Modelagem Hierárquica e Heterogênea para Avaliação de Disponibilidade de Aplicações Big Data na Nuvem Privada

A. C. Cunha[1], **G. R. A. Callou**[1], **E. T. G. Sousa**[1], **E. A. G. Tavares**[2]

[1]Departamento de Computação – Universidade Federal Rural de Pernambuco (UFRPE)
Recife, PE – Brazil

[2]CIn - Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife, PE – Brazil

{alex.cunha,gustavo.callou,erica.sousa}@ufrpe.br, eagt@cin.ufpe.br

Abstract. *The growing development of cloud services and an exponential increase in the volume of data produced has made Hadoop the standard platform for managing and processing data in Big Data environments. Ensuring the high availability of cloud services is an essential criterion. However, it becomes a complicated task, given the number of components in the system. Therefore, we propose a modeling strategy based on stochastic Petri nets (SPN) and reliability block diagrams (RBD) to assess the availability and reliability of cloud services, considering different redundant mechanisms. This article adopts sensitivity analysis to evaluate the component that most affects the availability of the Hadoop cluster configured on the OpenStack cloud platform.*

Resumo. *O crescente desenvolvimento de serviços em nuvem e um aumento exponencial no volume de dados produzidos fez do Hadoop a plataforma padrão para gerenciamento e processamento de dados em ambientes de Big Data. Garantir a alta disponibilidade dos serviços em nuvem é um critério essencial. No entanto, torna-se uma tarefa complicada, dado o número de componentes do sistema. Portanto, propomos uma estratégia de modelagem baseada em redes de Petri estocásticas (SPN) e diagramas de blocos de confiabilidade (RBD) para avaliar a disponibilidade e confiabilidade dos serviços em nuvem, considerando diferentes mecanismos redundantes. Este artigo adota análise de sensibilidade para avaliar o componente que mais afeta a disponibilidade do cluster Hadoop configurado na plataforma de nuvem OpenStack.*

1. Introdução

Com o crescimento exponencial dos dados produzidos pelo uso massivo da Internet, o processamento de *Big Data* passou a ter um protagonismo significativo, despertando o interesse cada vez maior no meio acadêmico e corporativo quanto ao estudo e desenvolvimento de aplicações. *Big Data* é um termo que descreve a grande quantidade de dados gerada no mundo em rede, digitalizado, com diversos sensores e orientado por infomações[Chang and Grady 2019].

Dentre as tecnologias de *Big Data* voltadas para processamento de dados, o *Apache Hadoop* é a mais popular devido sua simplicidade, econômia, alta escalabilidade e

tolerância a falhas [Bhathal and Dhiman 2018]. A computação em nuvem é um modelo que permite acesso onipresente, conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou intervenção do provedor de serviços[Mell and Grance 2011].

Esse trabalho propõe um conjunto de modelos que permita avaliar o impacto na disponibilidade dos serviços *Hadoop cluster* considerando infraestruturas de nuvem privada. A abordagem proposta leva em consideração uma estratégia de modelagem hierárquica e heterogênea baseada em formalismos matemáticos, diagrama de blocos de confiabilidade e redes de Petri estocásticas. Este trabalho entende que a modelagem pode mitigar problemas relacionados à disponibilidade do serviços na nuvem privada antes da sua implementação de modo a aplicar estratégias de redundância a componentes de maior impacto no sistema.

O artigo está organizado de acordo com as seções a seguir. A Seção II apresenta os trabalhos relacionados. A Seção III apresenta os conceitos básicos. A Seção IV apresenta a estratégia de modelagem hierárquica e heterogênea para avaliação de ambientes *Big D*ata na nuvem privada. A Seção V mostra os modelos RBD e SPN adotados para avaliação de disponibilidade de aplicações *Big Data* na nuvem privada. A Seção VI mostra o estudo de caso proposto e na Seção VII são apresentadas as conclusões do trabalho.

2. Trabalhos Relacionados

[Melo et al. 2017] propõem um modelo para avaliar a capacidade do nó em um ambiente de computação em nuvem com base na quantidade de recursos de *hardware* diponíveis. O estudo combinou modelos para avaliação da disponibilidade, como diagramas de blocos de confiabilidade (RBD) e redes de Petri estocásticas para avaliar a disponibilidade orientada à capacidade. Com isso pode-se determinar a quantidade real de recursos da nuvem privada disponíveis em um intervalo de tempo predeterminado. A análise de sensibilidade foi adotada para determinar o componente com maior impacto na métrica de intereresse. A análise da capacidade orientada à disponibilidade constatou uma degradação no provimento de VM's no nó, devido a falhas e respectivos períodos de reparação.

