UFPE

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS ALEXANDRE SILVA DE MELO

Planejamento de infraestruturas computacionais para o provimento de serviços baseados em blockchain

Recife
2021

CARLOS ALEXANDRE SILVA DE MELO

Planejamento de infraestruturas computacionais para o provimento de serviços baseados em blockchain

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

Orientador: Paulo Romero Martins Maciel
Coorientador: Jamilson Ramalho Dantas

Recife
2021

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

M528p Melo, Carlos Alexandre Silva de
Planejamento de infraestruturas computacionais para o provimento de serviços baseados em blockchain / Carlos Alexandre Silva de Melo. – 2021.
141 f.: il., fig., tab.

Orientador: Paulo Romero Martins Maciel.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências e apêndices.

1. Redes de computadores e sistemas distribuídos. 2. Blockchain. 3. Custo. 4. Confiabilidade. I. Maciel, Paulo Romero Martins (orientador). II. Título

004.6 CDD (23. ed.) UFPE - CCEN 2021 – 195

Carlos Alexandre Silva de Melo

“Planejamento de Infraestruturas Computacionais para o Provimento de Serviços Baseados em Blockchain”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos

Aprovado em: 10/09/2021.

_____

Orientador: Prof. Dr. Paulo Romero Martins Maciel

BANCA EXAMINADORA

_____

Prof. Dr. Nelson Souto Rosa
Centro de Informática / UFPE

_____

Prof. Dr. Kelvin Lopes Dias
Centro de Informática / UFPE

_____

Prof. Dr. Gustavo Rau de Almeida Callou
Departamento de Computação / UFRPE

_____

Prof. Dr. Jó Ueyama
Instituto de Ciências Matemáticas e de Computação / USP

_____

Prof. Dr. Virgilio Augusto Fernandes Almeida
Departamento de Ciência da Computação / UFMG

Dedico este trabalho a Deus, ao meu pai, a minha mãe e em especial a minha esposa, Raquel, meu porto seguro perante todas as dificuldades enfrentadas.

AGRADECIMENTOS

Agradeço a Deus e a minha família pela oportunidade para lutar, aprender e crescer todos os dias em busca de meus ideais, seja profissionalmente ou como ser humano. Agradeço também ao Professor Paulo Maciel, principalmente pela sua paciência e orientação desde o início do mestrado (2014) até o presente momento, sabendo, de antemão, que as oportunidades por ele a mim fornecidas me levarão ainda mais longe.

Muito obrigado aos membros da banca por sua disponibilidade, empenho e presente participação nesta tese de doutorado. Aos amigos Jamilson, Renata, Ronieirison, Paulo, Tullyo, Jean, Rubens e Danilo pelo apoio e colaborações de pesquisa realizadas ao longo dos últimos anos. Agradeço também a minhas amigas Brenda e Nayá pelos melhores cafés do mundo e pela amizade sem igual, e também aos amigos de graduação que ainda hoje continuam ao meu lado para o que der e vier, um forte abraço a Samuel, Emanoel, Micael, Joseniel, José, Ygor, Adriano, João e Valter.

Por fim, gostaria de agradecer a Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) pelo apoio financeiro no desenvolvimento desta pesquisa e ao Centro de Informática da Universidade Federal de Pernambuco por ceder o espaço e os recursos necessários a implementação do presente trabalho.

RESUMO

Através do surgimento das redes *peer-to-peer*, nos tornamos detentores de grandes poderes e responsabilidades no processo de provimento de serviços através da Internet. Até então, nosso alcance era bidirecional, uma limitação inerente ao modelo cliente-servidor adotado pela grande maioria dos serviços em operação na Internet. Hoje em dia, graças ao uso de aplicações como o BitTorrent, somos também capazes de prover o conteúdo que consumimos, adicionando um nível extra de complexidade à uma rede outrora estática. Uma das precursoras da mais recente evolução no âmbito de redes *peer-to-peer* são as tecnologias baseadas em registro distribuído, como as blockchains, que permitem a seus usuários serem os auditores, executores e clientes que submetem as mais diversas transações a um enorme ecossistema interconectado. Todavia, assim como em modelos tradicionais de prestação de serviços através da Internet, faz-se necessária a quantificação de sua viabilidade, seja a nível pessoal ou empresarial. A presente tese de doutorado avalia e apresenta um conjunto de modelos formais para avaliação de disponibilidade e confiabilidade de infraestruturas computacionais capazes de hospedar aplicações baseadas em blockchain. Além disso, apresentamos os custos associados à implantação e manutenção dessas infraestruturas, bem como, a sua respectiva avaliação de desempenho, visando o estabelecimento de uma relação de performabilidade entre as métricas citadas. Dentre os resultados obtidos, podemos citar o impacto das políticas de endossamento sobre os custos de manutenção, disponibilidade e confiabilidade das infraestruturas avaliadas. As infraestruturas baseadas na política de endossamento do tipo AND apresentaram piores resultados gerais, maiores custos, maior indisponibilidade e menor confiabilidade em relação as políticas de endossamento KooN e OR. Além disso, avaliamos o desempenho de uma plataforma na execução de uma aplicação básica, métricas como vazão e latência foram consideradas, tal estudo aponta crescimento no consumo de recursos que pode estar relacionado ao Envelhecimento de Software.

Palavras-chave: modelos; disponibilidade; confiabilidade; custo; desempenho; blockchain.

ABSTRACT

Through the rise of peer-to-peer networks, we have become holders of great powers and responsibilities in providing services over the Internet. Until then, our reach was bidirectional, a limitation inherent in the client-server model adopted by the vast majority of services operating in the Internet. Thanks to BitTorrent-like applications, we can become true content providers by adding an extra level of complexity to an otherwise static network. One of the forerunners of the latest evolution in peer-to-peer networks is distributed registry-based technologies, such as blockchains, which allow their users to be the auditors, executors, and customers who submit the most diverse transactions to a huge ecosystem interconnected. However, as in traditional models of providing services through the Internet, it is necessary to quantify their viability, whether personal or business level. This doctoral thesis evaluates and presents a set of formal models for assessing the availability and reliability of computational infrastructures capable of hosting blockchain-based applications. In addition, we present the costs associated with the implementation and maintenance of these infrastructures and their performance evaluation, aiming at establishing a performance relationship between the metrics mentioned above. AAmong the obtained results, we can mention the impact of the endorsement policies on the maintenance costs, availability, and reliability of the evaluated infrastructures. Infrastructures based on the AND-type endorsement policy showed the worst overall results, higher costs, greater unavailability, and lower reliability than the KooN and OR endorsement policies. In addition, we evaluated the performance of a platform in the execution of a basic application, metrics such as throughput and latency were considered, and this study points to a growth in resource consumption that may be related to Software Aging.

Keywords: models; availability; reliability; cost; performance; blockchain.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BaaS	Blockchain como Serviço
CLI	Interface de Linha de Comando
CTMC	Cadeias de Markov de Tempo Contínuo
DDB	Saldo Decrescente Duplo
DDoS	Ataque de Negação de Serviço
DLT	Tecnologias de Registro Distribuído
DoE	Design of Experiments
DTMC	Cadeias de Markov de Tempo Discreto
FT	Árvore de Falha
KooN	K-out-of-N
KVM	Kernel-based Virtual Machine
MoDCS	Modeling of Distributed Concurrent System
MTTF	Tempo Médio Até a Falha
MTTR	Tempo Médio Até o Reparo
NRT	Non-repair Time
Opex	Despesas de Operação
pBFT	Protocolo Prático de Consenso Tolerante a Falhas Bizantinas
PoW	Prova de Trabalho
RBD	Diagramas de Bloco de Confiabilidade
RBFT	Tolerância Redundante a Falhas Bizantinas
SLA	Acordos de Nível de Serviço
SPEC	Standard Performance Evaluation Corporation
SRN	Stochastic Reward Nets
SUT	Sistema Sob Teste
TCO	Custo Total de Posse
TPC	Transaction Processing Performance Council
TPS	Transactions per Second
TTR	Time to Repair
VMs	Máquinas Virtuais

SUMÁRIO

1 INTRODUÇÃO

> *Dreams. Each man longs to pursue his dream. Each man is tortured by this dream, but the dream gives meaning to his life. Even if the dream ruins his life, man cannot allow himself to leave it behind. In this world, is man ever able to possess anything more solid, than a dream?*
>
> — Void, *Berserk*, 1989

Blockchain é uma tecnologia voltada à solução de problemas de confiança por parte de usuários e na centralização de dados e informações em grandes organizações (GUPTA, 2017; SWAN, 2015). Tais soluções são resultado de características inerentes à Blockchain, uma infraestrutura para aplicações distribuídas que tem na imutabilidade de registros compartilhados a garantia de que após a realização de uma transação e, o seu respectivo armazenamento no sistema, a mesma não poderá (facilmente) ser alterada (ZYSKIND; NATHAN; PENTLAND, 2015).

Os registros distribuídos possuem mecanismo de armazenamento similar aos fornecidos pelos tradicionais bancos de dados distribuídos. Todavia, sem a obrigatoriedade de intermediários ou cópias primárias hospedadas em servidores únicos (CROMAN et al., 2016; KUROSE; ROSS; ZUCCHI, 2007). Além desta característica, existem alguns outros benefícios relacionados à confiança na realização de transações, estes fornecidos pela utilização de políticas de consenso (ZYSKIND; NATHAN; PENTLAND, 2015; TAPSCOTT; TAPSCOTT, 2016). Essas políticas implicam a necessidade única de confiança no sistema como um todo e não nos indivíduos que o integram (NAKAMOTO et al., 2008).

A blockchain traz consigo a possibilidade de movimentações financeiras, realização de acordos, transferência de posse de bens, valores e serviços (GUPTA, 2017). Para tanto, podem fazer uso de *smart contracts* ou contratos inteligentes, cuja funcionalidade se dá de forma similar a de contratos convencionais, mas sem a necessidade de órgãos reguladores que determinem o seu cumprimento. Deste modo, sua execução se dá de forma autônoma com base em resultados, premissas ou metas previamente estabelecidas nas regras de negócio.

A presente tese de doutorado avalia infraestruturas computacionais capazes de prover blockchain para terceiros, ou seja, de prover o ambiente necessário à execução de aplicações e serviços baseados nessa tecnologia. Esses ambientes devem atender a uma série de requisitos mínimos de disponibilidade, confiabilidade, desempenho e custos. Também propomos uma ferramenta, composta por modelos de disponibilidade, confiabilidade e custos para dar suporte a empresas e pessoas que planejam hospedar ou prover aplicações e serviços que façam uso de blockchain. A seguir, apresentamos as justificativas que nos levaram a escolha do presente tema de pesquisa.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

A Blockchain é a Máquina de Confiança por trás do Bitcoin[1], tal tecnologia passou (e ainda passa) por diversos processos transformativos ao longo de sua curta história. O que antes era considerado *apenas* um substituto aos modelos de negócio tradicionais, inclusive instituições financeiras e mecanismos de pagamento, tornou-se um mecanismo associado à segurança e transparência para as mais diversas aplicações.

As aplicações vão desde as já mencionadas criptomoedas(NAKAMOTO et al., 2008) aos prontuários médicos de pacientes em um hospital(EKBLAW et al., 2016). Através de tecnologias baseadas em blockchain, novos métodos de detecção e proteção contra Ataque de Negação de Serviço (DDoS) foram implementados(RODRIGUES et al., 2017). Também é possível fiscalizar a cadeia de suprimentos e todo o processo de produção dos produtos que chegam à nossa mesa, algo que pode estar a um clique do seu smartphone(SABERI et al., 2019). Os exemplos são diversos e não exaustivos, e vão além do impacto multibilionário sobre o mercado financeiro internacional, mas também ao seu impacto ambiental[2] e social.

Graças ao sucesso adquirido, a blockchain tornou-se uma tendência entre os grandes *players* de computação em nuvem como Amazon[3], Microsoft[4], SalesForce[5] e Alibaba[6]. Nesse novo e competitivo mercado qualquer contribuição poderá mudar completamente o que sabemos sobre este modelo computacional.

Essa nova revolução tecnológica permitiu o surgimento de uma série de desdobramentos que possibilitaram àqueles interessados em desenvolver, contratar ou utilizar recursos e serviços baseados em blockchain escolher por fazê-lo em infraestruturas computacionais públicas ou privadas.

Ambientes privados podem se mostrar viáveis em muitos cenários como no armazenamento de dados sensíveis, nesta abordagem, geralmente, utilizamos de plataformas abertas e de software livre para redução de custos associados à sua implantação. Todavia, os custos tendem a ser bastante elevados se comparados àqueles relacionados a serviços providos por um grande *player*.

1.2 OBJETIVOS

O principal objetivo desta tese é responder a seguinte pergunta de pesquisa:

[1] The Economist: https://www.economist.com/leaders/2015/10/31/the-trust-machine
[2] https://www.bbc.com/news/technology-56012952
[3] Amazon: https://aws.amazon.com/pt/blockchain/
[4] Microsoft: https://azure.microsoft.com/pt-br/solutions/blockchain/
[5] SalesForce: https://www.salesforce.com/company/news-press/press-releases/2019/05/192915-i/
[6] Alibaba: https://www.alibabacloud.com/products/baas

> Que tipo de infraestrutura computacional é capaz de hospedar aplicações e serviços baseados em blockchain atendendo a uma série de requisitos mínimos de disponibilidade, confiabilidade, desempenho e custo?

Para alcançarmos este objetivo, propomos uma metodologia para avaliação de desempenho de aplicações baseadas em blockchain, além de uma ferramenta de análise e avaliação de disponibilidade e custo capaz de auxiliar o processo de tomada de decisão. A ferramenta proposta leva em conta as necessidades dos analistas, tomadores de decisão e a relação geral entre custo-benefício do serviço prestado. Tal objetivo geral poderá ser alcançado através de um subconjunto de passos e metas que caracterizam os objetivos específicos a seguir:

- Modelar e avaliar disponibilidade e confiabilidade;
- Implantar e avaliar o desempenho de aplicação baseada em blockchain;
- Conceber modelos de custo por meio de expressões algébricas, associáveis aos custos de implantação e manutenção das infraestruturas previamente avaliadas;
- Implementar ferramenta a partir dos modelos propostos e generalizar o processo de identificação de gargalos;
- Criar e avaliar cenários que demonstrem a viabilidade dos modelos e da ferramenta proposta.

1.3 CONTRIBUIÇÕES

Podemos citar como possíveis contribuições da tese de doutorado os seguintes itens ordenados por grau de importância (do maior para o menor):

1. Modelos para avaliação de disponibilidade e confiabilidade de infraestruturas computacionais capazes de hospedar aplicações baseadas em blockchain;
2. Aplicação para avaliação de disponibilidade e custos de diversas infraestruturas com base em políticas de endossamento distintas;
3. Avaliação de infraestrutura privada visando a obtenção de valores de vazão, latência e consumo de recursos;
4. Estabelecimento de relação custo $\times$ benefício entre os valores de disponibilidade obtidos através da avaliação dos modelos e o custo de aquisição e manutenção das infraestruturas avaliadas;
5. Metodologia de pesquisa que possibilita a replicação das demais contribuições citadas.

1.4 ORGANIZAÇÃO DA TESE

O restante desta tese é organizado como segue: A Seção 2 apresenta a fundamentação teórica do trabalho como conceitos acerca de virtualização, blockchain, desempenho e técnicas de análise e avaliação de disponibilidade e confiabilidade de sistemas computacionais. Já a Seção 3 lista os principais trabalhos relacionados ao que é proposto na presente tese e o que os diferencia do que propomos. A Seção 4 apresenta um conjunto de metodologias de modelagem e avaliação seguidas por esta proposta de tese. A Seção 5 apresenta o conjunto de modelos propostos. Já a Seção 6 apresenta a ferramenta derivada dos modelos propostos e posteriormente utilizada na Seção 7, como exibição de sua aplicabilidade através de dois estudos de caso do mundo real e outros estudos relacionados à avaliação de desempenho e confiabilidade. As conclusões, limitações e trabalhos futuros estão expostos ao longo da Seção 8.

2 FUNDAMENTAÇÃO TEÓRICA

> *Humans cannot create anything out of nothingness. Humans cannot accomplish anything without holding onto something. After all, humans are not gods.*
>
> — Yoshiyuki Sadamoto, *Neon Genesis Evangelion*, 1995

Esta seção apresenta alguns dos principais conceitos e tecnologias empregados no processo de desenvolvimento desta tese. Iniciamos a nossa apresentação pela virtualização de aplicações através do Docker, e percorremos o conjunto de técnicas para análise e modelagem dos atributos de dependabilidade, além de custos de implantação e manutenção, avaliação de desempenho e blockchain.

