Avaliação da Disponibilidade do Serviço Nextcloud Hospedado em Nuvem Privada

Wenderson de Souza Leonardo[1], **Gustavo Callou**[1]

[1]Universidade Federal Rural de Pernambuco (UFRPE)

{wenderson.leonardo,gustavo.callou}@ufrpe.br

Abstract. *Cloud computing has become a strategic solution to address the growing demands for connectivity and flexibility in accessing computational resources. Consequently, it is necessary to identify the components with the greatest impact on the system and implement redundancies efficiently, ensuring high availability, optimal performance, and cost reduction. This paper proposes the use of Reliability Block Diagrams (RBD) to assess the dependability of a private cloud environment, focusing on the metrics of availability, uptime, and downtime. The case study aims to identify and analyze critical components for system availability, applying redundancies to these components to reduce downtime and increase uptime. The results demonstrated that the application of redundancies can lead to an increase of up to 25% in system availability.*

Resumo. *A computação em nuvem tem se consolidado como uma solução estratégica para atender às crescentes demandas de conectividade e flexibilidade no acesso a recursos computacionais. Nesse contexto, surge a necessidade de identificar os componentes de maior impacto no sistema e implementar redundâncias de forma eficiente, garantindo alta disponibilidade, desempenho ideal e redução de custos. O presente trabalho propõe a utilização de Diagramas de Bloco de Confiabilidade (RBD) para avaliar a dependabilidade de um ambiente de nuvem privada, com ênfase nas métricas de disponibilidade, uptime e downtime. O estudo de caso visa identificar e analisar os componentes críticos para a disponibilidade do sistema, aplicando redundâncias nesses componentes, com o objetivo de reduzir o tempo de inatividade (downtime) e aumentar o tempo de operação (uptime). Os resultados demonstraram que a aplicação das redundâncias pode levar a um aumento de até 25% na disponibilidade do sistema.*

1. Introdução

Com o avanço constante da tecnologia, os dispositivos eletrônicos têm se tornado cada vez mais essenciais na vida cotidiana de grande parte da população mundial. Não se limitando ao uso pessoal, computadores e celulares passaram a ser utilizados em diversas áreas, desde indústrias automotivas até pecuária e agricultura. Além disso, a crescente demanda por conectividade tem impulsionado os usuários a recorrerem a soluções online para resolver uma ampla variedade de problemas de maneira prática e rápida [Asio et al. 2021].

Ambientes de computação em nuvem estão emergindo como soluções eficazes para atender a essas necessidades de resolução de problemas via web. Computação em

nuvem é um modelo computacional que permite acesso conveniente e sob demanda a um conjunto de recursos computacionais configuráveis [Cloud 2011]. Esses ambientes são caracterizados por alta performance, escalabilidade e disponibilidade, oferecendo flexibilidade para atender a diferentes demandas.

Devido as 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 em nuvem privados. Dessa forma, as empresas ganham maior controle sobre os serviços e a segurança de suas informações. No entanto, esses serviços privados exigem uma alta demanda por disponibilidade e confiabilidade, uma vez que são essenciais para garantir a continuidade das operações e a integridade dos dados. Logo, para oferecer um serviço de computação em nuvem privado, é fundamental contar com recursos e desempenho adequados, que atendam à demanda e assegurem a qualidade e segurança do serviço prestado.

A medida em que as organizações adotam essa tecnologia, a complexidade dos sistemas em nuvem cresce, o que torna a análise detalhada de sua arquitetura cada vez mais necessária. A falha de um sistema sem redundâncias pode gerar uma interrupção no serviço prestado que pode durar horas, então para evitar falhas em serviços críticos um sistema deve implantar redundâncias e assim reduzir o tempo de inatividade [Bauer and Adams 2012]. Identificar os componentes críticos, cuja falha pode causar interrupções graves ou perda de dados, de um sistema em nuvem é essencial para mitigar riscos e garantir o desempenho e a segurança dos sistemas em nuvem.