Em [Marynowski et al. 2015], os autores propuseram uma abordagem para testar a tolerância a falhas no MapReduce com base na geração e execução de casos de falha representativos. Os casos de falha derivam do gráfico de alcançabilidade de uma rede de Petri (PN) que modela o mecanismo de tolerância a falhas. O modelo proposto representa o processo para iniciar o *master* e os *workers*, iniciar um aplicativo e interromper os componentes posteriormente; o tratamento de falhas que ocorrem enquanto um *worker* executa um map ou tarefa de redução ou quando todos os *workers* falham, causando a falha do aplicativo. A ferramenta *HadoopTest* foi selecionada para automatizar a execução de casos de falha no ambiente distribuído, controlar a execução de diferentes componentes *MapReduce* em várias máquinas, garantir que as falhas sejam injetadas nos componentes em circunstâncias corretas, monitorar o comportamento do sistema e validar se ele se comportou conforme o esperado. Os resultados obtidos mostraram que os casos de falha foram representativos para testar o *Hadoop*, permitindo identificar alguns erros, confiabilidade da rede.

Porém, nenhum dos artigos propôs uma abordagem para avaliação de disponibli-

dade de serviços de Big Data na nuvem privada.

3. Conceitos Básicos

1. ***Apache Hadoop*** é uma plataforma usada para suportar o processamento de grandes conjuntos de dados em um ambiente de computação distribuído. A estrutura do *framework* é composta pelos módulos: *Hadoop Common*, *Hadoop Distributed File System (HDFS)*, *Hadoop Yarn* e *Hadoop MapReduce*. A estrutura permite que a especificação de uma operação seja aplicada a um grande conjunto de dados, dividindo o problema e os dados e executando-os em paralelo. As duas funções no *MapReduce* são mapeamento e redução. O *Mapeamento* usa pares de chave/valor como entrada e gera um conjunto intermediário de pares de chave/valor. A *Redução* mescla todos os valores intermediários associados à mesma chave intermediária [Bhosale et al. 2018].
2. ***OpenStack*** é uma plataforma de *software* livre que usa recursos virtuais agrupados para criar e gerenciar nuvens públicas e privadas. A plataforma permite que usuários implantem máquinas virtuais e outras instâncias que lidem com diferentes tarefas para gerenciar um ambiente de nuvem em tempo real[Liberman 2015]. A plataforma *OpenStack* é composta por vários componentes, os principais são: Nova, Swift, Cinder, Neutron, Horizon, Keystone, Glace e Ceilometer[Liberman 2015].
3. ***Confiabilidade:*** a **disponibilidade** de um sistema é sua capacidade em manter-se ativo e eficaz por um determinado intervalo de tempo. [Trivedi et al. 2012] definem que a disponibilidade é utilizada para estimar a proporção de tempo que o sistema está ativo durante um determinado período. A disponibilidade pode ser obtida através da Equação 1.

$$disponibilidade = \frac{tempoAtivo}{tempoAtivo + tempoInativo} \tag{1}$$

4. ***Diagrama de Bloco de Confiabilidade:*** Os diagramas de bloco de confiabilidade (RBD), são modelos combinatórios usados para calcular a disponibilidade e confiabilidade dos sistemas [O'Connor 1998].
   De acordo com [Florin et al. 1991], os diagramas de bloco de confiabilidade definem os relacionamentos lógico entre os componentes de um sistema. Os relacionamentos típicos são: série (Figura 1(a)) e paralelo (Figura 1(b)) ou combinação dessas estruturas.

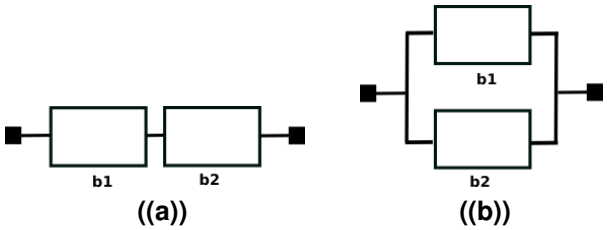

Figura 1. Diagramas de Bloco de Confiabilidade

A disponibilidade e confiabilidade de dois blocos conectados em série é obtida através do produto das disponibilidades de cada um dos *n* blocos componentes. Sendo assim, a disponibilidade pode ser calculada a partir da Equação 2, onde, P_s