2.1 VIRTUALIZAÇÃO E DOCKER

Docker corresponde ao conjunto de tecnologias que inclui, mas não limita-se, a um formato padrão de imagens para contêineres e uma biblioteca para sua execução. Um contêiner é uma plataforma virtualizada que executa uma imagem contendo programas e arquivos necessários ao provimento de um serviço. O Docker também conta com uma Interface de Linha de Comando (CLI) própria e um conjunto de Application Programming Interface (API) para gerenciamento que garantem o completo ciclo de vida de aplicações em um universo de microsserviços (ARUNDEL; DOMINGUS, 2019).

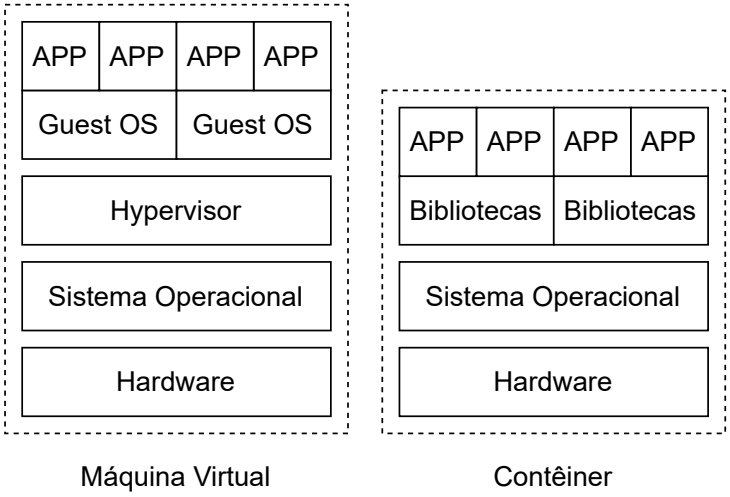
O Docker também pode ser resumido ao padrão atual para criação e gerenciamento de contêineres. Essa tecnologia possibilita a usuários e companhias virtualizar seus recursos sem a necessidade da completa virtualização de um novo sistema operacional. Por esses e outros motivos contêineres são ditos Máquinas Virtuais (VMs) leves, haja vista a necessidade única de virtualização da aplicação e de seus recursos empacotados em uma única imagem.

Uma aplicação virtualizada através do Docker faz uso do Kernel do sistema operacional da máquina hospedeira. Cada aplicação é distribuída sob um conjunto de *namespaces* responsáveis pelo processo de isolamento completo entre as várias instâncias em execução dentro de um único nó computacional. Este funcionamento difere das tradicionais VMs, que geralmente são gerenciadas por hipervisores em operação no sistema operacional hospedeiro, tais como Kernel-based Virtual Machine (KVM), VMWare e Virtual Box.

A Figura 1 apresenta uma comparação entre a virtualização de aplicações em máquinas virtuais e a virtualização através de contêineres que podem ser geridos pelo Docker. O número de camadas em cada pilha difere substancialmente entre os dois tipos de virtualização, o que, geralmente, garante um maior desempenho das aplicações que executam em contêineres. No âmbito da computação em nuvem e grandes centros de dados, essa

característica tem feito a diferença em processos de *autoscaling* e consequentemente do aumento na disponibilidade de serviços.

Figura 1 – Máquinas Virtuais vs. Contêineres



Fonte: (ARUNDEL; DOMINGUS, 2019)

2.2 DEPENDABILIDADE

Dependabilidade é a habilidade de um sistema computacional de entregar ou fornecer um serviço de maneira justa e confiável (AVIZIENIS et al., 2004). Esta definição considera a dependabilidade como uma grande caixa preta que só pode ser analisada do ponto de vista do usuário do sistema. Sob esta perspectiva, a sua avaliação se dá através de análise do comportamento esperado desta caixa, com base em suas respostas para cada entrada recebida (TRIVEDI; BOBBIO, 2017).

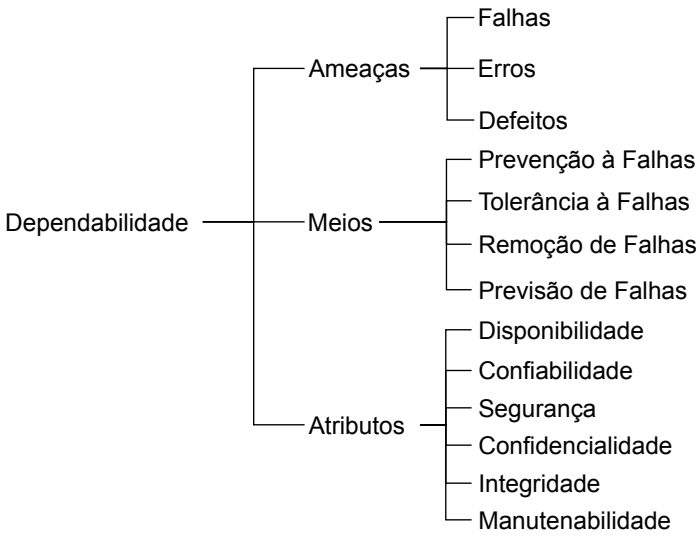
A avaliação de dependabilidade costuma se inter-relacionar ao estudo de um conjunto de **Ameaças** que impedem a entrega do serviço de maneira desejada; dos **Meios**, responsáveis pela garantia dos resultados inerentes aos **Atributos** que dizem respeito à entrega do serviço. Tais propriedades podem ser esquematizadas através de uma árvore, como na Figura 2.

No âmbito desta tese, realizamos a avaliação da disponibilidade e da confiabilidade de infraestruturas capazes de hospedar aplicações baseadas em blockchain. Estes são dois dos principais atributos presentes na árvore de dependabilidade.

2.2.1 Disponibilidade

A disponibilidade é a probabilidade de um sistema estar operacional, ou seja, acessível aos seus usuários dentro de um período de tempo preestabelecido, seja este agora ou no futuro (IEEE, 1990). Esta probabilidade, pode ser dada através do tempo em que o sistema

Figura 2 – Árvore de Dependabilidade



Fonte: (AVIŽIENIS et al., 2001)

tende a ficar disponível ou seu *uptime*, e o seu período de indisponibilidade, também chamado de *downtime*. Estes valores são legalmente determinados em Acordos de Nível de Serviço (SLA). A Expressão 2.1 representa o cálculo de disponibilidade (MACIEL et al., 2011) para um sistema ou serviço.

$$\mathrm{A} = \frac{\mathrm{E[Uptime]}}{\mathrm{E[Uptime]} + \mathrm{E[Downtime]}} \tag{2.1}$$

onde **A** equivale à *availability* ou disponibilidade do sistema, **E** é o valor esperado para a métrica, ou seja, sua esperança matemática.

Uma outra alternativa para a representação da disponibilidade de um sistema é em termos de Tempo Médio Até a Falha (MTTF) e Tempo Médio Até o Reparo (MTTR) do sistema (MACIEL et al., 2011), como pode ser visto através da Expressão 2.2 (TRIVEDI; BOBBIO, 2017).

$$\mathrm{A} = \frac{\mathrm{MTTF}}{\mathrm{MTTF+MTTR}} \tag{2.2}$$

Em contrapartida ao cálculo da disponibilidade, existe, também, o cálculo da indisponibilidade do sistema. Este que é representado pelo inverso da disponibilidade, ou seja, *1 - A*.

A Equação 2.3 representa o cálculo para o tempo médio até a ocorrência de uma falha do sistema (MACIEL et al., 2011).

$$\mathrm{MTTF} = \int_0^\infty R(t)dt \tag{2.3}$$

Já o cálculo do tempo médio até o reparo (MTTR) do sistema, depende diretamente do resultado proveniente da derivada anterior, e é representado através da Equação 2.4

(MACIEL et al., 2011):

$$\text{MTTR} = \text{MTTF} \times (\frac{\text{UA}}{\text{A}}), \tag{2.4}$$

O período de *downtime* do sistema pode ser entendido como o período em que o serviço esteve indisponível dentro de um intervalo de tempo. O cálculo desta métrica leva em consideração a relação entre o tempo levado para uma equipe de manutenção locomover-se ao local da falha e a detecção de sua ocorrência. Este período é conhecido como *período-sem-reparo* ou Non-repair Time (NRT), e o tempo necessário para reparo da falha ou Time to Repair (TTR). Deste modo, o *downtime* de um serviço pode ser calculado através da expressão **Downtime = NRT + TTR** (MACIEL et al., 2011; TRIVEDI et al., 1996).

Também é possível representar a disponibilidade do sistema através do número de noves equivalente (ÖHMANN; SIMSEK; FETTWEIS, 2014), como mostra a Tabela 1. Por exemplo, um sistema com quatro noves de disponibilidade pode ser classificado como tolerante a falhas, tendo um downtime anual de menos de uma hora, enquanto que a alta disponibilidade é alcançada através de sistemas com cinco ou mais noves.

Tabela 1 – Disponibilidade em Número de Noves

# de 9's	Disponibilidade (%)	Tipo de Sistema	Downtime Anual
1	90	Não gerenciado	5 Semanas
2	99	Gerenciado	4 Dias
3	99,9	Bem Gerenciado	9 Horas
4	99,99	Tolerante a Falhas	1 Hora
5	99,999	Alta Disponibilidade	5 minutos
6	99,9999	Muito Alta Disponibilidade	30 segundos
7	99,99999	Ultra Disponibilidade	3 segundos

Fonte: (ÖHMANN; SIMSEK; FETTWEIS, 2014)

O cálculo do número de noves pode ser feito através da expressão $|\log_{10}(\text{UA})|$ que leva em conta a indisponibilidade do sistema, ou 1 - A, como anteriormente mencionado. É importante salientar que a operação de módulo é utilizada para que o valor retornado seja sempre positivo.

2.2.2 Confiabilidade

Há um outro atributo de dependabilidade ligado diretamente aos tempos de falha do sistema, mas que diferentemente da disponibilidade, não leva em consideração os valores associados ao seu reparo, este atributo é a **confiabilidade**.

A confiabilidade nada mais é do que a probabilidade do sistema ter executado sua função sem interrupções até um limite de tempo previamente estabelecido (AVIŽIENIS et al., 2001; BOLCH et al., 2006).

A confiabilidade de um sistema pode ser medida de maneira semelhante ao cálculo da disponibilidade, porém, as taxas relacionadas ao reparo do mesmo não são consideradas. A Equação 2.5 apresenta como este cálculo pode ser realizado.

$$R(t) = P(T > t), t \geq 0 \tag{2.5}$$

onde T é a variável aleatória que representa o tempo esperado para a falha do sistema, e que está dentro do intervalo $[0, t]$. Já R é a confiabilidade em si (KUO; ZUO, 2003), do inglês *reliability*.

2.3 MODELOS

Um modelo é uma abstração ou visão geral de um sistema real (MENASCE et al., 2004). Seu nível de granularidade, ou seja, o seu grau de detalhes, depende principalmente do sistema que está sendo modelado e da habilidade do analista em representá-lo (MACIEL et al., 2011). No geral, existem dois tipos de modelos científicos utilizados no processo de avaliação de sistemas de recursos compartilhados, os modelos analíticos e os modelos de simulação (MENASCE et al., 2004; MACIEL et al., 2011).

Os modelos analíticos correspondem ao conjunto de fórmulas e/ou algoritmos computacionais que proveem os valores desejados para métricas de desempenho e atributos de dependabilidade de um sistema, tudo isso com base em um conjunto de parâmetros de entrada (MENASCE et al., 2004; TRIVEDI; BOBBIO, 2017). Enquanto que os modelos de simulação consistem de programas computacionais que executam ou simulam atividades similares ao sistema real.

Os modelos de simulação geralmente são mais caros de se conceber que os modelos analíticos, muitas vezes levando ao questionamento de sua viabilidade perante a implementação do sistema real ou de um protótipo que o represente. Todavia, os modelos analíticos apresentam menor acurácia, mas uma melhor relação custo × benefício em seu desenvolvimento. Quanto à representação de modelos analíticos em si, estes geralmente são classificados em modelos combinatoriais (sem espaço de estados) e modelos baseados em estado:

- Modelos combinatoriais capturam as condições que podem fazer um sistema falhar ou continuar trabalhando de acordo com as relações estruturais entre os componentes do sistema. Os Diagramas de Bloco de Confiabilidade (RBD) e as Árvore de Falha (FT) são os principais exemplos deste tipo de modelo;

- Modelos de espaço de estados representam o comportamento do sistema (ex., atividades de falha e reparo) com base na ocorrência de eventos (taxas ou funções de distribuição). Esses modelos permitem a representação de relações mais complexas entre os componentes do que permitiriam os modelos combinatoriais, possibilitando

também o seu uso no processo de avaliação de desempenho. Alguns dos principais exemplos deste tipo de modelo são: Redes de Petri e Cadeias de Markov.

A modelagem possibilita uma visão de alto nível do sistema real, mesmo quando trabalhamos em um ambiente sob a perspectiva do campo das ideias, ou até mesmo com um computador idealizado (SIPSER, 2006). Outra vantagem é sua flexibilidade, que permite a adoção de parâmetros e obtenção de resultados rapidamente. Além disso, possuem um baixo custo para avaliação, possibilitando uma série de simplificações e uma visão de alto nível que pode ser apresentada aos vários níveis hierárquicos de uma organização.

2.3.1 Diagramas de Bloco de Confiabilidade

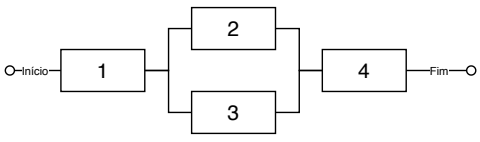
Os Diagramas de Bloco de Confiabilidade (RBDs) são formalismos inicialmente propostos para a análise de confiabilidade de sistemas e suas interações. Esta técnica permite a utilização de combinações para representar o comportamento de um sistema e seus módulos, produzindo uma análise confiável.

Mais tarde, os RBDs tiveram suas características estendidas ao cálculo da disponibilidade dos sistemas computacionais, quer sejam estes pequenos e simples ou grandes e complexos, porém não concorrentes ou que apresentem dependência entre seus componentes (MACIEL et al., 2011; MACIEL; LINS; CUNHA, 1996; KUO; ZUO, 2003).

Através de RBDs é possível descrever a interação entre os diversos componentes de um sistema, utilizando-se apenas de um conjunto de blocos de confiabilidade. Cada bloco pode representar um único componente do sistema ou subcomponentes que o compõem. Em um modelo RBD, um sistema estará disponível se for possível se, e somente se, for possível realizar o tracejo de uma linha entre os vértice de (**Início**) e (**Fim**). Além disso, todos os componentes dentro desta linha precisam estar funcionais.

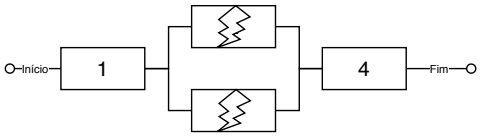
Um possível exemplo do funcionamento de um RBD é apresentado na Figura 3, que mostra um sistema com quatro componentes. Observa-se aqui dois dos principais tipos de blocos utilizáveis por RBDs: série, **Componentes 1 e 4**, e em paralelo, **Componentes 2 e 3**. O sistema estará disponível apenas se for possível desenhar uma linha conectando o **início** ao **fim**, o que é possível por dois caminhos distintos. Em caso de inexistência de um caminho entre **início** e **fim**, dizemos que o sistema está em estado de falha, como na Figura 4, que mostra um cenário onde os **Componentes 2 e 3** não estão mais operacionais.

Figura 3 – Exemplo de Diagrama de Bloco de Confiabilidade Operacional



Fonte: (MACIEL et al., 2011)

Figura 4 – Exemplo de Diagrama de Bloco de Confiabilidade em Falha



Fonte: (MACIEL et al., 2011)

Para delinear os caminhos possíveis dentro de um RBD é necessário estabelecer uma função estrutural. Essa função é capaz de avaliar a disponibilidade do sistema com base nos seus modos de operação. Considere um sistema S com dois componentes: 1 e 2, podendo este sistema estar em um estado de falha ou operacional, de acordo com o estado de cada um de seus componentes. O estado de um $componente_i$ é dado por uma variável aleatória x_i, tal que:

$$x_i = \begin{cases} 0, & \text{se o componente estiver em falha} \\ 1, & \text{se o componente estiver operacional} \end{cases} \tag{2.6}$$

A confiabilidade e a disponibilidade para o sistema de exemplo, caso os componentes estejam conectados em série, podem ser calculadas através da Equação 2.7 (MACIEL et al., 2011).

$$R_S(t) = R_1(t) \times R_2(t) \tag{2.7}$$

onde $R_1(t)$ é a confiabilidade ou disponibilidade do componente 1, e $R_2(t)$ é a confiabilidade ou disponibilidade do componente 2. Já para um sistema em série com n-elementos, a Equação 2.8 (MACIEL et al., 2011) é adotada.

$$R_S(t) = \prod_{i=1}^{n} R_i(t) \tag{2.8}$$

Para uma sequência de blocos em paralelo a Equação 2.9 pode ser utilizada (MACIEL et al., 2011), com n representando o número de blocos em paralelo.

$$R_P(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \tag{2.9}$$

Uma outra possível representação dos RBDs é através de blocos K-out-of-N (KooN). Blocos KooN são capazes de representar generalizações de estruturas onde um determinado subsistema encontra-se em operação, se e somente se, ao menos k componentes de um total de n estiver operacional. Supondo uma infraestrutura com três componentes (n), dos quais pelo menos dois (k) devam estar operacionais para provimento de algum serviço, dizemos então que essa infraestrutura é do tipo *1 - out - of - 4*.