A implementação de redundância em componentes críticos é uma estratégia eficaz para reduzir riscos e aumentar a resiliência do sistema. A redundância, que consiste na duplicação de componentes ou funções, não só garante a continuidade da disponibilidade dos serviços, mas também oferece uma camada extra de proteção contra falhas inesperadas. Este artigo examina as razões e benefícios de realizar uma análise sistemática dos sistemas em nuvem, ressaltando a importância de identificar os componentes críticos e adotar estratégias de redundância. De forma mais específica, as contribuições principais deste trabalho são:

- Proposição de modelos formais em diagramas de bloco de confiabilidade (RBD) e em redes de Petri estocástica (SPN) para computar a disponibilidade de serviço em nuvem privada.
- A partir dos modelos propostos, é utilizada uma estratégia para identificação dos componentes de maior impacto para a disponibilidade do sistema.
- Proposição de melhorias a partir de técnicas de redundânica (*cold* e *hot standby*) a fim de melhorar a disponibilidade do sistema.

Este trabalho está organizado da maneira a seguir. Seção 2 explana alguns conceitos necessários para uma melhor compreensão deste trabalho. Seção 3 apresenta trabalhos similares encontrados na literatura. Seção 4 descreve a arquitetura do ambiente de experimentos. Seção 5 apresenta o estudo de caso e os modelos utilizados nele. Seção 6 conclui o artigo e fala de objetivos futuros para esta pesquisa.

2. Referencial Teórico

Essa seção apresenta conceitos fundamentais para o entendimento deste trabalho.

2.1. Rede de Petri Estocástica

As redes de Petri (PN) 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]. As Redes de Petri Estocásticas (SPN) [Haas 2006] estendem as PNs ao incorporar o conceito de tempo em seus modelos.

A Figura 1 apresenta os elementos fundamentais das SPNs. 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. 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. Na transição estocástica pode ser associado um tempo, o qual representa o intervalo necessário até que a transição esteja habilitada, correspondendo ao período de execução da atividade. O arco inibidor permite que o modelo verifique a presença de tokens no lugar de origem, inibindo a ativação da transição conectada.

Figura 1. Elementos de uma SPN.

A Figura 2 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 o disparo 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 o seu disparo retorna o sistema ao estado de ligado.

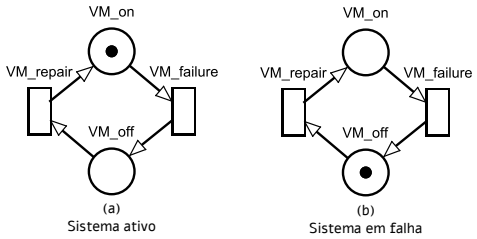

Figura 2. Exemplo de SPN.

2.2. Diagramas de Bloco de Confiabilidade

Os diagramas de bloco de confiabilidade (RBD) surgiram inicialmente como um modelo combinatório proposto como uma técnica para calcular a confiabilidade de um sistema. 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 et al. 2004].

A Figura 3(a) apresenta um modelo em RBD de um sistema em série, no qual o sistema só estará em funcionamento se todos os componentes estiverem ativos. A Figura 3(b) 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.

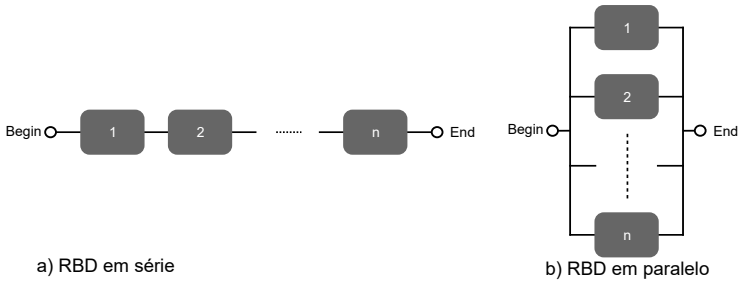

Figura 3. Exemplo RBD.

Para avaliar as métricas de dependabilidade, como confiabilidade ou disponibilidade, o RBD utiliza fórmulas matemáticas. O cálculo da disponibilidade para blocos conectados em série (*Rs*), considerando *n* componentes, é obtido por meio da Equação 1:

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

A disponibilidade do sistema considerando blocos conectados em paralelo (*Rp*), considerando n componentes, é obtida por meio do cálculo da Equação 2.

$$Rp = 1 - \prod_{i=1}^{n} (1 - R_i) \tag{2}$$

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.