(*Availability*) corresponde ao valor da disponibilidade do sistema, $P_i(t)$ descreve a confiabilidade $R_i(t)$, a disponibilidade instantânea $A_i(t)$ e a disponibilidade de estado estacionário A_i do bloco B_i.
A disponibilidade e a confiabilidade de dois blocos conectados em paralelo é obtida através da Equação 3.

$$P_s = \prod_{i=1}^{n} P_i(t) \qquad (2) \qquad P_s = 1 - \prod_{i=1}^{n} (1 - P_i(t)) \qquad (3)$$

5. ***Redes de Petri Estocásticas:*** Redes de Petri estocásticas (SPN) são uma extensão das redes de Petri, onde o conceito de tempo é adicionado permitindo a avaliação de desempenho e dependabilidade[Trivedi 2016].
A Figura 2 apresenta um exemplo de modelo SPN básico, os círculos representam lugares, as transições são representadas por retângulos. Os arcos conectam os lugares às transições (ou transições aos lugares). Os pequenos círculos (tokens) presentes no local definem o estado inicial do modelo SPN.

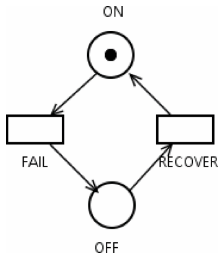

Figura 2. Modelo básico SPN

4. Estratégia de Modelagem Hierárquica e Heterogênea para Ambientes de *Big Data* na Nuvem Privada

Está seção mostra a estratégia de modelagem baseada em diagramas de bloco de confiabilidade (RBD) para capturar as condições que provocam falhas no sistema ou propiciam o seu funcionamento quando são consideradas as relações estruturais dos seus componentes e redes de Petri estocástica (SPN) que representam as relações de dependência entre os componentes dos sistemas para avaliar a disponibilidade de ambientes *Big Data* na nuvem privada.

A adoção da abordagem hieráquica reduz de forma significativa a complexidade para representar grandes sistemas, tornando-se mais adequado para representar os subsistemas e combinar os resultados para obter o resultado da avaliação do modelo do sistema.

Figura 3 mostra um modelo RBD de um sistema computacional que compõem a Plataforma OpenStack (processador(PR), memória principal (MP) e memória sencundária (MS)). Esses recursos são agrupados de forma a gerarem um componente. Após obter MTTF e MTTR, um modelo basedo em SPN ou RBD é gerado para representar o componente da plataforma OpenStack em cold standby.

Modelos baseados diagramas de blocos de confiabilidade e redes de Petri estocástica são concebidos para representação dos componentes, subsistemas e mecanimos de redundância da infraestrutura de nuvem. Desta forma, o Modelo do Sistema Computacional foi concebido para representar a infraestrutura da nuvem. Os MTTFs e MTTRs dos componentes da infraestrutura de nuvem são obtidos através da Tabela 1. Os componentes

são combinados para obtenção dos MTTFs e dos MTTRs dos subsistemas. O modelo do *Compute Node*, o modelo do *Controller Node*, o modelo do *Network Node* e o modelo do *Cloud Controller* e o modelo do *Storage Node* foram concebidos para representação dos subsistemas da infraestrutura de nuvem. Os modelos dos componentes e subsistemas são baseados em diagramas de blocos de confiabilidade (RBD). Posteriormente, um modelo de sistema é construído compondo os modelos de subsistema. Este modelo de sistema é baseado em redes de Petri estocásticas (SPN) ou diagramas de blocos de confiabilidade (RBD), dependendo dos mecanismos de redundância que serão atribuídos ao sistema. Se o mecanismo de redundância *Hot standby* for atribuído a infraestrutura de nuvem, o modelo do sistema de nuvem será baseado em diagramas de blocos de confiabilidade (RBD). Quando os mecanimos de redundância ativo-ativo e *cold standby* forem adotados, o modelo do sistema de nuvem será baseado em redes de Petri estocásticas (SPN). Desta forma, o modelo do sistema de nuvem será modelo com SPN quando houver a necessidade de representar a dependência entre os subsistemas e mecanismos de redundância.