As estruturas em série e paralelo previamente apresentadas são casos especiais de um KooN. Quando dispomos os blocos em série, dizemos que o valor de K é igual ao de N,

já em paralelo esperamos que ao menos 1 de N precisa estar operante (TRIVEDI et al., 1996). Blocos KooN podem ser matematicamente representados através de uma relação binomial, como mostra a Equação 2.10.

$$R(t) = \sum_{k}^{n} \frac{n!}{k!(n-k)!} \left(e^{-\lambda t}\right)^{k} \left(1 - e^{-\lambda t}\right)^{n-k} \tag{2.10}$$

A visão de alto nível fornecida por modelos do tipo RBD permite que sejam utilizados na representação de muitos sistemas. Porém, dificilmente poderemos demonstrar a existência de dependências através de RBDs, ou seja, só é possível representar módulos independentes. Deste modo, para a representação de sistemas considerados mais complexos são necessárias técnicas mais refinadas de modelagem, como Redes de Petri e Cadeias de Markov.

2.3.2 Cadeias de Markov de Tempo Contínuo

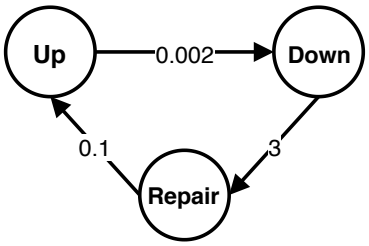
As Cadeias de Markov são capazes de descrever o funcionamento do sistema com base em um conjunto de estados e transições que podem descrever eventos de falha e reparo dentro de um sistema ou componente. As cadeias de Markov são mais convenientes que os RBDs quando queremos descrever propriedades dinâmicas dos sistemas, elas são capazes de descrever dependências entre componentes além de permitir a avaliação de desempenho e concorrência em um sistema (BOLCH et al., 2006; KOLMOGOROFF, 1931).

Cada transição em uma cadeia de Markov representa um processo estocástico $X(t)$, onde $t \in T$ é um conjunto de variáveis aleatórias definidas sobre o mesmo espaço de probabilidades capazes de assumir valores no espaço de estados $S_i \in S$ (CASSANDRAS; LAFORTUNE, 1999; TRIVEDI, 1982). Deste modo, se o conjunto T a qual pertence a variável for discreto, ou seja, com $t = 1, 2, 3, ...$, o processo é dito processo de parâmetro discreto ou tempo discreto. Se T for um conjunto contínuo, tem-se um processo de parâmetro contínuo ou tempo contínuo, assumindo-se distribuições geométricas (Cadeias de Markov de Tempo Discreto (DTMC)) ou exponenciais (Cadeias de Markov de Tempo Contínuo (CTMC))) (SOUSA et al., 2009),(MACIEL et al., 2011).

Um processo estocástico é classificado como um processo de Markov se, para todo $t_0 < t_1 < ... < t_n < t_{n+1}$ e para todo $X(t_0), X(t_1), X(t_2), ..., X(t_n), X(t_{n+1})$, a distribuição condicional de $X(t_{n+1})$ depende somente do último valor anterior $X(t_n)$ e não dos valores que o antecedem $X(t_0), X(t_1), ..., X(t_{n-1})$, isto é, para qualquer número real $X_0, X_1, X_2, ..., X_n, X_{n+1}$, $P(X_{n+1} = s_{n+1} \mid X_n = s_n,\ X_{n-1} = s_{n-1}, ..., X_0 = s_0 = P(X_{n+1} = s_{n+1} \mid X_n = s_n$ (BOLCH et al., 2006). Essa característica, conhecida como ausência de memória, é comumente encontrada em distribuições do mesmo tipo ou que se assemelham às distribuições exponenciais ou geométricas. A ausência de memória garante que os valores do estado atual em uma cadeia de Markov são totalmente independentes dos valores dos estados anteriores (GARG et al., 1995).

Através de uma cadeia de Markov podemos modelar o comportamento de um sistema graficamente, utilizando apenas estados e transições. A Figura 5 mostra um exemplo de uma cadeia de Markov com apenas três estados: **UP**, **DOWN** e **REPAIR**, ou seja, operante, não-operante e em reparo respectivamente, elas representam estados de um sistema genérico; há também três arcos rotulados (transições).

Figura 5 – Exemplo de uma cadeia de Markov



Fonte: (MATOS et al., 2015)

Além da representação gráfica, similar à de autômatos finitos, as cadeias de Markov podem ser representadas em forma de matriz, chamada de matriz de taxa de **transição Q**. Nessa matriz, estão contidos os detalhes que descrevem as transições de todos os estados dentro de uma cadeia de Markov e que são utilizadas durante o processo de resolução de problemas. Os elementos localizados fora da diagonal principal representam a taxa de ocorrência dos eventos que ocasionam transição dos estados da cadeia de Markov. Já os elementos que se encontram na diagonal principal representam os valores necessários para que a soma entre elementos de cada linha seja igual à zero. Além disso, a CTMC apresentada na Figura 5 pode ser representada pela Matriz 2.11, ou matriz Q equivalente, composta por componentes do tipo q_{ii} e q_{ij}, onde $i \neq j$ e $\Sigma q_{ij} = -q_{ii}$.

$$Q = \begin{pmatrix} q_{00} & q_{01} & q_{02} \\ q_{10} & q_{11} & q_{12} \\ q_{20} & q_{21} & q_{22} \end{pmatrix} = \begin{pmatrix} -0.002 & 0.002 & 0 \\ 0 & -3 & 3 \\ 0.1 & 0 & -0.1 \end{pmatrix} \tag{2.11}$$

A matriz de transição Q possui um papel importante no processo de computação do vetor de probabilidade de estados π. A probabilidade de cada estado fornece uma solução de regime estacionário com base em $\pi Q = 0$, onde Q é a matriz de estados e π é o autovetor correspondente ao autovalor unitário da matriz de transição, resultando em um vetor nulo. A soma dos elementos do vetor de probabilidade π deve ser sempre igual a 1, ou seja, $\|\pi\| = 1$ (MACIEL et al., 2011). Já as probabilidades de transição de estados são calculadas através da Equação 2.12.

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

Quanto a soluções que dependem do tempo, ou seja, do tipo transiente, tempos de execução longos são considerados, podendo-se mostrar que a probabilidade dos estados

converge para valores constantes (HERZOG, 2002). O comportamento transiente da cadeia de Markov nos fornece informações de desempenho e dependabilidade sobre os instantes iniciais do sistema (MACIEL et al., 2011).

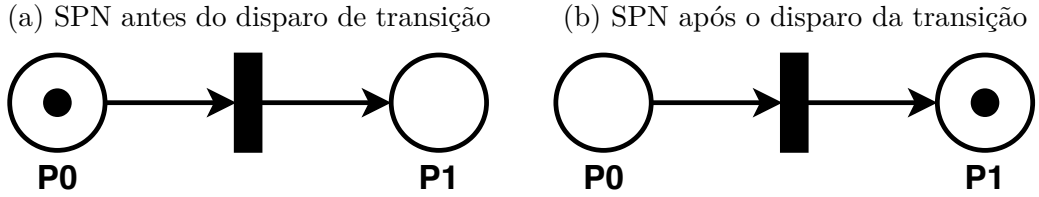
Sistemas mais complexos, ou seja, com um grande número de estados, por vezes, têm sua avaliação de forma gráfica ainda mais complexa. Além de limitações consideráveis no processo de avaliação analítica, perdendo o apelo visual, praticidade e representação em alto nível. Deste modo, faz-se necessária a utilização de abordagens que possam automatizar alguns processos, mas mantendo o mesmo poder de representação, estas são as SPNs, explicadas a seguir.

2.3.3 Redes de Petri Estocásticas

Redes de Petri são uma poderosa ferramenta para análise e modelagem comportamental de sistemas (AGERWALA, 1979; BALBO, 2001). Os principais componentes desse formalismo são os lugares (representados por círculos), transições (representadas por retângulos), arcos (mesma conotação que possuíam nas CTMCs), além das marcações ou *tokens* (pequenos círculos internos aos lugares). De modo geral, os lugares são os possíveis estados alcançáveis pelo sistema, as transições são as ações por ele e nele realizadas, os arcos conectam os lugares às transições enquanto as marcações representam a quantidade de recursos ou o estado atual do sistema (MACIEL; LINS; CUNHA, 1996).

A Figura 6a apresenta uma rede de Petri simples em seu estado inicial, antes do disparo de uma transição. Os disparos de transições ou a ocorrência de ações em uma rede de Petri só são possíveis se o número de recursos/marcações no lugar que as antecede for suficiente e o peso dos arcos não exceder o seu número total. Isso pode ser visto através da Figura 6b que apresenta a ocorrência do disparo de uma transição, onde uma marcação é movida do lugar P0 para o lugar P1.

Figura 6 – Exemplo de uma rede de Petri



Fonte: (MACIEL; LINS; CUNHA, 1996)

De acordo com (MURATA, 1989), uma rede de Petri pode ser definida através de uma $5-tupla$, do tipo PN $= (P, T, F, W, M_0)$, onde:

$P = (p_1, p_2, ..., p_n)$ é um conjunto finito de lugares,

$T = (t_1, t_2, ..., t_m)$ é um conjunto finito de transições,

$F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos,

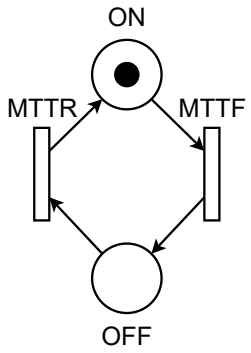
$W : F \to \{1, 2, 3, ...\}$ é a função de peso,

$M_0 : P \to \{0, 1, 2, 3, ...\}$ são as marcações iniciais,

$P \cap T = \oslash$ e $P \cap T \neq \oslash$. ...

O avanço no estudo das redes de Petri proporcionou uma evolução em seu poder de representação. Hoje é possível modelar, também, sistemas assíncronos, temporizados, concorrentes e não-determinísticos através de redes de Petri estocásticas (SPN) (GERMAN, 2000). As transições em uma SPN são temporizadas e sua representação gráfica difere daquelas anteriormente apresentadas. Existem, também, os arcos inibidores, que possuem um círculo na ponta de sua representação gráfica, indicando que só irá disparar na inexistência de marcações no lugar que o antecede. A Figura 7 apresenta o exemplo de uma rede de Petri estocástica aplicada a avaliação de disponibilidade de um determinado componente ou sistema.

Figura 7 – Exemplo de uma SPN aplicada a avaliação de disponibilidade



Fonte: (MACIEL et al., 2017)

A marcação no lugar ON é um indicativo de que o componente está operacional, as transições MTTF e MTTR são ambas temporizadas e requerem a passagem de tempo para que possam disparar, representando a respectiva falha ou reparo desse componente. O lugar OFF indica a inoperabilidade do componente ou sistema.

Os arcos inibidores são capazes de determinar se um dado local possui ou não marcações. As transições temporizadas possuem o tempo de disparo preestabelecido e baseado em uma distribuição de probabilidade. As transições comuns anteriormente apresentadas passaram a se chamar transições imediatas e possuem prioridade de disparo ante os demais tipos de transição.

A rede de Petri estocástica (MACIEL; LINS; CUNHA, 1996) pode ser definida através de uma $9 - tupla$, do tipo SPN $= (P, T, I, O, H, \Pi, G, M_0, ATTs)$, onde:

$P = (p_1, p_2, ..., p_n)$ é o conjunto de lugares,

$T = (t_1, t_2, ..., t_m)$ é o conjunto de transições imediatas e temporizadas $P \cap T \neq \oslash$,

$I \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações),

$O \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações),

$H \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações),

$\Pi \in \mathbb{N}$ é um vetor que associa o nível de prioridade de cada transição,

$G \in (\mathbb{N} \to \{verdadeiro, falso\})^m$ é o vetor que associa uma transição de guarda relacionada à marcação do lugar a cada transição,

$M_0 = P \to \{0, 1, 2, 3, ...\}$ é o vetor que associa a marcação inicial de cada lugar;

ATTs compreende o conjunto de atributos associados a transições, como a função de distribuição, política de memória, grau de ocorrência e o peso de cada transição.

As transições temporizadas são caracterizadas por semânticas de disparo distintas. Estas que podem ser de três tipos: *single server*, onde apenas uma marcação é disparada por vez; *multiple server* é possível realizar k disparos de uma única vez;, e *infinite server*, onde é possível realizar um número infinito de disparos (MARSAN et al., 1994).

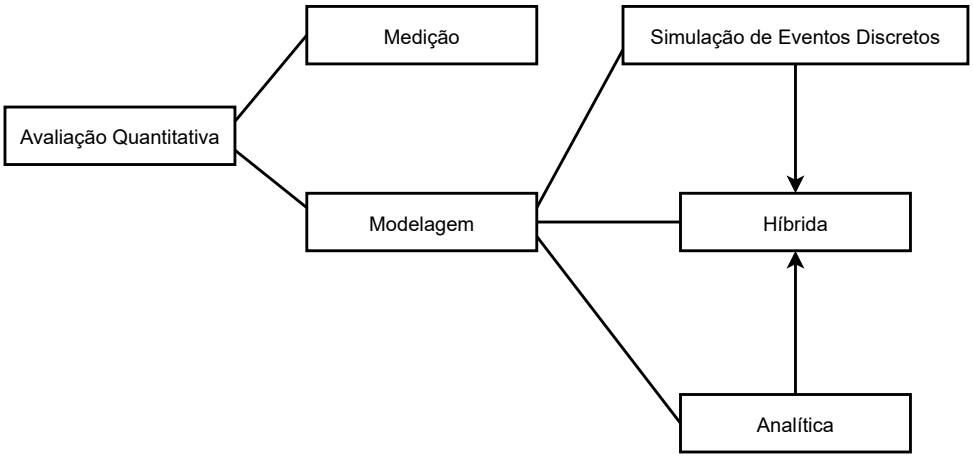
Redes de Petri estocásticas com um número finito de lugares e transições são isomórficas às cadeias de Markov (MOLLOY, 1981), deste modo, é possível gerar uma cadeia de Markov a partir de uma SPN que represente o mesmo sistema. Todavia, podemos nos deparar com um problema relacionado à explosão no espaço de estados, algo que pode vir a ser contornado por meio de técnicas de simulação.

2.4 DESEMPENHO E BENCHMARKING

A avaliação de desempenho trata do processo de medição e análise de um determinado sistema sob teste (Sistema Sob Teste (SUT)), objetivando sua respectiva compreensão e documentação com base em situações específicas (LILJA, 2000; JAIN, 1991).

Tanto as métricas de desempenho, quanto os atributos de dependabilidade podem ser alcançados de várias formas, duas delas são consideradas principais (JAIN, 1991; MENASCE et al., 2004): modelagem e experimentação (medição). A **medição**, também dita análise empírica, possui dentre suas principais vantagens, a alta precisão em seus resultados, bem como, a possibilidade de se avaliar uma série de fatores inerentes às arquiteturas como linguagens de programação, dispositivos de hardware, sistemas operacionais e alocação de memória (SIPSER, 2006).

Figura 8 – Processo de Avaliação Quantitativa de Sistemas



Fonte: (TRIVEDI; BOBBIO, 2017)

Todavia, só é possível realizar medição em ambientes já existentes, o que depende de uma série de fatores como a habilidade do programador, custo financeiro e ambientes heterogêneos. A Figura 8 apresenta um resumo das opções disponíveis quando precisamos realizar uma avaliação quantitativa de um sistema, seja esta através de medição ou modelagem.

Os benchmarks são um conjunto de ferramentas padrão utilizadas para comparar um sistema a outro ou com versões anteriores dele mesmo (HYPERLEDGER, 2018a). Um tipo específico de benchmark é o de desempenho. Os benchmarks desse tipo são utilizados para avaliação de desempenho e podem ter implementação informal, com base em *scripts* ou testes manuais, ou ainda fazer parte de um conjunto de padrões como Standard Performance Evaluation Corporation (SPEC) e Transaction Processing Performance Council (TPC). Geralmente, o processo de realização de benchmarks é mais controlado do que a avaliação de desempenho em si, sendo direcionado à métricas pré estabelecidas, e com uma carga de trabalho que pode tanto ser fixa quanto variável, mas sob controle daquele que a estabeleceu (HYPERLEDGER, 2018a).