2.3. Disponibilidade

A disponibilidade pode ser descrita como a probabilidade de que o sistema analisado esteja operacional durante um determinado período de tempo [Maciel et al. 2010]. A disponibilidade pode ser computada a partir da relação entre o somatório dos tempos de falha e dos tempos de reparo dos dispositivos que compõem o sistema, conforme mostrado na Equação 3. Nessa equação, MTTF (*Mean Time to Failure*) representa o tempo médio até a falha do sistema, e MTTR (*Mean Time to Repair*) representa o tempo médio para o reparo do sistema. Esse trabalho também computa a disponibilidade em termos do número de noves, $-log_{10}(1 - D)$.

$$D = \frac{MTTF}{MTTF + MTTR} \tag{3}$$

O índice de importância da disponibilidade (ID) [Figueirêdo et al. 2011] refere-se ao conceito que quantifica a contribuição relativa de cada componente individual para a disponibilidade global de um sistema. Nesse contexto, a compreensão do índice de

importância dos diferentes componentes permite direcionar os esforços de manutenção para os elementos mais críticos, promovendo, assim, uma melhoria geral na confiabilidade e na disponibilidade do sistema. Existem diferentes formas de aumentar a disponibilidade de um sistema, e uma das principais estratégias é a adição de redundâncias. Os dois tipos clássicos de redundância são o *hot standby* e o *cold standby*.

Na técnica de redundância *hot standby*, os equipamentos redundantes permanecem ativos e em sincronia com o equipamento principal. Eles participam do processo de operação da mesma forma que o componente principal e, em alguns casos, podem até compartilhar a carga de trabalho. Dessa forma, esses equipamentos podem assumir a carga do sistema caso o componente principal falhe, e vice-versa, sem um atraso na ativação [Johnson 1988]. A Figura 4(a) ilustra o funcionamento do *hot standby*, onde a falha e o reparo de ambas as máquinas virtuais (VMs) são independentes entre si.

Na redundância *cold standby*, os componentes redundantes não iniciam ativos, como ocorre com os componentes principais. Eles permanecem inativos, aguardando até que os componentes principais falhem, momento em que são ativados [Johnson 1988]. Nesse tipo de redundância, a ativação dos componentes redundantes ocorre após um certo período de tempo, denominado Mean Time To Activate (MTTA). A Figura 4(b) ilustra o funcionamento do *cold standby*. A transição MTTA possui uma expressão de guarda que garante que ela seja ativada apenas quando não houver token no lugar VM_principal_on, ou seja, quando a máquina virtual principal estiver em falha. Já a transição imediata T3 é ativada quando houver token no lugar VM_principal_on, indicando que a máquina virtual principal está funcionando.

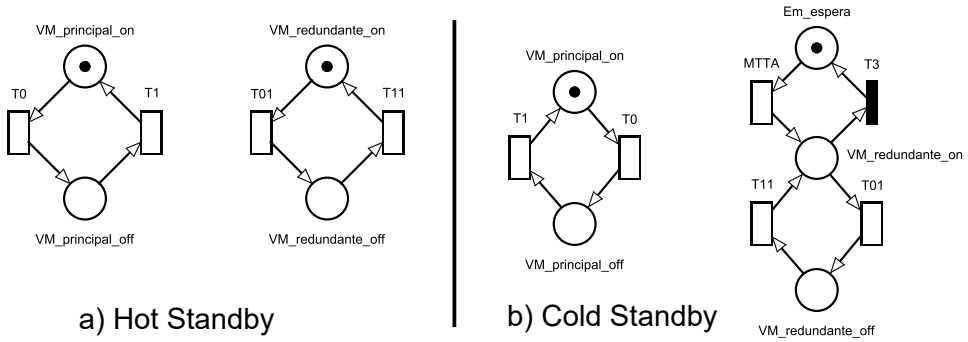

Figura 4. Tipos de Redundância.

3. Trabalhos Relacionados

Essa seção apresenta uma revisão dos trabalhos relacionados encontrados na literatura sobre a avaliação de disponibilidade de ambientes de computação em nuvem. Por exemplo, em [Araújo et al. 2021], os autores utilizam modelos formais baseados em SPN e RBD para avaliar a disponibilidade e confiabilidade de uma Computação em Nuvem Veicular. Análises de sensibilidade foram realizadas com o objetivo de identificar quais componentes do modelo têm maior impacto. Os autores também propõem uma extensão do modelo original para introduzir redundâncias. Em [Pereira et al. 2021], modelos analíticos de disponibilidade são propostos para a avaliação de ambientes de borda e névoa (*edge* e *fog*). 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. A

partir do modelo analítico proposto para planejar a infraestrutura, os autores mostraram que é possível aumentar a disponibilidade do sistema.