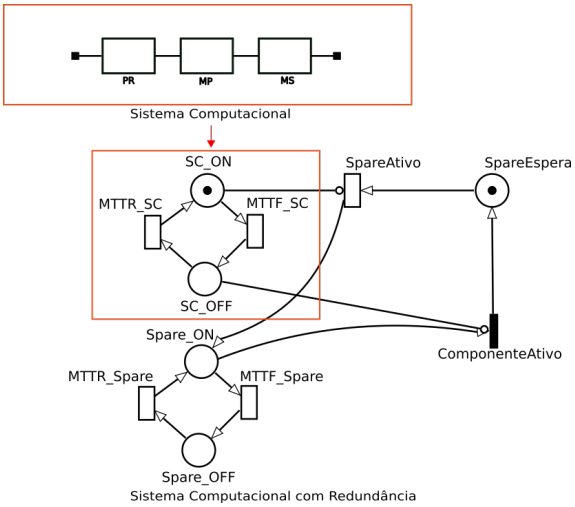

Figura 3. Sistema Computacional com Mecanismo de Redundância *Cold Standby*

5. Modelos

Esta seção apresenta os modelos RBD e SPN adotados para avaliação de serviços *Big Data* na nuvem privada. Os modelos propostos são: sistema computacional, compute node, network node, cloud controller, storage node e nuvem privada. Os modelos de redundância apresentados são: ativo-ativo, *hot standby* e *cold standby*.

1. ***SC***: o modelo RBD do Sistema Computacional está representado na Figura 4, e representa infraestrutura de processamento (PR), a memória principal (MP) e memória secundária (MS) da máquina física. Esse modelo RBD é adotado para estimar o MTTF o MTTR do sistema computacional. O modo operacional desse modelo é $MO_{SC} = (PR \wedge MP \wedge MS)$.

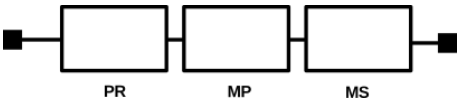

Figura 4. Modelo do Sistema Computacional

2. ***CN***: o *Compute Node* refere-se a um servidor *OpenStack* que executa um hipervisor KVM. É responsável por executar instâncias de máquinas virtuais. O modelo é composto pelo Sistema Computacional (SC), Sistema Operacional (SO), *Hypervisor*, *Neutron-openvswitch-agent* e *Openstack-nova-compute*. A Figura 5 apresenta modelo do Compute Node. O modo operacional desse modelo é $MO_{CN} = (SC \wedge SO \wedge HYPERVISOR \wedge NEUTRON - OPENVSWITCH - AGENT \wedge OPENSTACK - NOVA - COMPUTE)$.

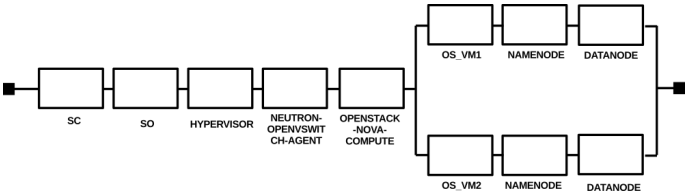

Figura 5. Modelo do Compute Node

3. ***NN***: o *Network Node* provê controle centralizado para todos os *Compute Nodes*. Ele executa os vários agentes *Neutron* que controlam as funcionalidades da rede nível 3 no *Cluster*. O modelo é composto pelos componentes Sistema Computacional (SC), Sistema Operacional (SO), *Neutron-dhcp-agent*, *Neutron-l3-agent*, *Neutron-metadata-agent* e *Neutron-openvswitch-agent*. A Figura 6 apresenta esse modelo. O modo operacional desse modelo é $MO_{NN} = (SC \wedge SO \wedge NEUTRON - DHCP - AGENT \wedge NEUTRON - L3 - AGENT \wedge NEUTRON - METADATA - AGENT \wedge NEUTRON - OPENVSWITCH - AGENT)$.

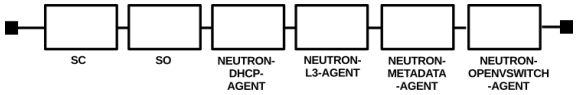

Figura 6. Modelo Network Node

4. ***SN***: *Storage Nodes* são servidores de armazenamento que podem armazenar, recuperar e excluir objetos armazenados em dispositivos locais. Nesta arquitetura de referência, há três servidores de armazenamento *Swift* dedicados. O modelo é composto pelo Sistema Computacional (SC), Sistema Operacional(SO), *swift-account*, *swift-container*, *swift-object*. A Figura 7 apresenta o modelo *Storage Nodes*. O modo operacional desse modelo é $MO_{SN} = (SC \wedge SO \wedge SWIFT - ACCOUNT \wedge SWIFT - CONTAINER \wedge SWIFT - OBJECT)$.