2.5 ANÁLISE DE SENSIBILIDADE

O termo análise de sensibilidade se refere a um conjunto de métodos utilizados para identificação dos componentes de maior influência sobre uma métrica de interesse (FRANK, 1978). Com base nos resultados obtidos através de uma análise de sensibilidade é possível definir onde podemos, ou devemos, aplicar nossos esforços, para obtermos melhorias significativas nos resultados de uma métrica.

Existe uma série de técnicas que podem ser utilizadas para se fazer uma análise de sensibilidade, dentre elas estão: **Design of Experiments**, **Percentage Difference** e **Partial Derivatives**, explanadas a seguir.

O Design of Experiments (DoE) é uma técnica que permite a especificação do número adequado de experimentos/avaliações a serem realizadas dentro de um sistema ou modelo estocástico, com base em um número de fatores envolvidos (parâmetros) e seus respectivos níveis (valores). Definindo uma quantidade de parâmetros variáveis a serem alterados a cada nova etapa do experimento, por exemplo, valores de MTTF e MTTR do sistema operacional, podemos obter o máximo de informações possível sobre este sistema (JAIN, 1991; MONTGOMERY, 2006).

Um prático exemplo de um DoE é o *Full Factorial* (Fatorial Completo), este consiste na realização de todas as possíveis combinações entre fatores e níveis (JAIN, 1991). Isto nos permite encontrar os efeitos e impactos de todos os fatores e da interação entre eles na métrica de interesse. A Equação 2.13 apresenta como é realizada esta análise.

$$n = \prod_{i=1}^{k} n_i \tag{2.13}$$

Onde um número k de fatores, com o *i-ésimo* fator possuindo n_i níveis, deverá executar uma quantidade n de experimentos. Apesar de gerar todos os resultados possíveis, o fatorial completo possui, como desvantagem, o número de níveis para cada fator. Quanto maior o número de níveis de um fator, maior a demanda por recursos computacionais e maior o tempo de execução. Uma recomendação para reduzir esta desvantagem é a redução do número de níveis para cada fator, esta que pode ser feita através de uma análise minuciosa das métricas de interesse (JAIN, 1991).

Já a **Percentage Difference** ou diferença percentual é uma técnica de análise de sensibilidade considerada prática e de fácil implementação, podendo ser adaptada à linguagem de *scripts* da ferramenta Mercury (SILVA et al., 2015). A diferença percentual consiste na fixação de uma lista de parâmetros, enquanto um parâmetro específico tem seu valor variado, o que se repete até que todos os parâmetros tenham sido variados um a um. A Equação 2.14 mostra como o cálculo da análise de sensibilidade com diferença percentual é feito.

$$\text{SI} = (\frac{D_{max} - D_{min}}{D_{max}}), \tag{2.14}$$

Onde SI é o índice de sensibilidade para o parâmetro selecionado, D_{max} é o máximo valor que aquele parâmetro pode assumir e, D_{min} corresponde ao valor mínimo para aquele parâmetro.

Por fim, a análise de sensibilidade do tipo **Partial Derivatives**, é realizada computando-se as derivadas parciais para as métricas de interesse com respeito a cada parâmetro de entrada. Portanto, o índice de sensibilidade de uma medida Y, depende de um parâmetro λ específico. Este parâmetro é computado pelas Equações 2.15 e 2.16 para uma

sensitividade escalonada, também conhecida como elasticidade (WHITT, 2006).

$$\mathrm{S}_\theta(Y) = \frac{\partial Y}{\partial \theta} \tag{2.15}$$

$$\mathrm{SS}_\lambda(Y) = \frac{\lambda}{Y}\frac{\partial Y}{\partial \lambda} \tag{2.16}$$

Métodos específicos para realização da análise de sensibilidade em modelos analíticos são necessários quando queremos extrair fórmulas fechadas para computar medidas de interesse e encontrar a respectiva expressão de derivação.

2.6 CUSTOS

É comum associarmos o termo custo à questões financeiras inerentes à aquisição, distribuição, manutenção e contratação de um bem ou serviço (MEGLIORINI, 2001). No eixo da computação ainda podemos associar o termo custo a parte temporal da execução de alguma tarefa ou processo.

No geral, podemos definir o preço de um produto ou serviço com base nos valores gastos para sua fabricação, montagem, manutenção, instalação, transporte, além da margem de lucro e de uma série de outros fatores ligados a impostos e outras regionalidades (MEGLIORINI, 2001). No âmbito desta tese, consideramos o custo como uma relação entre o valor de aquisição de servidores e a manutenção do serviço em operação, ou seja, atividades de reparo e troca de componentes de hardware que apresentem falha. Além disso, acoplamos os custos energéticos referentes a sua respectiva operação e refrigeração. Claro, tais custos são inerentes ao de uma infraestrutura privada, criada e mantida unicamente pelo provedor do serviço.

Os custos de uma infraestrutura privada geralmente são superiores quando comparados aos de infraestruturas públicas(CHOPRA, 2017). Além dessa visível desvantagem, outras podem ser diretamente apontadas como a escalabilidade e a flexibilidade de infraestruturas públicas que podem representar a vida ou morte para companhias de *e-commerce*, por exemplo (PEREIRA et al., 2021). Todavia, nem apenas de desvantagens vive provimento de serviços em infraestruturas privadas, há também suas próprias vantagens associadas.

Customização, segurança, controle e custos previsíveis são quatro das principais vantagens dos ambientes privados (CHOPRA, 2017). Além disso, há a possibilidade de configurá-lo conforme o interesse da empresa, algo consideravelmente mais complexo quando lidamos com ambientes mais *gerais* como os fornecidos por infraestruturas de nuvem pública. Quanto aos custos previsíveis, podemos estimar de antemão o quanto iremos gastar em cada etapa do processo de provimento de serviço, ajustando a margem de lucro ou de economia as necessidades do empreendimento.

Resumimos a avaliação de custos da presente tese através da Expressão 2.17. É importante ressaltar que não consideramos um Custo Total de Posse (TCO) em sua completude.

As despesas associadas à construção do centro de dados em si como aquisição de terrenos, prédios, salas-cofre entre outros mecanismos subjetivos à localização, bem como, sua respectiva degradação ao passar dos anos não foram avaliadas (BARROSO; HÖLZLE; RANGANATHAN, 2018). Todavia, nosso Capex considera a aquisição de equipamentos e sua depreciação. Um outro aspecto importante é o das Despesas de Operação (Opex), que englobam o pagamento as equipes de manutenção, custos elétricos de operação e refrigeração da infraestrutura (BARROSO; HÖLZLE; RANGANATHAN, 2018).

$$\text{TCO} = \text{Capex} + \text{Opex} \tag{2.17}$$

2.6.1 Despesas de Capital

O Capex avaliado na presente tese considera unicamente o custo de aquisição de servidores. Esse custo é variável, e depende de uma série de fatores, como fabricante, modelo, capacidades e necessidades do serviço que está sendo provido. Podemos precisar de um número maior ou menor de servidores, e obviamente precisaremos de mais *switches*, roteadores, cabeamento, além de redundância de componentes, o que pode elevar consideravelmente nossos custos (BARROSO; HÖLZLE; RANGANATHAN, 2018). Além disso, conforme cresce a infraestrutura, cresce a necessidade de refrigeração dos componentes.

A depreciação associada aos servidores é variável. Muitos autores consideram a vida útil de um servidor como algo entre três e cinco anos (BARROSO; HÖLZLE; RANGANATHAN, 2018), mas optamos por deixar esta definição a cargo do avaliador que fará uso de nossos modelos e da ferramenta proposta. Além disso, no âmbito da presente tese a depreciação diz respeito a uma relação entre o custo para compra de um servidor e a sua respectiva vida útil, abstraindo assim o valor residual ou valor de venda do servidor após a passagem do tempo de vida útil estimado.

A Equação 2.18 apresenta o cálculo da depreciação anual para um ativo. Para o caso de maquinário, por exemplo, o que inclui servidores e computadores no geral, a Receita Federal do Brasil considera uma taxa de depreciação de 20% ao ano sob a perspectiva de uma vida útil de 5 anos, de acordo com a normativa de número 1700 de Março de 2017.

$$\text{Depreciação Anual} = \frac{\text{Taxa de Depreciação}}{\text{Vida Útil}} \tag{2.18}$$

Onde a Taxa de Depreciação corresponde ao quanto do equipamento desvaloriza anualmente e, a Vida Útil, ao quanto esperamos que o equipamento dure até estar totalmente desvalorizado.

A Equação 2.19 apresenta o cálculo da taxa de depreciação através do método de Saldo Decrescente Duplo (DDB). Este método calcula a depreciação acelerada. Ela é mais alta

para o primeiro ano e vai decrescendo nos anos que seguem[1].

$$\text{DDB} = (\text{Custo Original (\$)} - \text{Depreciação Acumulada (\$)}) \times \text{Taxa de Depreciação} \tag{2.19}$$

Já a Equação 2.20 apresenta o método de depreciação em linha reta ou direto. Este é o método considerado mais simples. O DDB considera a diferença entre o Custo Original do equipamento, ou seja, o seu valor de compra original e, sua Depreciação Acumulada, ou seja, o quanto o equipamento depreciou até aquele instante de tempo. Por fim, essa diferença é numeradora à Vida Útil restante, esta que representa a quantidade de tempo restante para aquele equipamento[2].

$$\text{Linha Reta} = \frac{(\text{Custo Original (\$)} - \text{Depreciação Acumulada (\$)})}{\text{Vida útil restante}} \tag{2.20}$$

Por fim, estabelecemos a depreciação anual através de uma função que relaciona o valor máximo entre DDB e a Linha Reta: $\text{Max}(\text{DDB}, \text{Linha Reta})$. Por exemplo, um servidor que custa \$1.200,00, tem uma vida útil estimada em três anos, e uma taxa de depreciação de 0,67 (Equação 2.18), terá sua depreciação anual calculada como apresentado na Tabela 2[3].

Tabela 2 – Exemplo de Depreciação

Ano	**DDB**	**Linha Reta**	**Depreciação Anual(\$) Max(DDB,Linha Reta)**	**Depreciação Mensal(\$)**
1	(1200-0) × 0,67	(1200-0)/3	804,00	67,00
2	(1200-804) × 0,67	(1200-804)/2	265,32	22,11
3	(1200-1069,32) × 0,67	(1200-1069,32)/1	130,68	10,89

Fonte: (BARROSO; HÖLZLE; RANGANATHAN, 2018)

A primeira coluna da tabela apresenta o período de tempo avaliado, neste caso de 1 a 3 anos. A segunda mostra o cálculo da depreciação através do método DDB e a terceira o mesmo cálculo mas através do método de linha reta. O maior valor ou valor máximo obtido comparando-se ambos os métodos de depreciação. Por fim, estabelecemos uma depreciação mensal para cada ano com base no valor total depreciado para o período.

2.6.2 Despesas Operacionais

As Opex são caracterizadas pelo custo energético e o custo para operações de reparo de servidores inerentes a equipe de manutenção (BARROSO; HÖLZLE; RANGANATHAN, 2018).

[1] <https://support.microsoft.com/pt-br/office/bdd-fun>
[2] <https://www.portal-gestao.com/artigos/7984-f>
[3] <https://bit.ly/3ynvWDM>

2.6.2.1 Custo Energético

Iniciamos pelos custos energéticos, cujo valor dependerá da Potência (W) e do tempo requerido de operação de cada servidor. Além disso, o custo do kilowatt hora (kwh) na região de implantação do serviço pode variar, estando diretamente associado ao tipo de energia utilizada. Um conjunto de outros fatores regionais também podem estar relacionados ao estabelecimento de custo, como impostos, tributos e taxas extras para horários de pico.

Além do custo com operação de servidores há também os custos com os componentes de refrigeração. Esse valor pode ser maior ou igual ao custo energético para o maquinário, e está diretamente relacionado ao tipo de refrigeração utilizada. Se a refrigeração nativa do *rack* em que os servidores estão dispostos for suficiente para mantê-los operacionais e sem degradação acelerada dos componentes haverá uma redução dos custos, mas, geralmente, essa não é a realidade, sendo necessária a aquisição e uso de ar-condicionado ao longo de todo período de operação dos servidores. Para esta tese, seguindo orientação apresentada em (CHOPRA, 2017), optamos por igualar o consumo energético por parte dos servidores com o consumo dos dispositivos de refrigeração, ou seja, para cada real gasto com energia de um servidor, gastaremos um outro real em refrigeração.

O custo do kilowatt hora pode ser encontrado na conta de luz e é um dado importante para o cálculo do custo energético. Podemos calcular kilowatt hora para uma infraestrutura considerando a expressão 2.21[4]. É possível, também, obtermos o custo energético total para infraestrutura ao multiplicarmos o valor obtido pela referida expressão pelo respectivo valor do kWh na região de implantação da infraestrutura.

$$\text{kWh} = \frac{(\#\text{ Watts}) \times (\#\text{ de horas por dia}) \times (\#\text{ de dias por ano})}{1000} \tag{2.21}$$

onde kWh é acrônimo para kilowatt hora, e o símbolo # é indicativo de quantidade, ou seja, quantidade de Watts, quantidade de horas por dia e quantidade de dias por ano que a respectiva infraestrutura está operação.

2.6.2.2 Custo de Manutenção

Assim como os demais custos, o custo de manutenção é variável, e dependerá não somente da quantidade de pessoas trabalhando, mas de onde elas vêm, se fazem parte ou não no quadro de funcionários da empresa. De modo geral, para uma empresa que possua funcionários específicos para operação dos componentes que compõem a infraestrutura, cerca de 5% do valor de Capex são gastos com manutenção (BARROSO; HÖLZLE; RANGANATHAN, 2018).

$$\#\text{Total de Manutenções} = \sum_{i=1}^{k} \frac{\text{Período}}{(\text{MTTF} + \text{MTTR})_i} \tag{2.22}$$

[4] <https://bit.ly/3u1pFue>

Em cenários onde o número de servidores seja reduzido, o número de funcionários também será. Além disso, na maioria das vezes esses 5% não representarão a realidade, e o pagamento ao nosso funcionário será, por vezes, maior que o custo de aquisição de novos servidores (SAHA et al., 2016). Isso pode nos levar à alternativas mais práticas, como a contratação de *freelancers* e empresas terceirizadas especializadas em manutenção de computadores.

Acordos de nível de serviço podem ser firmados também em infraestruturas privadas, afinal de contas, trata-se de um serviço que está sendo provido (PATEL; RANABAHU; SHETH, 2009). O tempo necessário para troca dos componentes é relevante e o custo dessa troca deve ser pensado no processo de estimativa de vida útil daquele equipamento. No caso de software e aplicativos, muitos problemas podem ser solucionados apenas com o reinício de servidores e aplicações. Apresentando, assim, um baixo custo quando comparamos seus problemas aos problemas de falha de hardware, porém, falhas de software são, em tese, muito mais frequentes que as de hardware.

2.7 BLOCKCHAIN

As tecnologias baseadas em blockchain são inseridas no contexto de Tecnologias de Registro Distribuído (DLT), juntamente com outras tecnologias como a Byteball[5] e Nano[6] que não são contempladas por esta tese, mas que também são sistemas distribuídos para o armazenamento de arquivos e dados.

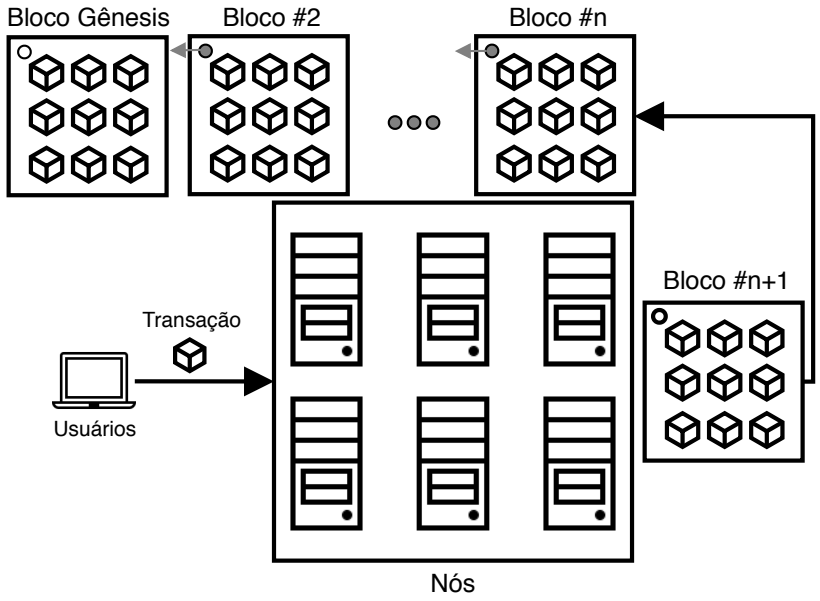
Em redes blockchain, dá-se o nome de registros aos itens que estão compartilhados entre os nós que compõe a rede. Tais registros são utilizados no processo de armazenamento de um conjunto de transações em entidades chamadas *blocos*. Esses blocos são inseridos um após o outro na forma de lista ligada dentro de uma arquitetura *peer-to-peer* (GUPTA, 2017). Cada bloco, além de armazenar dados acerca das transações realizadas, também possui um ponteiro para o bloco imediatamente anterior. A Figura 9 mostra o funcionamento de uma blockchain típica.