Em [Faraji Shoyari et al. 2021], os autores propõem uma modelagem hierárquica que combina RBD com CTMC para avaliar uma nuvem privada baseada no OpenStack. Dados a partir de um ambiente real foram coletados e utilizados nos modelos. Os resultados demonstram um impacto positivo na disponibilidade, 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.

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 que a utilização de redundâncias proporciona melhores resultados. [Maciel et al. 2022] apresentam uma pesquisa sobre a disponibilidade e confiabilidade de arquiteturas computacionais baseadas em nuvem, neblina (*fog*) e borda (*edge*). Os autores abordam as diferenças entre *edge* e *fog*, além de realizar uma análise do estado da arte. Foi identificado que 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. Os autores em [Silva and Callou 2025] avaliaram a disponibilidade de serviços ofertados por nuvem privada. Modelos em SPN e em RBD foram propostos. No entanto, o foco não foi o de identificar os componentes que mais impactam a disponibilidade.

A Tabela 1 apresenta uma comparação entre o presente trabalho e os estudos encontrados na literatura. Este trabalho utiliza um conjunto de indicadores, o qual engloba disponibilidade, *uptime* e *downtime*. Os estudos citados, como [Maciel et al. 2022], [Silva and Callou 2025] e [Faraji Shoyari et al. 2021], analisam a disponibilidade, mas nem todos focam em identificar os componentes de maior impacto no sistema. No que se refere às técnicas de modelagem, este trabalho utiliza RBD com SPN, o que possibilita a avaliação da confiabilidade estrutural da arquitetura com a capacidade de modelar algo além das limitações de um diagrama de blocos. Abordagens similares podem ser encontradas em estudos como [Araújo et al. 2021] e [Faraji Shoyari et al. 2021], embora aplicadas a cenários distintos. Assim, este trabalho se distingue dos demais ao realizar uma análise da disponibilidade do serviço Nextcloud hospedado em nuvem privada. Através do índice de Importância da Disponibilidade, identificam-se os componentes críticos do sistema e, assim, redundâncias são aplicadas.

4. Ambiente de Experimentos

A Figura 5 ilustra a arquitetura adotada para a realização dos experimentos que foi baseada em [Callou and Vieira 2024]. Essa arquitetura é composta por servidores, máquinas

Tabela 1. Trabalhos Relacionados

Artigos	Métricas	Modelagem	Aplicação de Redundâncias
[Araújo et al. 2021]	Disponibilidade, Confiabilidade	SPN, RBD	sim
[Pereira et al. 2021]	Disponibilidade	CTMC	não
[Faraji Shoyari et al. 2021]	Disponibilidade, Downtime, Custo	RBD, CTMC	não
[Pereira et al. 2022]	Disponibilidade	Arvore de Falha, CTMC	sim
[Clemente et al. 2022]	Disponibilidade	SPN	sim
[Maciel et al. 2022]	Disponibilidade, Confiabilidade	-	não
[Silva and Callou 2025]	Disponibilidade	SPN, RBD	sim
Este trabalho	Disponibilidade, Uptime, Downtime	SPN, RBD	sim

físicas Dell (Host1 e Host2), com a seguinte configuração: processador Intel Core i5-10400F de 10ª geração, 8 GB de memória RAM, 500 GB de armazenamento e sistema operacional CentOS 7. Além disso, a arquitetura conta com uma máquina física HP, utilizada como cliente, e que tem a seguinte configuração: processador AMD A8-550B 3.2 GHz, 8 GB de memória RAM, 500 GB de armazenamento e sistema operacional Windows 10. Nessa máquina foi instalado o Apache JMeter[1], uma ferramenta de geração de carga que é responsável por enviar requisições ao servidor, simulando o comportamento de múltiplos usuários.