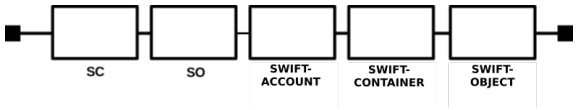

Figura 7. Storage Nodes

5. ***CC***: o *Cloud Controller* é o componente responsável pelo gerenciamento da nuvem, mantendo uma visão de alto nível dos recursos para consultas de *API* baseadas em *REST* para a maioria dos serviços *OpenStack*. O modelo *Cloud Controller* é composto pelos componentes Sistema Computacional (SC), Sistema Operacional (SO), *rabbitMQ*, *Neutron-server*, *cinder-api*, *cinder-schedule*, *cinderr-volume*, *glance-api*, *glance-registry*, *heat-api-cfn*, *heat-api*, *heat-engine*, *keystone*, *nova-api*, *nova-cert*, *nova-conductor*, *nova-consoleauth*, *nova-novncproxy*, *nova-scheduler*, *sahara-all* e *swift-proxy*. A Figura 8 apresenta esse modelo. O modo

operacional desse modelo é $MO_{CC} = (SC \wedge SO \wedge rabbitMQ \wedge NEUTRON - SERVER \wedge CINDER - API \wedge CINDER - SCHEDULER \wedge CINDER - VOLUME \wedge GLACE - API \wedge GLACE - REGISTRY \wedge HEAT - API - CFN \wedge HEAT - API \wedge HEAT - ENGINE \wedge KEYSTONE \wedge NOVA - API \wedge NOVA - CERT \wedge NOVA - CONDUCTOR \wedge NOVA - CONSOLEAUTH \wedge NOVA - NOVNCPROXY \wedge NOVA - SCHEDULER \wedge SAHARA - ALL \wedge SWIFT - PROXY)$.

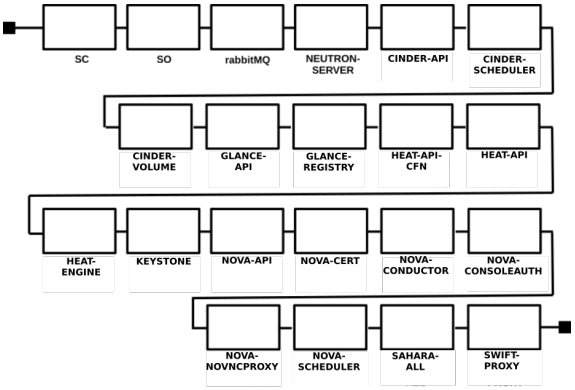

Figura 8. Modelo Cloud Controller

6. ***Modelo da Plataforma Nuvem Privada***: os componentes da plataforma *OpenStack* podem ser descritos como *Cloud Controller* (CC), *Compute Node* (CN), *Network Node* (NN), *Storage Node* (SN), *switch* (SW) e roteador (RT). A Figura 9 apresenta o modelo RBD adotado para estimar a diponibilidade da infraestrutura da plataforma *OpenStack Sahara*. O modo operacional desse modelo é $MO_{NP} = (CC \wedge CN \wedge NN \wedge SN \wedge SW \wedge RT)$.

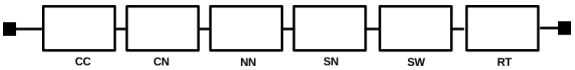

Figura 9. Modelo RBD da nuvem privada

5.1. Modelos de Redundância

5.1.1. Ativo-Ativo

Os mecanismos de redundância do tipo ativo-ativo são empregados quando os componentes primários e secundários atendem às requisições dos usuários do sistema. Na ocorrência de falha de um dos componentes, o componente redundante será o responsável por atender às requisições dos usuários do sistema. Portanto, classificamos este mecanismos de redundância como **N+K**, onde **K** componentes secundários idênticos aos **N** componentes primários, compartilham a carga de trabalho do sistema[Bauer et al. 2012]. O modo operacional do Modelo de Redundância Ativo-Ativo proposto é $MO_{AA} = (CN1\_ON = 0 \vee CN2\_ON = 0)$. A Figura 10 mostra o modelo SPN adotado para estimar a disponibidade do sistema com redundância ativo-ativo. As marcações dos lugares CN1_ON e CN2_ON denotam os estados operacionais do CC, CN, NN, SN, SW, RT, Componente Principal e Componente Redundante, respectivamente, e as marcações nos lugares CN1_OFF e CN2_OFF denotam os estados defeituosos desses componentes.