O passo-a-passo que descreve uma transação entre usuários e o caminho de um bloco dentro de uma rede blockchain são apresentados a seguir:

- Um determinado **Usuário** inicia uma requisição para uma rede blockchain;
- Essa requisição é convertida em uma transação a ser validada pelos componentes que integram a rede em um sistema de consenso. Esses componentes são computadores comuns que recebem o nome de nós em uma arquitetura P2P;
- Após a validação de uma transação ela é inserida em um bloco junto de outras transações;

[5] Byteball: https://obyte.org/
[6] Nano: https://nano.org/en

Figura 9 – Funcionamento de uma Blockchain



Fonte: (GUPTA, 2017)

- Esse bloco, contendo um conjunto de transações e informações acerca de seu antecessor é então inserido na cadeia de blocos;
- A transação é então concluída.

2.7.1 Tipos de Blockchains

Análogo aos tipos de ambientes de computação em nuvem, as blockchains podem ser divididas em três subcategorias distintas: públicas, permissionadas e privadas. Esta divisão se dá de acordo com os seguintes critérios: (i) quais clientes podem submeter transações? (ii) quais os nós que podem realizar a validação das transações? e (iii) como os clientes se juntam a rede?

Nas blockchains públicas qualquer indivíduo ou máquina pode participar do processo de validação de transações, não sendo necessária a sua respectiva identificação. Este tipo de blockchain geralmente envolve incentivos econômicos e fazem uso de um sistema baseado em Prova de Trabalho (PoW). Este que é o caso das redes Ethereum (BUTERIN, 2016) e Bitcoin (NAKAMOTO et al., 2008).

Já as redes permissionadas são operadas por entidades conhecidas, todos os membros possuem uma identidade, sejam nós ou usuários e cada um pode realizar transações específicas ao seu contexto de negócio. Nas redes permissionadas, novos nós só são adicionados com a devida permissão de um regulador, geralmente não sendo necessário o incentivo econômico aos nós que cedem seu respectivo poder computacional para conclusão de seus objetivos.

Por fim, as redes blockchain privadas dizem respeito a um tipo específico de redes blockchain permissionadas. A principal diferença entre uma blockchain privada e uma permissionada está em sua operação. Enquanto que em redes permissionadas temos varias entidades ou organizações comunicantes que a operam e mantém, em redes privadas há apenas uma única entidade.

Quanto mais nós em uma rede blockchain, maior será a disponibilidade dos registros, em contrapartida, menor tende a ser o respectivo desempenho desta rede. Na rede da criptomoeda bitcoin, por exemplo, no melhor cenário possível o acréscimo de um novo nó a rede não reduzirá em nada o seu desempenho[7]. Porém, em computação, é *normal* sempre trabalharmos visando o pior caso (SIPSER, 2006).

2.7.2 Características de uma Blockchain

A seguir apresentamos algumas das principais características de uma blockchain.

Endossamento é a simulação da transação por parte de um nó. Este que realiza a execução do contrato inteligente em conjunto dos parâmetros de entrada para transação, e após a verificação e obtenção de um resultado positivo atribui uma assinatura própria e devolve a transação ao cliente (SUKHWANI et al., 2018);

Consenso é o processo distribuído pelo qual uma rede de nós provê uma ordem única para execução das transações realizando a sua respectiva validação, através de sua inserção em blocos (HYPERLEDGER, 2018c). O processo de consenso pode ser implementado de diferentes maneiras, como através de algoritmos baseados em loteria (lottery-based algorithms) como o PoW utilizado pela rede Bitcoin, ou através de algoritmos baseados em votação (Voting-based algorithm) como o Protocolo Prático de Consenso Tolerante a Falhas Bizantinas (pBFT) e o Tolerância Redundante a Falhas Bizantinas (RBFT).

Algoritmos baseados em loteria são mais vantajosos em termos de escalabilidade, ou seja, quando o número de nós tende a crescer. Esses algoritmos baseiam-se na escolha de um "vencedor", este que encontrou um novo bloco e este bloco é transmitido à rede para validação (HYPERLEDGER, 2018c). Todavia, esse tipo de algoritmo pode promover bifurcações, caso dois ou mais nós encontrem o mesmo bloco simultaneamente, resultando na necessidade de solução de bifurcação pela rede e levando a um tempo maior para conclusão das transações.

Já os **algoritmos baseados em votação**, geralmente levam um tempo menor à conclusão de transações, basta apenas que a maioria dos nós valide a transação ou bloco para que haja consenso da rede (HYPERLEDGER, 2018c). Porém, quanto maior

[7] Hyperledger Performance Metrics: https://www.hyperledger.org/resources/publications/blockchain-performance-metrics

o número de nós na rede maior o tempo necessário à realização do consenso. Neste caso, trocamos o desempenho pela escalabilidade.

As características importantes a qualquer sistema de consenso são a garantia da **segurança** (*safety*) e da **vivacidade** (*liveness*) (HYPERLEDGER, 2018c). A segurança em termos de garantia de que todos os nós da rede recebam a mesma entrada e produzam a mesma saída e; a vivacidade, na garantia de que todos os nós que não estejam em estado de falha recebam todas as transações submetidas ao sistema.

Commit é o mecanismo responsável por escrever uma transação em uma base de dados, que em redes blockchains necessita especificar quando isto deve acontecer em meio a múltiplos nós (HYPERLEDGER, 2018c). Tipicamente um bloco é considerado um registro de transações *commitadas* e essas transações só são *commitadas* quando seu respectivo bloco atravessar toda a rede e estiver presente em todos os nós que a compõe.

Finality é a propriedade que garante que, após o *commit* de uma transação, a mesma não mais poderá ser revertida, ou seja, seus dados não podem mais voltar ao estado anterior (HYPERLEDGER, 2018c).

O **Tamanho da Rede** diz respeito ao número de nós que realizam validação, ou seja, participando do processo de consenso (HYPERLEDGER, 2018c). Esta conta não inclui nós observadores ou outros nós que não participem do processo de validação. Uma rede blockchain, com apenas um nó de consenso, possui tamanho um e, em virtude disso, possui ponto único de falha.

Query é a habilidade de realizar operações ad-hoc ou buscas dentro de um conjunto de dados pertencentes a uma blockchain (HYPERLEDGER, 2018c). Já as **Leituras** são operações resultantes de uma query que se diferenciam de transações por não gerar uma mudança no estado da blockchain. Esta que, essencialmente, pode ser vista como uma máquina de estados. Nessa máquina de estados, cada transação ou bloco de transações corresponde a uma transição de mudança de estado. Este que, por sua vez, trata-se do conteúdo da cadeia em determinado instante de tempo.

Transação é o mecanismo de modificação de dados em uma rede blockchain, sendo tipicamente proposta por clientes e então analisada, validada e com seu devido *commit* dado pela rede (HYPERLEDGER, 2018c);

Smart Contract é um programa de computador que executa sobre uma rede blockchain (HYPERLEDGER, 2018d). Smart contracts podem ser essencialmente simples como um processo de atualização ou extremamente complexos como contratos reais dotados de toda uma lógica de negócio. Existem dois tipos principais de smart contracts:

O **Smart contract instalado** é aquele onde a lógica de negócio é instalada sob os nós responsáveis pela validação das transações e devem ser instalados antes da rede ser iniciada (HYPERLEDGER, 2018d);

Já os **Smart contracts na cadeia** são aqueles onde o código que define a lógica de negócio faz parte do registro compartilhado e é executado como uma transação que executa outras transações quando determinado requisito for alcançado (HYPERLEDGER, 2018d).

2.7.3 Hyperledger Fabric

O Hyperledger Fabric é uma plataforma para construção de soluções baseadas em registros compartilhados e distribuídos. Esta plataforma é de implementação *open-source*. O HLF faz uso de tecnologias baseadas em contêineres para operar *smart contracts*. Esses que no âmbito do Fabric recebem o nome de *chaincodes* e são do tipo instalado, necessitando de configuração prévia na rede antes da mesma ser criada (HYPERLEDGER, 2018d).

Os *chaincodes* também são responsáveis por gerir as regras de negócio que determinam o funcionamento da rede blockchain e podem ser escritos por padrão em Go e Javascript. Mas, podendo ser readaptados a outras linguagens de programação como Python e Java, através da devida implementação de sua interface. Existem dois tipos de chaincodes:

Os **chaincodes de sistema** tipicamente gerenciam as transações relacionadas ao sistema, como a gerência do ciclo de vida das transações e a política de configuração (HYPERLEDGER, 2018d);

Já os **chaincodes de aplicação** são responsáveis pela gerência dos estados da aplicação e dos registros compartilhados, incluindo bens ou dados arbitrários (HYPERLEDGER, 2018d).

O HLF pode trabalhar em cima de ambientes permissionados de blockchain que, como mencionado na subseção 2.7.2, pode apresentar um desempenho superior ao de um ambiente público no processo de consenso e validação de transações, como ocorre nas redes blockchain de algumas criptomoedas, como o Bitcoin (SUKHWANI et al., 2017).

No âmbito desta tese escolhemos a plataforma Hyperledger Fabric para avaliação por: (i) se tratar de uma das principais plataformas para provimento e gerência de blockchains, o que pode ser visto na Tabela 3; (ii) ser *open source*; (iii) ser permissionada, o que dá maior controle sobre o ambiente que será avaliado.

A plataforma Hyperledger Fabric é disposta em três contêineres que são responsáveis pelo processo de provimento de um serviço ou aplicação baseada em blockchain. Essa aplicação está ligada à uma rede que necessita ter seu contrato previamente implantado nesses contêineres.

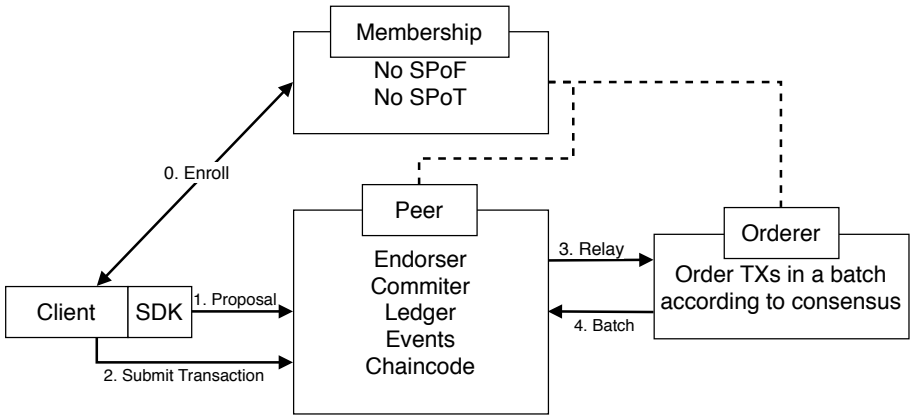
Tabela 3 – Onde encontro o Hyperledger Fabric?

Provedor de BaaS	Ethereum	Quorum	Corda	Hyperledger Fabric	MultiChain	Digital Asset
AWS	✓	✓	✓	✓		
Azure	✓	✓	✓	✓	✓	
Google	✓			✓		✓
HPE			✓			
IBM				✓		
Oracle				✓		
SAP				✓	✓	

Fonte: Ledger Insights

A Figura 10 apresenta como o cliente e o provedor de serviço se comunicam nesta plataforma, bem como os passos que devem ser seguidos até que uma determinada transação seja inteiramente aceita pelo sistema (HYPERLEDGER, 2018b; HYPERLEDGER, 2018c).

Figura 10 – Visão Geral do Hyperledger Fabric



Fonte: (HYPERLEDGER, 2018b)

O cliente conecta-se (0) ao sistema através do serviço de filiação ofertado pelo contêiner **Membership** Service Provider (MSP). O MSP verifica as credenciais deste usuário e provê o seu respectivo acesso ao sistema e a possibilidade de submissão de transações. Após o acesso, o cliente poderá propor (1) transações ao nó *Peer*. Existem muitos tipos de peers e nosso foco são os peers de endossamento (*endorsers*). Peers de endossamento simulam uma transação e a enviam de volta ao cliente com uma assinatura que certifica a veracidade daquela transação, implicando no fato de que a mesma poderá ser efetuada pelo sistema.

Como exemplo de uma transação, podemos citar a transferência de bens (*assets*) entre dois clientes de um banco. Ambos os clientes precisam existir no sistema, e aquele que é dito emissor do valor deverá ter em sua conta valor igual ou maior que o que ele deseja transferir, descontando-se alguma taxa estabelecida pela rede em contrato. Taxa esta que

no âmbito de redes permissionadas, como já mencionado, é inexistente ou deveria ser irrelevante já que todas as partes presentes na rede possuem algum interesse na realização desta operação.

O endossamento de uma transação é baseado em um conjunto de políticas descritas no código fonte daquela blockchain ou em seu *smart contract*. As políticas especificam quantos e quais nós ou peers precisam concordar ou atestar através de sua assinatura a veracidade daquela transação. As três principais políticas de endossamento são AND, OR e K-out-of-N (KooN).

Supondo que tenhamos três servidores a disposição, ou seja, do lado do provedor do serviço, cada um hospedando um único peer de endossamento. Se a política de endossamento do tipo AND for adotada, então todos os três nós precisarão atestar uma transação que chegue ao sistema e enviá-la de volta ao cliente que a submeteu. No caso de uma política de endossamento do tipo OR, pelo menos um dos três nós precisaria atestar a autenticidade desta transação. O mesmo aplica-se à uma política de endossamento do tipo KooN, onde nós determinamos quantos peers disponíveis precisam assinar a respectiva transação, por exemplo 1-out-of-3, 2-out-of-3, ou até mesmo 3-out-of-3.

Após a realização do endossamento, a transação é submetida ao contêiner de Ordenação (*orderer*). O orderer é responsável por juntar todos as transações submetidas ao sistema em um *batch* ou bloco, e o envia aos peers. Os peers realizam o *commit* do novo bloco ao registro distribuído de transações com base nos requisitos pré-definidos da blockchain.

2.8 HYPERLEDGER CALIPER

O desempenho é um dos principais questionamentos acerca da transição de serviços hospedados em ambientes centralizados para blockchains. Será realmente viável trocar o certo pelo duvidoso? O desempenho de serviços de pagamento ou de transferência de dados e informações em ambientes centralizados é amplamente conhecido. Todavia, estudos acadêmicos sobre o desempenho de ambientes de blockchain para esse propósito tiveram início apenas em 2014.

O Hyperledger Caliper é uma ferramenta para realização de benchmark em plataformas blockchain desenvolvido pelo consórcio Hyperledger, é resultado de mais uma parceria com a Linux Foundation. O Caliper permite a seus usuários medir o desempenho de uma implementação específica de uma blockchain com um conjunto predefinido de estudos de caso. Os relatórios produzidos pelo Caliper possuem uma série de indicadores de desempenho, tais como Transactions per Second (TPS), latência, utilização de recursos por parte dos contêineres.

Uma das principais características do Caliper é sua capacidade de adaptação que permite ao usuário comparar o desempenho de várias implementações distintas de blockchain e auxiliar a definir qual a melhor delas, de acordo com seus interesses.

2.9 CONSIDERAÇÕES FINAIS

Esta seção apresentou a fundamentação teórica que possibilita a compreensão de alguns dos principais temas, mecanismos e técnicas de avaliação abordadas por esta tese. Apresentamos conceitos relacionados a custos, blockchain, além da avaliação de um dos principais atributos de dependabilidade de sistemas computacionais através de modelagem.

3 TRABALHOS RELACIONADOS

> *Mistakes, are also important to me. I don't cross them out of my life, or memory. And I never blame others for them.*
>
> — Andrzej Sapkowski, *Blood of Elves*, 2008

Esta seção apresenta uma síntese dos principais trabalhos relacionados à pesquisa realizada nesta tese. Os trabalhos selecionados foram categorizados de acordo com o seu tema central. Sumarizamos esses trabalhos ao término desta seção através de uma tabela que os compara com o que é proposto pela presente tese. A busca realizada no processo de levantamento bibliográfico foi associada à modelagem, disponibilidade, confiabilidade e desempenho de plataformas baseadas em blockchain.