O Host1 é configurado como o host principal. Nesse host estão hospedados o controlador do CloudStack [2] que é o responsável pelo gerenciamento de recursos do sistema, e suas respectivas máquinas virtuais (VMs) de sistema (a SSVM, Secondary Storage VM, e a CPVM, Console Proxy VM). Sendo responsáveis a CPVM pelo acesso ao console das instâncias via web e a SSVM pelo armazenamento de templates e ISOs de sistemas operacionais. A infraestrutura de rede é formada por um switch, que interconecta os hosts do servidor, a máquina cliente e a internet. O switch garante a comunicação entre os diferentes componentes do ambiente e permite a geração de tráfego entre o cliente e o servidor.

Três máquinas virtuais (VMs) foram criadas para simular um serviço em nuvem, todas com as mesmas especificações de hardware: 1 core de 0,5 GHz, 512 MB de memória RAM e 20 GB de armazenamento. Dentro de duas VMs (VM1 e VM2), foi instalado o Nextcloud[3], um servidor de arquivos amplamente utilizado em ambientes de nuvem privada. Na terceira VM (LB), foi instalado o nginx[4], configurado para atuar como balanceador de carga. É importante destacar que as VM1 e VM2 são máquinas virtuais redundantes, ambas hospedando o serviço Nextcloud. Elas são as responsáveis por oferecerem a funcionalidade do servidor de arquivos. A VM LB foi configurada com nginx, e distribui as requisições dos clientes de maneira proporcional entre as duas outras VMs com o serviço Nextcloud. Essa configuração contribui para melhorar a escalabilidade e a disponibilidade do serviço, pois caso uma das VMs falhe, a outra pode continuar atendendo às requisições.

5. Estudo de Caso

Este estudo tem como objetivo analisar o sistema e identificar o nível de impacto de cada componente na disponibilidade total do sistema. Para alcançar tal objetivo é necessária a

[1]https://jmeter.apache.org/
[2]https://cloudstack.apache.org/
[3]https://nextcloud.com/
[4]https://nginx.org/

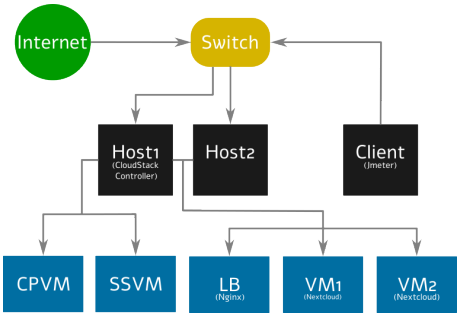

Figura 5. Arquitetura do Ambiente de Testes.

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 a presente pesquisa.

Como os experimentos deste estudo demandam o uso do RBD, é fundamental representar o sistema por meio desse formalismo. Portanto, o primeiro passo consiste na modelagem do sistema utilizando essa abordagem. A ferramenta Mercury [Silva et al. 2015] foi utilizada para criar e avaliar os modelos via análise estacionária. O sistema em questão refere-se ao ambiente de experimentos ilustrado na Figura 5. Em seguida, será 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 de maior impacto na disponbilidade do sistema, (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 pode ser repetido até que se consiga obter a disponibilidade desejada ou até se atingir uma quota limite de redundâncias definida pelo projetista. 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.

Para demonstrar os benefícios da aplicação de redundâncias, aos componentes mais críticos,para a disponibilidade do sistema, o presente estudo faz uso de um conjunto de experimenos. Em cada um desses experimentos teremos a arquitetura representada a partir de modelos em RBD que vão representar as variações do ambiente ilustrado na Figura 5. Esses modelos fazem uso dos valores de MTTF (*tempo médio para a falha*) e MTTR (*tempo médio para o reparo*) apresentados na Tabela 2. Esses dados foram obtidos a partir de [Martins Maciel 2016] e [Silva and Callou 2025].

Tabela 2. Valores Tempo de Falha e Reparo.

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

Tabela 3. 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	**0,9861991091741**	**2,6009287966512**

5.1. Cálculos da Importância da Disponibilidade

O experimento *e0* parte da arquitetura base apresentada na Figura 6 (a). Essa arquitetura considera o ambiente (Figura 5) com uma única máquina virtual (VM) fornecendo o serviço. A Tabela 3 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 6 (b). De acordo com o resultado obtido, o componente VM1 foi avaliado como o dispositivo que mais impacta a disponibilidade do sistema. 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.