Quando o componente principal falha, o componente redundante assume a carga de trabalho do sistema. As transições temporizadas MTTF_CN1, MTTF_CN2,

MTTR_CN1 e MTTR_CN2 representam a ocorrência de um evento de falha e de uma atividade de reparo nos componentes principal e redundante. Os tempos associados a essas transições temporizadas representam o MTTF e o MTTR desses componentes. Esse modelo também representa a probabilidade da detecção e da não detecção do evento de falha através de pesos associados as transições imediatas CN1_Detectado (0,8), CN2_Detectado (0,8), CN1_NDetectado (0,2) e CN2_Ndetectado (0,2). Após a ocorrência do evento de falha com os disparos das transições temporizadas MTTF_CN1 e MTTF_CN2, esse evento de falha pode ser detectado com os disparos das transições imediatas CN1_Detectado e CN2_Detectado e pode não ser detectado com os disparos das transições imediatas CN1_NDetectado e CN2_Ndetectado. Os lugares CN1_OFFCoberto e CN2_OFFCoberto denotam a detecção do evento de falha e os lugares CN1_OFFNCoberto e CN2_OFFNCoberto representam a não detecção do evento de falha.

Os componentes principal e redundante compartilham a carga de trabalho do sistema. Quando há a detecção de um evento de falha no componente principal, o sistema pode enviar a carga de trabalho para o componente redundante.

A detecção do defeito e envio da carga de trabalho para o componente redundante permite o reparo do sistema após o disparo das transições temporizadas MTTR_CN1 e MTTR_CN2. Quando o defeito não é detectado, há um erro de percepção, pois para o sistema não ocorreu nenhum defeito. As transições temporizadas ErroPercepcao_CN1 e ErroPercepcao_CN2 representam esse erro de percepção e os tempos associados a essas transições representam o período de duração desse erro de percepção. Após esse período de tempo, o defeito é percebido [Sousa 2015].

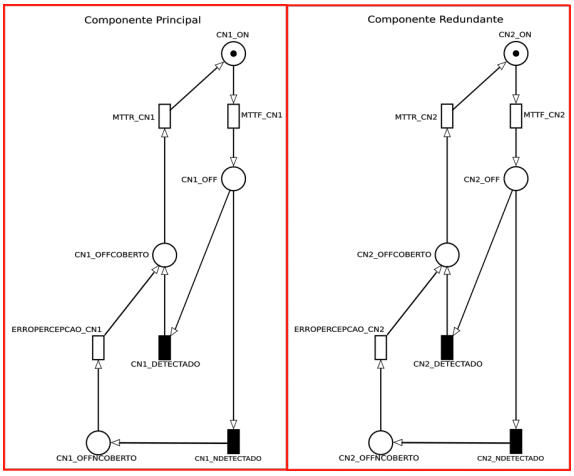

Figura 10. Modelo Ativo-Ativo

5.1.2. Hot Standby

A redundância *hot standby* mantém um módulo redundante sempre ativo, dessa forma, ao surgir uma ocorrência de falha no módulo principal, esse pode ser substituído pelo módulo redundante sem um atraso.

A Figura 11 mostra o modelo *hot standby* aplicado ao *Cloud Controller*. O componente (CC_MAIN) estão operacionais e um mecanismo de re-

dundância (CC_STANDBY). O modo operacional do modelo *hot standby* é $MO_{HS} = (CC\_MAIN \vee CC\_STANDBY)$.

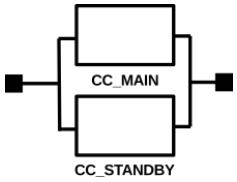

Figura 11. Hot Standby

5.1.3. Cold Standby

Em uma redundância *cold standby* o componente de redundância não está ativo. Ele espera para ser ativado quando o módulo principal falhar. Quando o múdulo principal falha, a ativação do módulo redundante ocorre em um certo período de tempo que é denominado de *Mean Time To Activate* (MTTA). Indicando que o ajuste por vezes pode ser manual[Sousa 2015][Bauer et al. 2012].

O modelo *cold stadby* é representado na Figura 12. O modo operacional do modelo Cold Standby é $MO_{CS} = (Componente\_On = 0 \vee Spare\_On = 0)$.