3.1 DESEMPENHO

A maioria dos trabalhos científicos encontrados ao longo de nossa pesquisa tem por foco métricas de desempenho. Em (ONIK; MIRAZ, 2019) os autores realizaram um survey e sumarizaram o estado da arte das tecnologias e plataformas necessárias ao provimento de Blockchain como Serviço (BaaS) em ambientes de computação em nuvem. Como resultado, foi provido um guia capaz de direcionar administradores e interessados nessas tecnologias através de um comparativo entre as principais provedoras de BaaS (Amazon AWS, Azure, Google, HPE, Oracle e SAP) e as tecnologias por elas utilizadas. A partir deste trabalho, ficou evidenciada a necessidade de se comparar os custos de implantação, manutenção e operação de infraestruturas privadas e públicas capazes de hospedar aplicações baseadas em blockchain. Por tratar-se de um trabalho inteiramente teórico, as diferenças entre ele e a presente tese vão além do escopo conceitual, passando por objetivo geral e resultados obtidos.

Enquanto que em (SUKHWANI et al., 2017) os autores analisaram o pBFT, algoritmo de consenso utilizado pela plataforma de blockchain Hyperledger Fabric em sua Versão 0.6, e determinaram se este poderia ou não ser um gargalo no desempenho da plataforma. Para esta avaliação, utilizou-se modelagem através do formalismo Stochastic Reward Nets (SRN), onde computou-se os tempos médios necessários À finalização do processo de consenso na plataforma avaliada. Este foi o primeiro trabalho a utilizar modelos analíticos para avaliação de uma plataforma de blockchain. Mas, o seu foco foi integralmente o desempenho da plataforma em termos de endossamento. Diferentemente da presente tese, onde além da avaliação de desempenho focada na vazão e latência do sistema, também, apontamos dados relacionados ao consumo de recursos, entre os quais estão Disco, RAM e CPU.

Em trabalho subsequente, Sukhwani et al. (SUKHWANI et al., 2018) foram além do algoritmo de consenso e estenderam a avaliação de desempenho para toda a plataforma Hyperledger Fabric através de modelos SRN. A versão 1.0 da plataforma foi o foco do estudo. Mas, o trabalho limitou-se a métricas de vazão e latência, ambas medidas através da execução da ferramenta de benchmark Hyperledger Caliper. A presente tese também avalia essas métricas, como já mencionado, através da aplicação do Hyperledger Caliper, todavia apresentamos detalhes de consumo de recursos e o impacto desse consumo sobre outras métricas, como a disponibilidade geral do sistema.

Já em (LI; MA; CHANG, 2018), os autores propuseram um processo markoviano para expressar a relação entre a taxa de chegada e a capacidade do servidor em atender as respectivas transações em uma rede blockchain. O foco único desse trabalho foi a representação da blockchain do Bitcoin através de um modelo genérico capaz de representar o número de transações por bloco, o seu tamanho, entre outras importantes informações correlatas ao desempenho. Já a presente tese avalia o desempenho em termos de recursos consumidos, vazão e latência de transações no âmbito do Hyperledger Fabric em sua versão 1.4.1.

Em (ZHENG et al., 2018), os autores utilizam CTMCs para simular o tempo de resposta de um sistema de consenso baseado no protocolo pBFT. O foco dos autores foi uma aplicação específica, um sistema de *Healthcare* que faz uso de tecnologia baseada em blockchain. O trabalho diferencia-se de (SUKHWANI et al., 2017) por tratar de um cenário mais específico de aplicação do algoritmo de consenso e de se utilizar de cadeias de Markov para avaliação de desempenho. Isso permitiu aos autores a extração e generalização do modelo em expressões matemáticas. O tempo de resposta foi a principal métrica de interesse avaliada. Mais uma vez, esta tese não apresenta modelos de desempenho, todavia avaliamos outras importantes características de desempenho não contempladas pelo trabalho de (ZHENG et al., 2018), como vazão, latência e consumo de recursos.

Em (JIANG; FANG; WANG, 2018), os autores avaliaram o uso de blockchain em redes veiculares. E para determinar sua viabilidade, testes de desempenho foram realizados. Os autores utilizaram modelos teóricos que representariam uma plataforma de blockchain e de modelos de simulação correspondentes que foram submetidos a uma série de cenários pré-estabelecidos. A latência foi a principal métrica de desempenho avaliada, na presente tese avaliamos a latência através de medição, e focamos em uma aplicação básica provida pelos desenvolvedores do benchmark Hyperledger Caliper, esta aplicação permitia a criação e consultas, bem como a transferência de bens entre contas.

Já em (ALASLANI; NAWAB; SHIHADA, 2019), os autores avaliaram o desempenho de uma rede blockchain considerando dispositivos de Internet das Coisas (IoT) e o impacto do atraso da rede sobre o mesmo; o sistema utilizado mais uma vez foi o Hyperledger Fabric e expressões matemáticas foram utilizadas como meio analítico para avaliação de desempenho. Mais uma vez o consumo de recursos computacionais foi ignorado, além

disso, a aplicação avaliada difere da mensurada nesta tese, mesmo que a plataforma de blockchain tenha sido a mesma.

Enquanto que em (KOUSHIK et al., 2019), o desempenho da plataforma Hyperledger Fabric foi novamente avaliado através de medição e com o foco em uma aplicação de registros médicos. A vazão e a latência se destacam como as métricas escolhidas pelos autores. Os recursos gerais foram novamente deixados de fora. Além disso, a aplicação diferiu da que teve o seu desempenho avaliado no âmbito da presente tese.

Em (MISIC et al., 2019) os autores proveem um modelo de fila do tipo M/G/1 para representar a rede Bitcoin. Este modelo considera o sistema como uma rede do tipo Jacksoniana, ou seja, a taxa de chegada segue um processo de Poisson e o tempo de serviço é exponencialmente distribuído. O tempo de resposta do nó e o tempo necessário para conclusão de transações são considerados, o que inclui o tempo de resposta e a vazão de transações. A plataforma avaliada pelos autores difere da avaliada por esta tese, assim como o tipo de modelo proposto e as métricas de interesse.

Os autores em (ROEHRS et al., 2019) propuseram e avaliaram a utilização da plataforma de blockchain omniPHR voltada ao armazenamento de dados médicos de pacientes em um hospital. Eles utilizaram expressões e medições para realizar a avaliação de desempenho desta plataforma. As métricas de interesse foram a vazão e a latência do sistema, desconsiderando a medição e avaliação do consumo de recursos por parte do servidor.

Já em (KHAN; ARSHAD; KHAN, 2020), os autores avaliaram a Multichain, uma plataforma capaz de trabalhar e interagir com outras plataformas distintas. O foco dos autores foram métricas de desempenho, mais especificamente a vazão do sistema em transações por segundo e a latência na realização dessas transações. Nenhum tipo de modelo foi utilizado neste trabalho, apenas a técnica de medição foi aplicada. Na presente tese, também realizamos a avaliação de desempenho através de medição, e consideramos o impacto do consumo de recursos sobre a disponibilidade do serviço.

Os autores em (KRISHNASWAMY, 2020) representam mais um grupo a avaliar o Hyperledger Fabric e o seu respectivo desempenho através de um modelo de simulação. Expressões que poderiam representar o funcionamento da plataforma e seu sistema de consenso foram derivadas e embasam o modelo de simulação proposto. O foco foram aplicações na borda da rede a executar em dispositivos genéricos, as métricas de desempenho avaliadas foram, novamente, a vazão em transações por segundo e latência do sistema, ambas as métricas também foram avaliadas nesta tese, mas, diferindo na aplicação avaliada e nos recursos disponíveis para realização do experimento.

Por fim, em (XIONG et al., 2019) os autores consideraram gerenciamento de recursos em ambientes de computação em nuvem e *fog computing*. O objetivo dos autores era o estabelecimento de uma relação custo-benefício na realização de mineração em redes blockchain, considerando uma abordagem baseada em *offloading* e teoria dos jogos. Expressões matemáticas foram utilizadas como forma analítica de avaliação de desempenho

do sistema e comparação com valores obtidos através de medição em cenários reais. A principal diferença entre esse trabalho e o proposto por esta tese quando tratamos de desempenho é o contexto de blockchain como serviço.

Em (BRINCKMAN et al., 2017), os autores avaliaram questões relacionadas à segurança no compartilhamento de contêineres em ambientes de computação em nuvem utilizando blockchain para assegurar sua confidencialidade. Esse trabalho utilizou a plataforma Hyperledger Fabric e tratou de implementação e simulação de um ambiente. Os autores também mensuraram o desempenho da plataforma, conforme verificavam a melhoria das políticas de segurança presentes no ambiente. No âmbito da presente tese, não lidamos com a questão da segurança em ambientes de blockchain. Todavia, ampliamos o nosso estudo a disponibilidade e confiabilidade desses sistemas atendendo, assim, um número maior de itens da árvore de dependabilidade.

3.2 DEPENDABILIDADE

Em (LIU et al., 2018), os autores avaliaram a confiabilidade de uma blockchain para Internet das Coisas (IoT). Para tanto, uma Cadeia de Markov de Tempo Contínuo CTMC foi proposta, tal cadeia considera o algoritmo de consenso utilizado pela plataforma, o número de dispositivos conectados à rede e, sua respectiva confiabilidade. Com base nessas informações, foi possível estabelecer o impacto de cada fator sobre a confiabilidade geral do sistema. Nenhuma técnica de análise de sensibilidade formal foi utilizada para determinação de fatores que mais impactam na confiabilidade geral do sistema. Além disso, a disponibilidade não foi avaliada, assim como o impacto do consumo de recursos sobre este importante atributo da dependabilidade de sistemas computacionais.

Já em (KANCHARLA et al., 2019), os autores propõem um *framework* para assegurar a confiabilidade de sistemas baseados em blockchains. Os autores realizaram um estudo teórico que fez uso de equações que denotam o comportamento do sistema em uma tentativa de averiguar os resultados obtidos e a viabilidade do *framework* proposto. A ferramenta proposta nesta tese teve como inspiração o framework proposto por (KANCHARLA et al., 2019). Porém, a base de nossa ferramenta são modelos analíticos, além de contemplar um número maior de soluções e assumir o papel de auxiliar na tomada de decisão.

Em (WEBER et al., 2017), os autores identificaram limitações na disponibilidade de duas das principais plataformas baseadas em blockchain: Ethereum e Bitcoin. Os autores não avaliaram questões relacionadas a ambientes virtualizados e sua respectiva avaliação não considerou a utilização de modelos analíticos, limitando-se apenas a operação das transações no sistema. Sua análise se deu através de *logs* públicos disponíveis para ambas as plataformas e foi compreendida como um sumário estatístico com base na ocorrência de picos em alguns períodos específicos ao longo do ano. Já a presente tese avalia a disponibilidade do Hyperledger Fabric e sua implantação em infraestruturas privadas. Já a nossa avaliação ocorreu através de um conjunto de modelos. Além disso, apontamos o

impacto do consumo de recursos sobre disponibilidade e confiabilidade de uma aplicação teste sob distintas políticas de endossamento.

Em (KANCHARLA et al., 2020), os autores avaliaram a dependabilidade de modo geral sob a perspectiva de operacionalidade e escalabilidade da IPFS (Inter-Planetary File System), sob um modelo teórico por eles proposto intitulado de Slim Chain. Este modelo é capaz de avaliar e determinar a probabilidade da escrita e leitura de uma dada transação dentro e fora da aplicação Blockchain com base em simulação. Tanto a plataforma avaliada quanto os objetivos de avaliação diferem dos da presente tese. Além disso, avaliamos atributos de dependabilidade de modo geral, sob a perspectiva do provedor de serviço, e não da probabilidade de sucesso ou falha de uma operação na rede.

Em trabalho subsequente, (KANCHARLA et al., 2020) avaliaram a dependabilidade sob a perspectiva de uma blockchain híbrida, investigando a relação entre uma rede blockchain privada e uma pública. Sua avaliação uma vez mais consistiu na perspectiva da transação, sobre a probabilidade da mesma ser escrita com sucesso ou não dentro e fora da blockchain pública e em determinar sua privacidade mesmo sob a interação entre duas redes distintas. As semelhanças entre esta tese e esse trabalho estão apenas no âmbito da infraestrutura privada. Todavia, a plataforma avaliada não foi a mesma, nem os atributos de dependabilidade e suas perspectivas.

3.3 VISÃO GERAL

A Tabela 4 apresenta uma comparação entre os trabalhos relacionados e a presente tese em termos de principais contribuições.

Como podemos ver, esta tese diferencia-se dos demais trabalhos previamente apresentados em muitas formas. É a única a avaliar os custos de implantação e manutenção, disponibilidade, confiabilidade e desempenho de infraestruturas capazes de hospedar serviços e aplicações baseadas em blockchain sob a perspectiva do provedor do serviço. Também apresentamos a avaliação de ambientes baseados no Hyperledger Fabric através de modelagem analítica e hierárquica. Além disso, apontamos os possíveis pontos de gargalo através de análise de sensibilidade, com base nos modelos de disponibilidade propostos desenvolvemos um framework completo capaz de auxiliar a tomada de decisão. O que, até o presente momento, não foi considerado por nenhum outro trabalho. Por fim, ainda avaliamos o desempenho de uma infraestrutura contendo os requisitos mínimos para execução de uma aplicação básica através de um benchmark, apresentando o impacto do consumo de recursos sobre a disponibilidade e confiabilidade geral de vários ambientes com base em políticas de endossamento distintas.

Também podemos concluir que a grande maioria dos trabalhos relacionados focaram no desempenho de plataformas permissionadas, como o Hyperledger Fabric. Esta avaliação é mais tangível, haja vista que blockchains públicas possuem dados públicos sobre o seu desempenho e experimentos com a mesma não seriam factíveis. Uma outra importante

Tabela 4 – Comparação entre os trabalhos relacionados considerados mais relevantes

Autores	**Tipo de Blockchain**	**Desempenho**	**Disponibilidade**	**Confiabilidade**	**Custo**	**Modelos**
(WEBER et al., 2017)	Pública		✓			
(BRINCKMAN et al., 2017)	Permissionada	✓				
(SUKHWANI et al., 2017)	Permissionada	✓				✓
(SUKHWANI et al., 2018)	Permissionada	✓				✓
(ZHENG et al., 2018)	Permissionada	✓				✓
(LI; MA; CHANG, 2018)	Pública	✓		✓		✓
(LIU et al., 2018)	Permissionada			✓		✓
(JIANG; FANG; WANG, 2018)	Permissionada					✓
(XIONG et al., 2019)	Pública e Permissionada	✓			✓	✓
(ALASLANI; NAWAB; SHIHADA, 2019)	Permissionada	✓				✓
(ONIK; MIRAZ, 2019)	Pública e Permissionada	✓				
(MISIC et al., 2019)	Pública	✓				✓
(ROEHRS et al., 2019)	Permissionada	✓				✓
(KOUSHIK et al., 2019)	Permissionada	✓				
(KANCHARLA et al., 2019)	Permissionada			✓		✓
(KHAN; ARSHAD; KHAN, 2020)	Pública	✓				
(KRISHNASWAMY, 2020)	Permissionada	✓				✓
(KANCHARLA et al., 2020)	Pública		✓	✓		✓
(KANCHARLA et al., 2020)	Pública e Permissionada		✓	✓		✓
(ALTARAWNEH et al., 2021)	Permissionada		✓	✓		✓
(KASAHARA, 2021)	Pública	✓			✓	
Esta Tese	Permissionada	✓	✓	✓	✓	✓

Fonte: Autoria Própria

característica é a avaliação através de modelos, expressões ou fórmulas fechadas focadas em métricas de latência e vazão do sistema. Alguns trabalhos também utilizaram de modelos de simulação para complementar as fórmulas definidas pelos autores.

3.4 CONSIDERAÇÕES FINAIS

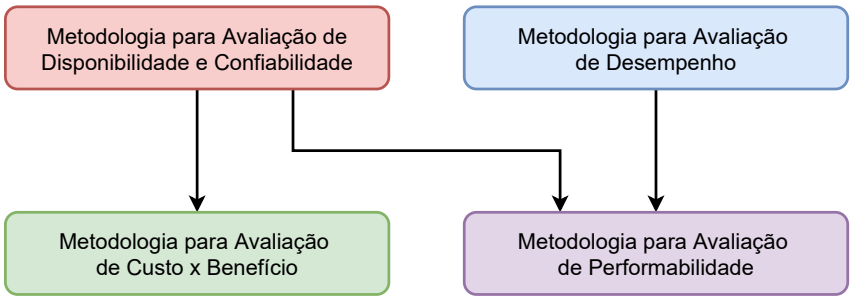
Esta seção apresentou os principais trabalhos relacionados à esta tese. É importante salientar, também, que a presente lista não é exaustiva e que muitos outros trabalhos relacionados podem ser encontrados em mecanismos de busca. Porém, o que queremos demonstrar é que o foco da grande maioria dos estudos é no desempenho dessas plataformas e métricas importantes como os atributos de dependabilidade muitas vezes acabam sendo deixados de lado, e é onde entram as principais contribuições da presente tese.