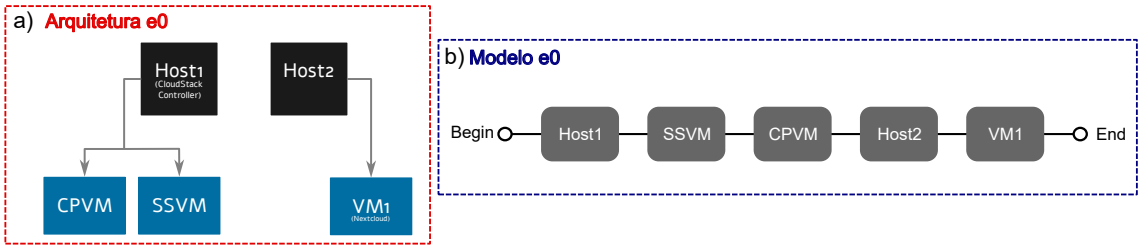

Figura 6. Experimento *e0*.

Seguindo a recomendação de redundância indicada pelo resultado anterior, o experimento *e1* considera uma nova arquitetura levando em consideração uma nova VM (VM2) redundante. A Figura 7 (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 7 (b). A Tabela 4 apresenta os resultados obtidos da avalição do índice ID para o experimento *e1*. Nesses resultados, 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.

O experimento *e2* apresenta uma nova arquitetura conforme mostrada na Figura 8 (a). Essa arquitetura *e2* inclui uma redundância para o balanceador de carga, mas como

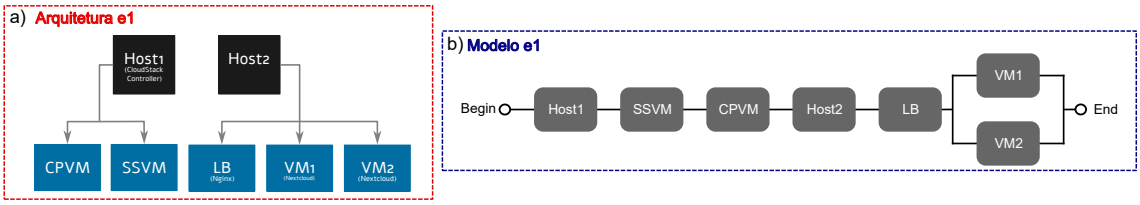

Figura 7. Experimento *e1*.

Tabela 4. 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

não faz sentido haver dois balanceadores de carga ativos simultaneamente, o LB redundante estará configurado em *Cold Standby*. Assim, o redundante será ativado apenas 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 8 (b). Dessa forma, foi proposto um modelo em SPN que inclui o LB e sua redundância em *Cold Standby* conforme mostrado na Figura 8 (c). Esse modelo usa como parâmetros de entrada os tempos de falha e reparo presentes na Tabela 2 e a transição *enable_redundant_LB* usa 0,05h. 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 5 apresenta o impacto de cada componente na arquitetura e2. 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 a métrica de importância da disponibilidade (AvI). Assim, a falha do Host2 passa a ter o maior impacto na disponibilidade do sistema.

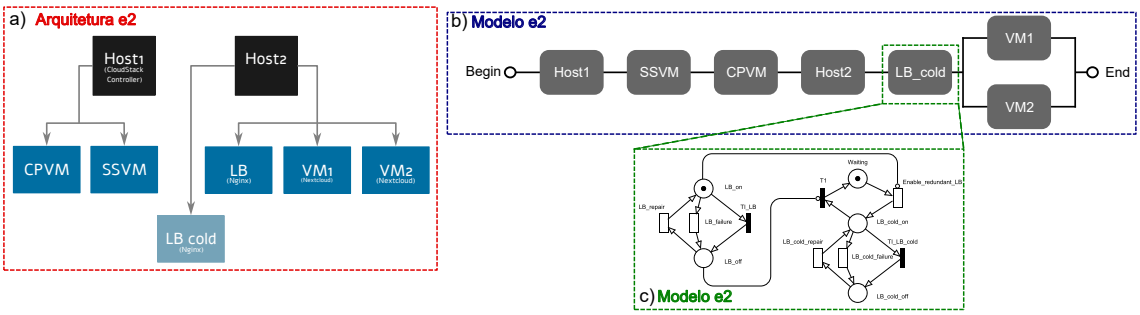

Figura 8. Experimento *e2*.