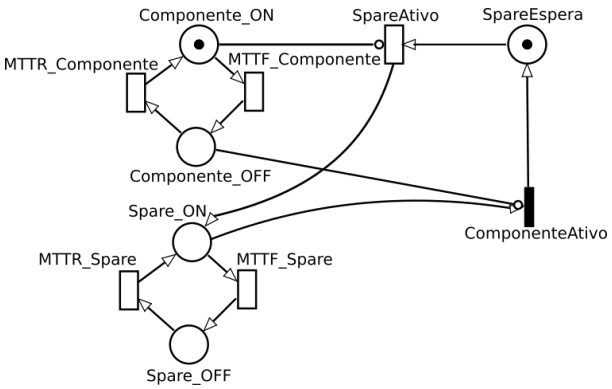

Figura 12. Modelo *Cold Standby*

6. Estudo de Caso

Esta seção apresenta um estudo de caso para ilustrar a aplicabilidade da modelagem hierárquica e heterogênea proposta para avaliar a disponibilidade de aplicações *Big Data* na nuvem privada.

O estudo de caso é adotada a análise de sensibilidade para identificar o componente do *Hadoop cluster* na plataforma *OpenStack* que mais afeta a disponibilidade do sistema. Esse componente terá a atribuição das redundâncias ativo-ativo, *hot standby* e *cold Standby* para avaliação do impacto das redundâncias na disponibilidade do sistema.

6.1. Configuração do Ambiente Modelado

O ambiente considerado para este estudo de caso, consiste em quatro máquinas sendo: Cloud Controller (CC), Compute Node (CN) responsável por controlar os recursos físicos e fornecer às máquinas virtuais rodando no hipervisor VirtualBox, Network Node (NN), Storage Node (SN), switch e um roteador.

6.2. Parâmetros dos Componentes da Plataforma de Nuvem OpenStack

Os tempos médios de falha (MTTF) e de reparo (MTTR) adotados foram obtidos a partir de [Guedes 2019] (Tabela 1).

Tabela 1. Parâmetros dos Componentes

Componente	MTTF(h)	MTTR(h)
PR	2.500,00	0,50
MP	480,00	0,50
MS	1.000,000	0,50
SC	2.480,87	0,50
SO	2.893,00	0,25
HYPERVISOR	2.990,00	1,0
SW	11.200	8,0
RT	80.000	8,0
NEUTRON-SERVER, CINDER-API, CINDER-SCHEDULER, CINDER-VOLUME, GLANCE-API, GLANCE-REGISTRY, HEAT-API-CFN, HEAT-API, HEAT-ENGINE, KEYSTONE, NOVA-API, NOVA-CERT, NOVA-CONDUCTOR, NOVA-CONSOLEAUTH, NOVA-NOVAVNCPROXY, NOVA-SCHEDULER, SAHARA-ALL, SWIFT-PROXY	3000	2,0

6.3. Parâmetros dos Modelo Plaforma Nuvem Privada

O Modelo do Sistema Computacional (SC) (Figura 4) foi adotado para calcular o MTTF e o MTTR do sistema computacional, com valores apresentados na Tabela 1. Os modelos Compute Node (CN) (Figura 5), Network Node (NN) (Figura 6), Cloud Controller (CC) (Figura 8) e Storage Node (SN) (Figura 7) foram adotados para calcular os MTTF's e os MTTR's dos componentes da plataforma *OpenStack*, com valores apresentados na Tabela 2. Esses parâmatros proporcionaram o cálculo da disponibilidade do *Hadoop cluster* na plataforma *OpenStack* através do Modelo da Plataforma Nuvem Privada (Figura 9), com valor de **97,92%**.

Tabela 2. Parâmetros dos modelos

Modelo	MTTF(h)	MTTR(h)
SC	2.480,87	0,50
CN	459.128	0,89
NN	480.289	1,42
SN	571.838	1,30
CC	141.201	1,84

6.4. Análise de Sensibilidade

Com o objetivo de avaliar os componentes mais impactantes na disponibilidade do *Hadoop cluster* configurado na plataforma de nuvem *OpenStack*, foi feita uma análise de sensibilidade paramétrica dos componentes. Essa análise de sensibilidade realizada por

meio do modelo da Plataforma de Nuvem Privada, conforme mostrado na Figura 9, e os parâmetros adotados para o modelos foram mostrados na Tabela 2. Os resultados da análise de sensibilidade são apresentados em ordem decrescente do índice de sensibilidade como mostra a Tabela 3.

6.5. Análise da Disponibilidade dos Modelos

Após a análise de sensibilidade, verificou-se que o componente da plataforma de nuvem *OpenStack* que mais impacta na disponibilidade do sistema é o CC. Com o objetivo de melhorar a disponibilidade do sistema que é **97,92%**, esse componente teve a atribuição das redundâncias ativo-ativo, *hot standby* e *cold standby*.