4 METODOLOGIAS

> *Perhaps there is a universal, absolute truth. Perhaps it clarifies every question. But that's beyond the reach of these small hands.*
>
> — Yang Wen-li, *LOGH*, 1982

Esta seção apresenta as metodologias de apoio utilizadas no processo de desenvolvimento desta tese. A Figura 11 apresenta uma relação de dependência entre as metodologias que serão descritas no decorrer desta seção. As metodologias para avaliação de dependabilidade e de avaliação de desempenho podem ocorrer paralelamente e de forma independente já. Todavia, a metodologia para avaliação da relação entre custo x benefício requer que a metodologia para avaliação de dependabilidade tenha sido finalizada, enquanto que a metodologia para avaliação de performabilidade depende que ambas as metodologias de primeiro nível (dependabilidade e desempenho) tenham sido finalizadas.

Figura 11 – Organização Hierárquica das Metodologias



Fonte: Autoria Própria

4.1 ELEMENTOS COMUNS A TODAS AS METODOLOGIAS

As metodologias apresentadas são dispostas em fluxogramas. Em cada uma delas os **retângulos** representam as macro-atividades. Sua respectiva ordem de execução se dá como a de um grafo direcionado, a partir do topo. Somente após a finalização de uma macro-atividade, o avaliador avança para a seguinte.

Os **losangos** representam escolhas, estas que poderão seguir por caminhos distintos a depender de sua saída que pode apresentar dois possíveis valores: Sim ou Não. As escolhas em nossas metodologias são utilizadas no processo de determinação da satisfatoriedade dos resultados analisados, ou seja, se os resultados obtidos forem considerados satisfatórios avançamos, caso contrário, retrocederemos a uma macro-atividade anterior. O conector lógico **OR** determina que apenas uma saída é factível para cada escolha e, mesmo na ocorrência de uma saída negativa (Não), retornaremos à uma única macro-atividade.

Por fim, as **elipses** tracejadas discriminam importantes micro-atividades para o contexto da macro-atividade em questão. A seguir, apresentamos detalhes sobre macro-atividades comuns à todas as metodologias e descrevemos suas pré-condições, entradas, ações, produtos e pós condições.

4.1.1 Análise de resultados

Essa análise, contempla o conjunto de resultados obtidos pela avaliação de modelos de disponibilidade, confiabilidade ou performabilidade, ou ainda pelo processo de avaliação de desempenho mensurado. Apontamos a viabilidade desses resultados de acordo com a respectiva métrica avaliada.

- Pré-requisitos: Avaliação das métricas de interesse;
- Entradas: Arquivos com dados das medições e resultados das avaliações dos modelos propostos;
- Ações: Análise de dados através de sumário estatístico;
- Produtos: Inferência estatística sobre os dados analisados e seu comportamento;
- Pós-requisitos: Conclusões sobre a viabilidade dos dados.

Caso os resultados obtidos sejam considerados satisfatórios, avançamos a macro-atividade seguinte, caso contrário, retornaremos à alguma das macro-atividades anteriores como aponta a nossa escolha de nome **Satisfatório?** presente em todas as metodologias.

4.1.2 Apresentação de Resultados e Recomendações

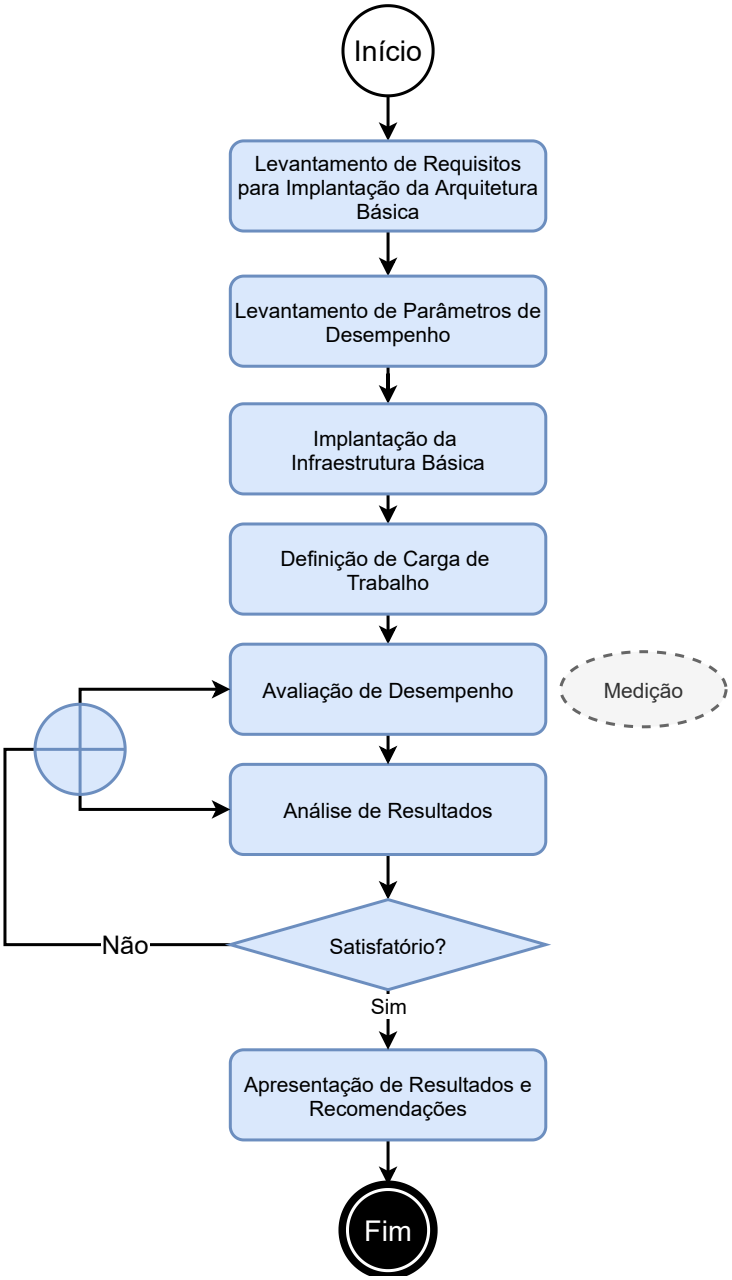
Os dados brutos analisados e considerados satisfatórios são então direcionados ao processo de apresentação de resultados e suas respectivas recomendações. Esta macro-atividade é caracterizada pela representação dos dados em formatos ditos de alto nível (gráficos e tabelas), compreensíveis por pessoas responsáveis pela tomada de decisão que não tenham conhecimento prévio em processos de avaliação de desempenho ou atributos de dependabilidade.

- Pré-requisitos: Dados analisados e considerados viáveis;
- Entradas: Arquivos com dados das medições e resultados das avaliações dos modelos propostos;
- Ações: Criação de elementos visuais que auxiliem o processo de tomada de decisão;
- Produtos: Gráficos e Tabelas;
- Pós-requisitos: Conclusões e recomendações com base nos resultados obtidos.

4.2 METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO

A Figura 12 apresenta uma visão de alto nível da metodologia de avaliação de desempenho e enumera os passos que foram seguidos para realização dessa avaliação em uma infraestrutura computacional com os requisitos mínimos necessários à operação de um serviço ou aplicação baseado no Hyperledger Fabric.

Figura 12 – Metodologia de Avaliação de Desempenho



Fonte: Autoria Própria

Levantamento de requisitos de implantação para a arquitetura básica

Esta macro-atividade consiste na identificação dos principais componentes do Hyperledger Fabric e como estes se relacionam entre si no processo de provimento do serviço. Tal conhecimento nos fornecerá os conhecimentos técnicos e teóricos necessários à implantação de uma infraestrutura privada capaz de hospedar um serviço.

- Pré-requisitos: Conhecimento teórico sobre o funcionamento de blockchain e suas tecnologias;
- Entradas: Vídeos, Tutoriais, Man-Pages;
- Ações: Identificação dos principais componentes do sistema;
- Produtos: Lista de requisitos mínimos necessários a implantação de serviço ou aplicação baseado no Hyperledger Fabric;
- Pós-requisitos: Compreensão do sistema a ser implantado e do seu funcionamento.

Levantamento de Parâmetros de Desempenho

A segunda macro-atividade consiste no levantamento de parâmetros e métricas de interesse.

- Pré-requisitos: Compreensão do sistema a ser implantado e do seu funcionamento;
- Entradas: Livros, White-Papers, Manuais e Artigos Científicos;
- Ações: Identificação das principais métricas e parâmetros de desempenho;
- Saídas: Lista de métricas e parâmetros de desempenho relacionados a plataformas e serviços baseados em blockchain;
- Pós-requisitos: Compreensão das métricas a serem avaliadas.

Implantação da Arquitetura Básica

Esta macro-atividade consiste na implantação de uma arquitetura com os componentes mínimos necessários à operação do sistema, fazendo uso de uma aplicação básica provida pela plataforma Hyperledger Fabric.

- Pré-requisitos: Compreensão do sistema;
- Entradas: Conjunto de hardware e software necessário a operação de um serviço baseado em blockchain;
- Ações: Implantação de serviço em uma infraestrutura privada;

- Produtos: Arquitetura básica;
- Pós-requisitos: Infraestrutura privada com um serviço em execução.

Definição de carga de trabalho

Implantado o sistema base, fomos capazes de definir a carga de trabalho a ser aplicada sobre ele. Esta macro-atividade toma como entrada o Hyperledger Caliper para realização do benchmark através de seus scripts, onde definimos tanto um conjunto quanto uma frequência de transações a ser submetida ao sistema.

- Pré-requisitos: Infraestrutura privada com um serviço em execução;
- Entradas: Hyperledger Caliper;
- Ações: Definição da carga de trabalho que será aplicada sobre o serviço em execução;
- Produtos: Scripts modificados para injeção de carga de trabalho e monitoramento do ambiente;
- Pós-requisitos: Ambiente de testes integrado.

Avaliação de Desempenho

Esta macro-atividade se dá pela aplicação da carga de trabalho previamente estabelecida através das modificações dos scripts presentes no âmbito do Hyperledger Caliper e do desenvolvimento de outros scripts voltados ao monitoramento do sistema avaliado. Vale salientar que os scripts de monitoramento tratam de recursos de hardware do servidor, como RAM, Disco e CPU.

- Pré-requisitos: Implantação da arquitetura básica;
- Entradas: Scripts de teste e monitoramento;
- Ações: Aplicação da carga de trabalho e monitoramento do serviço e do servidor;
- Produtos: Arquivos com a saída do experimento e do monitoramento realizados;
- Pós-requisitos: Dados sobre vazão e latência de transações no sistema e sobre o consumo de recursos como RAM, Disco e CPU do servidor.

Análise de resultados

A análise de resultados obtidos contempla a nossa avaliação de desempenho. Tal avaliação se deu através de medição, onde uma carga de trabalho foi aplicada sob um ambiente de testes. Métricas como vazão e latência, além do consumo de recursos estiveram dentre os resultados de nosso interesse. Para detalhes gerais sobre esse tipo de macro-atividade de análise de resultados consulte a subsubseção 4.1.1.

Apresentação de Resultados e Recomendações

Os dados brutos analisados e considerados satisfatórios, ou seja, factíveis para cenários do mundo real, foram então direcionados ao processo de apresentação de resultados e suas respectivas recomendações. Gráficos e tabelas correspondentes foram gerados e correspondem a uma visão de alto nível capaz de auxiliar o processo de tomada de decisão.

4.3 METODOLOGIA PARA AVALIAÇÃO DE DISPONIBILIDADE E CONFIABILIDADE

A segunda metodologia é apresentada na Figura 13, onde listamos os passos que foram seguidos para realização do processo de modelagem e avaliação de disponibilidade e confiabilidade de infraestruturas privadas baseadas no Hyperledger Fabric. Essa metodologia é composta por sete macro-atividades que serão descritas na sequência.

Identificação dos principais componentes do sistema

A primeira macro-atividade da metodologia para avaliação de dependabilidade diz respeito à identificação dos principais componentes da plataforma Hyperledger Fabric e como os mesmos se relacionam entre si no processo de provimento do serviço. O foco desta macro-atividade é a compreensão do sistema sob a perspectiva do provedor do serviço.

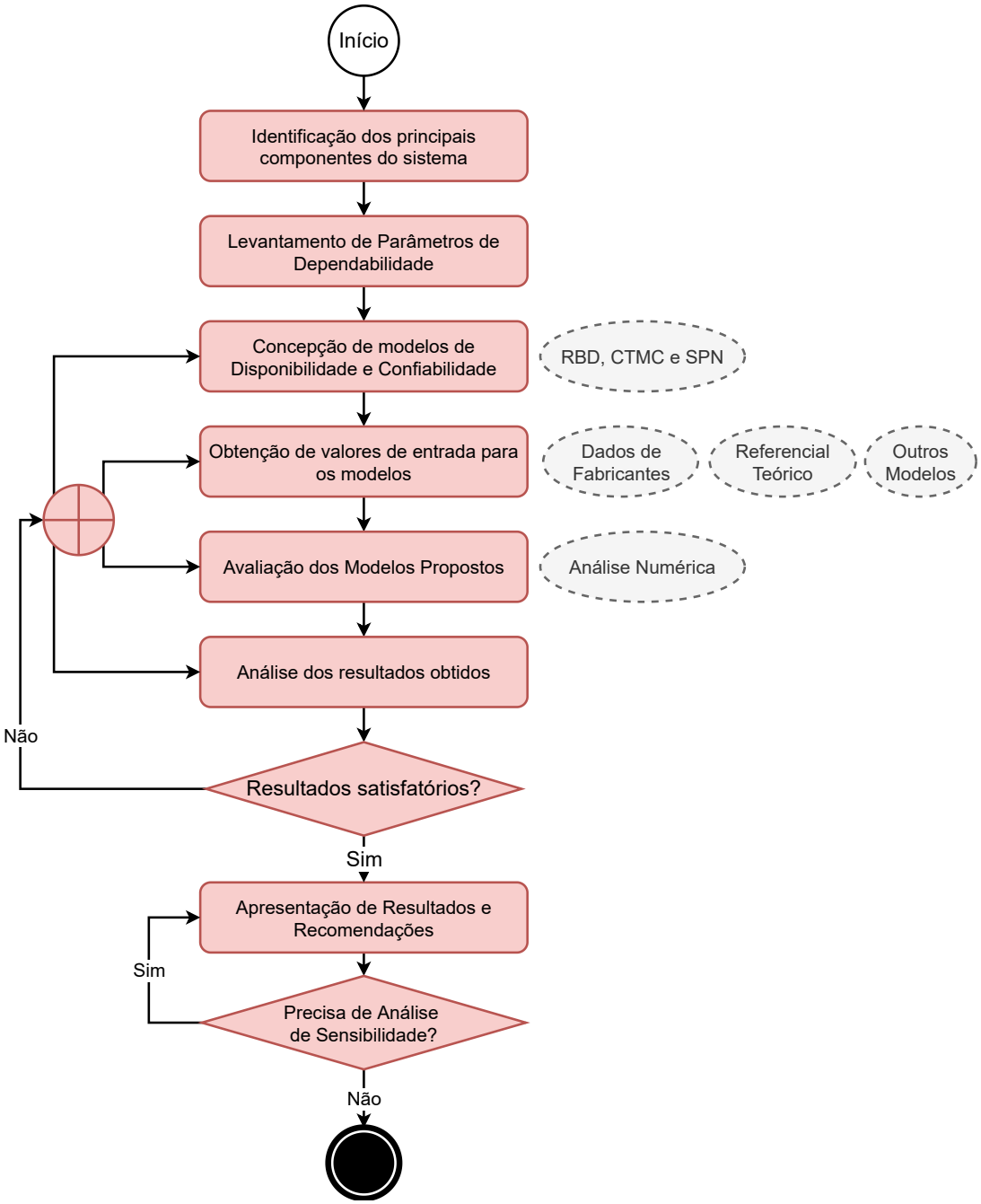
- Pré-requisitos: Conhecimento teórico e técnico sobre o funcionamento da plataforma Hyperledger Fabric;
- Entradas: Vídeos, Tutoriais, Man-Pages e White-Papers;
- Ações: Compreensão do sistema;
- Produtos: Lista com os principais componentes do Hyperledger Fabric e como estes se relacionam entre si;
- Pós-requisitos: Compreensão do sistema a ser avaliado e do seu funcionamento.

Levantamento de Parâmetros de Dependabilidade

Sabendo como o ambiente funciona poderemos agora delimitar o que queremos avaliar sob a perspectiva da dependabilidade de sistemas, estamos interessados nos atributos disponibilidade e confiabilidade, ou seja, a probabilidade do sistema estar funcional agora ou no futuro com ou sem interrupção em sua operação. Uma melhor compreensão desses atributos faz-se necessária através do referencial teórico.

- Pré-requisitos: Compreensão do sistema a ser avaliado;
- Entradas: Livros, Artigos Científicos;

Figura 13 – Metodologia para Avaliação de Disponibilidade e Confiabilidade



Fonte: Autoria Própria

– Ações: Identificação das principais métricas e parâmetros de dependabilidade;

– Produtos: Lista de métricas e parâmetros de dependabilidade que serão avaliados;

– Pós-requisitos: Compreensão das métricas a serem avaliadas.