Tabela 5. 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

Tabela 6. 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

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 9 (a), e seu modelo RBD correspondente é mostrado na Figura 9 (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 6 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.

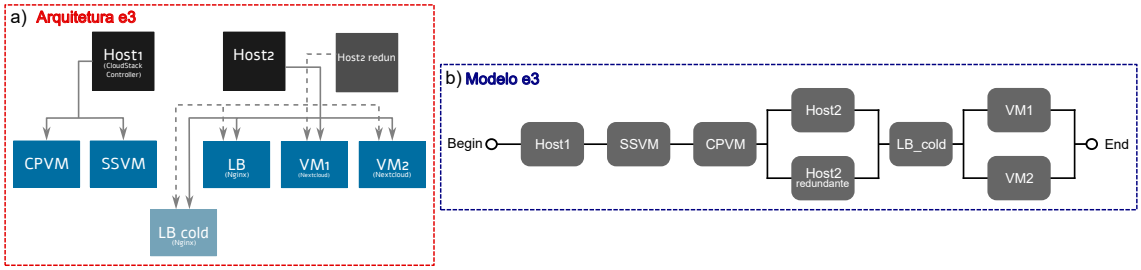

Figura 9. Experimento *e3*.

O Experimento *e4* propõe a implementação de uma redundância para o Host1. A Figura 10 (a) ilustra a arquitetura *e4* que inclui uma redundância para o componente Host1. Esse novo componente denominado Host1_redun é configurado em *Hot Standby* em relação ao componente Host1. A Figura 10 (b) apresenta o modelo correspondente e a Tabela 7 apresenta os resultados obtidos para os índices ID nesse sistema. Esses resultados demonstram 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 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

Tabela 7. 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

Tabela 8. 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

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.

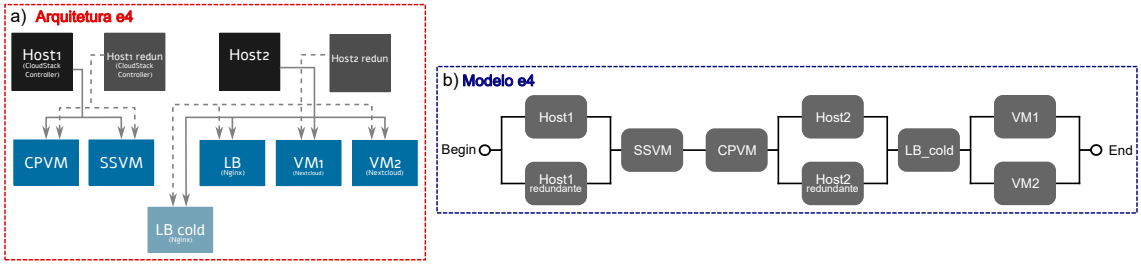

Figura 10. Experimento *e4*.

5.2. Resultados das Arquiteturas Propostas

A Tabela 8 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. A partir 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, houve uma leve queda na disponibilidade ao estender o Experimento *e0* para o *e1*. Esse decréscimo é explicado pelo fato de que a adição de um novo componente em série (LB) ao sistema 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 11 ilustra as melhorias na disponibilidade alcançadas com a aplicação das redundâncias nos experimentos. Tomando o experimento *e1* como valor de referência, a Figura 11 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.

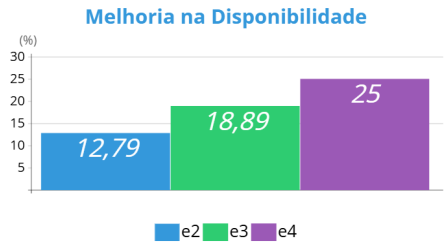

Figura 11. Porcentagem de melhoria em relação a e1.

