho de nuvens privadas: um comparativo entre owncloud, nextcloud e pydio. *Brazilian Journal of Development*, v. 6, n. 6, p. 40549–40566, 2020. Citado 3 vezes nas páginas 29, 31 e 32.

SANTOS, L. et al. Event-based moving target defense in cloud computing with vm migration: A performance modeling approach. *IEEE Access*, IEEE, 2024. Citado 2 vezes nas páginas 28 e 32.

SHOYARI, M. F. et al. Availability modeling in redundant openstack private clouds. *Software: Practice and Experience*, Wiley Online Library, v. 51, n. 6, p. 1218–1241, 2021. Citado 3 vezes nas páginas 27, 31 e 32.

SILVA, A. da et al. Análise de desempenho de servidores de arquivos em nuvem privada. In: SBC. *Anais do L Seminário Integrado de Software e Hardware*. [S.l.], 2023. p. 272–283. Citado na página 41.

SILVA, B. A framework for availability, performance and survivability evaluation of disaster tolerant cloud computing systems. Universidade Federal de Pernambuco, 2016. Citado na página 11.

SOUSA, E. et al. Performance and availability evaluation of big data environments in the private cloud. *Journal of Computer and Communications*, Scientific Research Publishing, v. 12, n. 12, p. 266–288, 2024. Citado 2 vezes nas páginas 31 e 32.

The Apache Software Foundation. *Apache JMeter - Performance and Load Testing Tool*. 2024. <https://jmeter.apache.org>. Accessed: 2025-05-28. Citado na página 35.

TORQUATO, M.; MACIEL, P.; VIEIRA, M. Model-based performability and dependability evaluation of a system with vm migration as rejuvenation in the presence of bursty workloads. *Journal of Network and Systems Management*, Springer, v. 30, n. 1, p. 3, 2022. Citado 2 vezes nas páginas 31 e 32.

TORRES, E. B. et al. Performance and availability evaluation of storage services in private cloud. In: IEEE. *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.], 2016. p. 1–6. Citado na página 18.

TRIVEDI, K. S. et al. Reliability analysis techniques explored through a communication network example. North Carolina State University. Center for Advanced Computing and Communication, 1996. Citado na página 14.

WASEEM, Q. et al. Quantitative analysis and performance evaluation of target-oriented replication strategies in cloud computing. *Electronics*, MDPI, v. 10, n. 6, p. 672, 2021. Citado 2 vezes nas páginas 30 e 32.

XIE, M.; DAI, Y.-S.; POH, K.-L. *Computing system reliability: models and analysis*. [S.l.]: Springer Science & Business Media, 2004. Citado na página 14.