A Tabela 4 mostra os resutados da disponibilidade do *Hadoop cluster* configurado na plataforma de nuvem *OpenStack*, quando não há a atribuição de nenhum mecanismo de redundância ao componente CC e quando os mecanimos de redundância ativo-ativo, *hot standby* e *cold standby* são atribuídos ao componente CC.

Tabela 3. Análise de Sensibilidade

Componente	Importância
CC	0,9920
NN	0,9821
SN	0,9814
CN	0,9811
SW	0,9799
RT	0,9793

Tabela 4. Disponibilidade

Modelo	Disponibilidade
Sem Redundância	97,92%
Hot Standby	99,18%
Cold Standby	99,11%
Ativo-Ativo	99,04%

A plataforma *OpenStack* apresentou maior disponibilidade com o mecanismo de redundância *hot standby* atribuído ao componente CC em relação aos mecanismos ativo-ativo e *cold standby*.

7. Conclusão e Trabalho Futuro

Este trabalho tenta reduzir o impacto na disponibilidade dos serviços do *Hadoop cluster* na nuvem privada *OpenStack*. Foi adotado uma estratégia de modelagem baseada em diagramas de blocos de confiabilidade e redes de Petri estocásticas para a análise da infraestrutura do *Hadoop cluster* configurado na plataforma *OpenStack*. A estratégia de modelagem adota os modelos Compute Node (CN), Network Node (NN), Cloud Controller (CC) e Storage Node (SN) para representar a plataforma *OpenStack* e os modelos ativo-ativo, *cold stadby* e hot standby para representar os mecanismos de redundância. A análise de sensibilidade foi adotada para identificar o componente de maior impacto na disponibilidade dos serviços onde se constatou ser o componente *Cloud Controller*. Foi atribuído diferentes mecanismos de redundância ativo-ativo, hot standby e cold standby ao componente com o objetivo de aumentar a disponibilidade dos serviços na nuvem privada. Verificou-se que o mecanismo de redundância *hot standby* garante maior disponibilidade do sistema dentre os mecanismos apresentados. A modelagem proposta pode ser adotada antes da implementação da arquitetura, mitigando problemas relacionados à disponibilidade dos serviços de *Big Data* configurados na plataforma *OpenStack*. Como trabalho futuro, pretende-se injetar falhas e analisar a disponibilidade do *Hadoop cluster* na plataforma de nuvem *OpenStack*.

Referências

Bauer, E., Adams, R., and Eustace, D. (2012). *Service Availability and Service Reliability*, pages 20–33.

Bhathal, G. S. and Dhiman, A. S. (2018). Big data solution: Improvised distributions framework of hadoop. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 35–38.

Bhosale, V., Thakar, A., Pandit, C., Deshpande, A., and Khanuja, H. (2018). Hadoop in action: Building a generic log analyzing system. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–7.

Chang, W. and Grady, N. (2019). Nist big data interoperability framework: Volume 1, definitions.

Florin, G., Fraize, C., and Natkin, S. (1991). Stochastic petri nets: Properties, applications and tools. *Microelectronics Reliability*, 31(4):669–697.

Guedes, E. A. C. (2019). Availability and capacity modeling for virtual network functions based on redundancy and rejuvenation supported through live migration.

Liberman, J. (2015). Arquiteture overview - deploying openstack sahara on red hat enterprise linux openstack platform 6.

Marynowski, J., Santin, A., and Pimentel, A. (2015). Method for testing the fault tolerance of mapreduce frameworks. *Computer Networks*.

Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA.

Melo, C., Matos, R., Dantas, J., and Maciel, P. (2017). Capacity-oriented availability model for resources estimation on private cloud infrastructure. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 255–260.

O'Connor, P. D. T. (1998). An introduction to reliability and maintainability engineering, charles e. ebeling, mcgraw-hill, 1997. *Quality and Reliability Engineering International*, 14(4):295–295.

Sousa, E. T. G. D. (2015). Modelagem de desempenho, dependabilidade e custo para oplanejamento de infraestruturas de nuvens privadas.

Trivedi, K. (2016). *Statistical Inference. Probability and Statistics with Reliability, Queuing and Computer Science Applications*, chapter 10, pages 661–752. John Wiley Sons, Ltd.

Trivedi, K., Matias, R., Maciel, P., and Kim, D. S. (2012). Dependability modeling. In *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, pages 53–97.