6. Conclusão

Este trabalho propôs modelos em RBD e SPN para avaliar a disponibilidade de um serviço Nextcloud hospedado em um ambiente de nuvem privada Apache CloudStack. A abordagem adotada permitiu realizar uma análise detalhada da arquitetura do sistema, identificando e analisando os componentes que mais impactam a disponibilidade do serviço. Adicionalmente, o trabalho propôs melhorias, adicionando redundâncias nos componentes críticos para aumentar a confiabilidade e reduzir o downtime, melhorando, assim, a disponibilidade geral do sistema. A implementação de redundâncias garantiu que o serviço se mantivesse operacional mesmo diante de falhas dos componentes de maior impacto na disponibilidade do sistema. Como trabalhos futuros, é proposto realizar uma análise comparativa entre o ambiente de experimentos estudado neste trabalho e as melhorias propostas sob a perspectiva de custos e outros impactos (como a eficiência). A análise incluirá uma avaliação detalhada do consumo elétrico associado às operações do sistema, permitindo entender melhor o impacto das melhorias de confiabilidade tanto no desempenho quanto nos custos operacionais relacionados ao consumo de energia.

Referências

Araújo, G., Rodrigues, L., Oliveira, K., Fé, I., Khan, R., and Silva, F. A. (2021). Vehicular cloud computing networks: Availability modelling and sensitivity analysis. *International Journal of Sensor Networks*, 36(3):125–138.

Asio, J. M. R., Gadia, E., Abarintos, E., Paguio, D., and Balce, M. (2021). Internet connection and learning device availability of college students: Basis for institutionalizing flexible learning in the new normal. *Studies in Humanities and Education*, 2(1):56–69.

Bauer, E. and Adams, R. (2012). *Reliability and availability of cloud computing*. John Wiley & Sons.

Callou, G. and Vieira, M. (2024). Availability and performance analysis of cloud services. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, LADC '24, page 262–271, New York, NY, USA. Association for Computing Machinery.

Clemente, D., Pereira, P., Dantas, J., and Maciel, P. (2022). Availability evaluation of system service hosted in private cloud computing through hierarchical modeling process. *The Journal of Supercomputing*, 78(7):9985–10024.

Cloud, H. (2011). The nist definition of cloud computing. *National institute of science and technology, special publication*, 800(2011):145.

Faraji Shoyari, M., Ataie, E., Entezari-Maleki, R., and Movaghar, A. (2021). Availability modeling in redundant openstack private clouds. *Software: Practice and Experience*, 51(6):1218–1241.

Figueirêdo, J., Maciel, P., Callou, G., Tavares, E., Sousa, E., and Silva, B. (2011). Estimating reliability importance and total cost of acquisition for data center power infrastructures. In *2011 IEEE international conference on systems, man, and cybernetics*, pages 421–426. IEEE.

Haas, P. J. (2006). *Stochastic petri nets: Modelling, stability, simulation*. Springer Science & Business Media.

Johnson, B. W. (1988). *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc.

Maciel, P., Dantas, J., Melo, C., Pereira, P., Oliveira, F., Araujo, J., and Matos, R. (2022). A survey on reliability and availability modeling of edge, fog, and cloud computing. *Journal of Reliable Intelligent Environments*, pages 1–19.

Maciel, P., Trivedi, K., and Kim, D. (2010). Dependability modeling in: Performance and dependability in service computing: Concepts, techniques and research directions. *Hershey: IGI Global, Pennsylvania, USA*, 13:1380.

Martins Maciel, P. R. (2016). Modeling availability impact in cloud computing. *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*, pages 287–320.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Pereira, P., Araujo, J., Melo, C., Santos, V., and Maciel, P. (2021). Analytical models for availability evaluation of edge and fog computing nodes. *The Journal of Supercomputing*, 77:9905–9933.

Pereira, P., Melo, C., Araujo, J., Dantas, J., Santos, V., and Maciel, P. (2022). Availability model for edge-fog-cloud continuum: an evaluation of an end-to-end infrastructure of intelligent traffic management service. *The Journal of Supercomputing*, pages 1–28.

Silva, A. and Callou, G. (2025). Models for availability evaluation of file servers in private clouds. *Computing*, 107(1):1–27.

Silva, B., Matos, R., Callou, G., Figueiredo, J., Oliveira, D., Ferreira, J., Dantas, J., Lobo Júnior, A., Alves, V., and Maciel, P. (2015). Mercury: An integrated environment for performance and dependability evaluation of general systems.

Xie, M., Dai, Y.-S., and Poh, K.-L. (2004). *Computing system reliability: models and analysis*. Springer Science & Business Media.