Concepção de modelos de Disponibilidade e Confiabilidade

Esta macro-atividade consiste no desenvolvimento de um conjunto de modelos para avaliação dos atributos de dependabilidade de nosso interesse. Para tanto, faremos uso dos componentes previamente identificados. Além disso, é necessário que tenhamos plena compreensão do funcionamento do sistema a ser avaliado, para melhor estabelecer a relação de operação dos componentes no processo de modelagem.

- Pré-requisitos: Compreensão das métricas a serem avaliadas;
- Entradas: Lista de principais componentes do sistema e como se relacionam;
- Ações: Concepção de modelos de disponibilidade e confiabilidade através da ferramenta Mercury;
- Produtos: Modelos (RBDs, CTMCs e SPNs) dispostos de forma hierárquica voltados a avaliação de disponibilidade e confiabilidade de infraestruturas capazes de fornecer blockchain como serviço;
- Pós-requisitos: Modelos de alto nível parametrizáveis e generalizáveis.

Obtenção de valores de entrada para os modelos

Para que os modelos representem adequadamente o comportamento do sistema necessitamos de um conjunto de valores de entrada. Para esta macro-atividade realizamos a obtenção dos valores de entrada de duas diferentes maneiras. A primeira delas foi através de levantamento bibliográfico já a segunda decorreu da avaliação de outros modelos constituindo assim uma abordagem de avaliação hierárquica, onde um conjunto de modelos de alto nível foi a base para outros modelos e submodelos.

- Pré-requisitos: Modelos de alto nível parametrizáveis e generalizáveis;
- Entradas: Livros, artigos científicos, manuais, white-papers, dados de fabricantes;
- Ações: Levantamento de valores de entrada para avaliação dos modelos propostos;
- Produtos: Lista com valores de MTTF e MTTR para cada componente da infraestrutura;
- Pós-requisitos: Modelos prontos para avaliação.

Avaliação dos Modelos Propostos

Esta macro-atividade trata dos processos relacionados à avaliação dos modelos de disponibilidade e confiabilidade propostos. Ela é resultante do processo de modelagem e, também, da obtenção de valores de entrada para esses modelos. A avaliação é através da ferramenta Mercury e a respectiva saída é armazenada para análise posterior.

- Pré-requisitos: Ferramenta Mercury, modelos e seus valores de entrada;
- Entradas: Modelos prontos para avaliação;
- Ações: Avaliação dos modelos de disponibilidade e confiabilidade através da ferramenta Mercury;
- Produtos: Resultados numéricos para as métricas de interesse;
- Pós-requisitos: Valores de disponibilidade e confiabilidade das infraestruturas avaliadas.

Análise dos resultados obtidos

A análise dos resultados obtidos no âmbito da nossa avaliação de disponibilidade e confiabilidade, aponta a viabilidade desses resultados de acordo com cada métrica de interesse. Caso os resultados obtidos sejam considerados satisfatórios avançamos a macro-atividade seguinte, caso contrário, poderemos ter de refinar os modelos propostos, obter novos valores de entrada ou reavaliar as métricas de interesse.

Apresentação de Resultados e Recomendações

Os dados brutos analisados e considerados satisfatórios para os modelos avaliados são então direcionados ao processo de apresentação de resultados e suas respectivas recomendações. Mais uma vez, esta macro-atividade é caracterizada pela representação dos dados em gráficos e tabelas.

Precisa de Análise de Sensibilidade?

A opção por realizar a análise de sensibilidade consiste na necessidade de se identificar os componentes que possuem maior impacto sobre a disponibilidade do sistema. No âmbito desta tese, ela foi aplicada aos modelos do tipo SPN que englobassem cenários de manutenção, optamos pela diferença percentual já presente na ferramenta Mercury. A análise realizada nos permitiu estabelecer um conjunto de índices de sensibilidade para cada componente dos modelos avaliados.

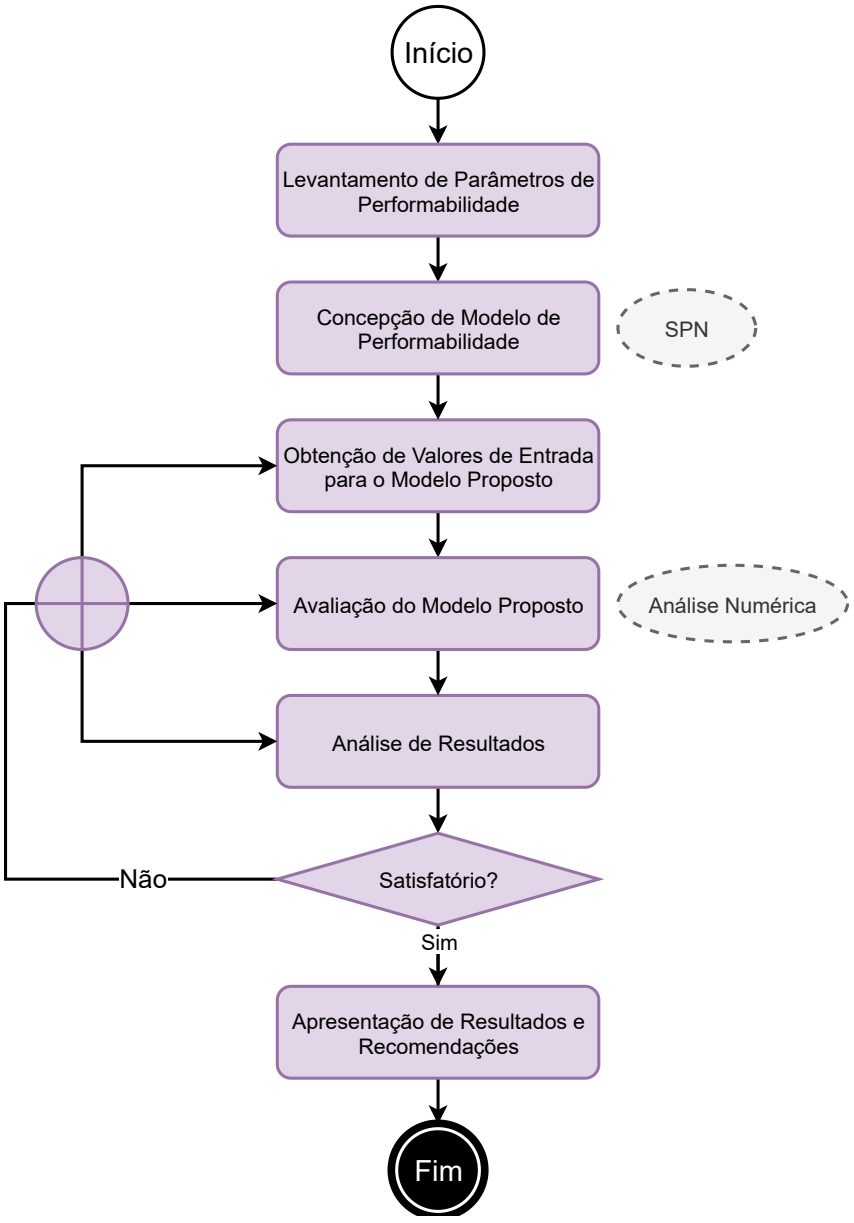
- Pré-requisitos: Ferramenta Mercury;
- Entradas: Modelos SPNs;
- Ações: Aplicação da técnica de análise de sensibilidade aos modelos;
- Produtos: Conjunto de índices de sensibilidade para cada componente;
- Pós-requisitos: Apontar componentes mais importantes do sistema para o provimento do serviço.

Feita a análise de sensibilidade poderemos retornar a macro-atividade anterior e apresentar um novo conjunto de dados, bem como uma série de novas recomendações baseadas no diagnóstico de gargalos feito através da técnica de diferença percentual.

4.4 METODOLOGIA PARA AVALIAÇÃO DE PERFORMABILIDADE

A terceira metodologia é apresentada na Figura 14, o conjunto de macro-atividades a ser seguido para que possamos avaliar a performabilidade de infraestruturas privadas baseadas no Hyperledger Fabric.

Figura 14 – Metodologia de Avaliação de Performabilidade



Fonte: Autoria Própria

Levantamento de Parâmetros de Performabilidade

Esta macro-atividade consiste no estabelecimento da relação entre desempenho e disponibilidade da plataforma na tentativa de obter a performabilidade do sistema. O objetivo primário é determinar o quanto uma carga de trabalho excepcionalmente elevada aplicada à nossa infraestrutura poderá afetar a sua operação, ou seja, a sua disponibilidade.

- Pré-requisitos: Avaliação de Desempenho e de Disponibilidade;
- Entradas: Livros e Artigos Científicos;
- Ações: Estabelecer uma relação entre desempenho e disponibilidade do sistema;
- Produtos: Relação entre o Consumo de Recursos e a Disponibilidade de infraestruturas com base na carga de trabalho;
- Pós-requisitos: Relação entre o desempenho e a disponibilidade.

Concepção de modelo de Performabilidade

O desenvolvimento de um modelo de performabilidade foi realizado no âmbito desta macro-atividade. Para tanto, combinamos as informações sobre o consumo de recursos durante a realização da avaliação de desempenho e a disponibilidade do ambiente de modo a determinar a falha do sistema com base na exaustão de recursos de disco ou memória.

- Pré-requisitos: Relacionar desempenho e disponibilidade;
- Entradas: Dados de consumo de recursos e modelo de disponibilidade;
- Ações: Concepção de modelo de performabilidade;
- Produtos: Modelo (SPN) capaz de combinar disponibilidade e consumo de recursos;
- Pós-requisitos: Modelo de alto nível parametrizável e generalizável.

Obtenção de valores de entrada para os modelos

Os valores obtidos no âmbito desta macro-atividade são oriundos da avaliação de desempenho e de disponibilidade realizada ao longo das metodologias anteriores.

- Pré-requisitos: Modelos parametrizáveis e generalizáveis;
- Entradas: Resultados da avaliação de disponibilidade e da avaliação de desempenho;

- Ações: Levantamento de valores de entrada para avaliação do modelo de performabilidade proposto;
- Produtos: Estabelecimento da métrica de recursos consumidos por unidade de tempo;
- Pós-requisitos: Modelos alimentados com parâmetros para avaliação.

Avaliação do Modelo Proposto

A avaliação do modelo proposto é realizada através da ferramenta Mercury. Os resultados obtidos convergem à disponibilidade geral do sistema sob um conjunto de cenários distintos que determinam o impacto do tipo de transação realizada e do consumo de recursos sobre a métrica de interesse.

- Pré-requisitos: Ferramenta Mercury, modelo e valores de entrada;
- Entradas: Modelos para avaliação de performabilidade;
- Ações: Avaliação do modelo de performabilidade através de análise numérica no âmbito da ferramenta Mercury;
- Produtos: Resultados numéricos para a métrica de interesse;
- Pós-requisitos: Impacto do desempenho sobre a disponibilidade do sistema.

Análise dos resultados obtidos

A análise dos resultados obtidos no âmbito da performabilidade, inclui conclusões acerca do impacto do consumo de recursos sobre a disponibilidade de um serviço que está sendo prestado. A avaliação de desempenho é um pré-requisito para esta macro-atividade e os resultados por ela fornecidos sejam parte dos parâmetros de entrada para o modelo de performabilidade proposto.

É importante salientar que caso os resultados obtidos sejam considerados satisfatórios avançamos a macro-atividade seguinte, caso contrário, poderemos ter de refinar o modelo proposto, obter novos valores de entrada ou reavaliar as métricas de interesse.

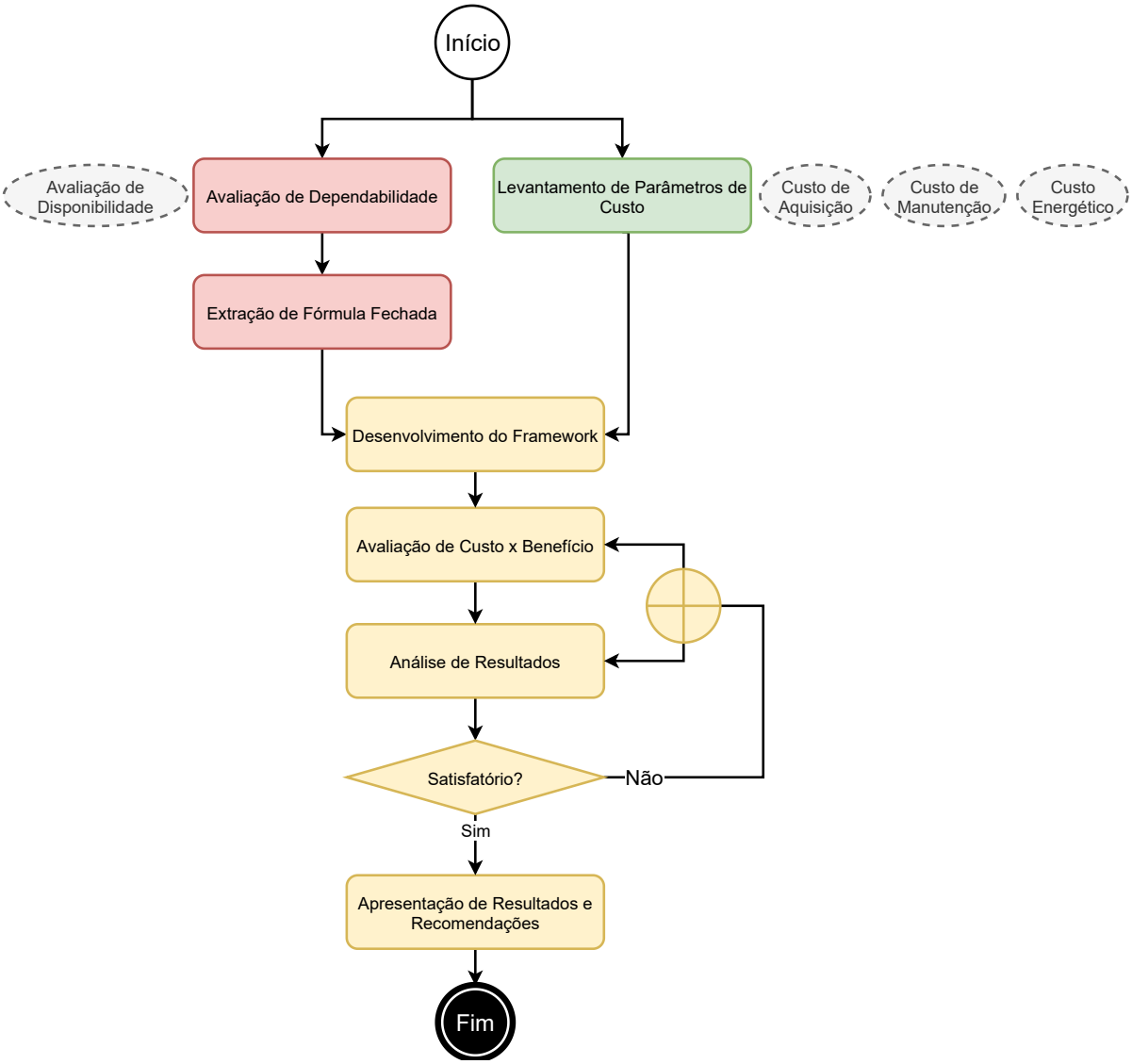
Apresentação de Resultados e Recomendações

Os dados da avaliação de nosso modelo de performabilidade são apresentados em gráficos e tabelas. Mais uma vez, esta macro-atividade é caracterizada pela representação dos dados em formatos compreensíveis pelos tomadores de decisão.

4.5 METODOLOGIA PARA AVALIAÇÃO DE CUSTO X BENEFÍCIO

A quarta e última metodologia é apresentada na Figura 15, e consiste de um conjunto de passos para a avaliação da relação Custo × Benefício. A presente metodologia é dependente da realização prévia das macro-atividades presentes na metodologia de avaliação de dependabilidade. Nosso interesse aqui está diretamente relacionado ao processo de avaliação de disponibilidade. O custo de interesse é o financeiro, já o benefício ao qual iremos relacioná-lo é o downtime anual do sistema, este que quanto menor, melhor é para o provedor do serviço e seus clientes.

Figura 15 – Metodologia de Avaliação da relação Custo × Benefício



Fonte: Autoria Própria

Avaliação de Dependabilidade

A avaliação de dependabilidade diz respeito ao benefício de interesse, àquilo que queremos melhorar através de algum investimento financeiro sobre o serviço que está sendo prestado.

Levantamento de Parâmetros de Custo

A avaliação do modelo proposto é realizada com a ferramenta Mercury. Os resultados obtidos convergem a disponibilidade geral do sistema sob um conjunto de cenários distintos que determinam o impacto das transações que serão realizadas no ambiente do Hyperledger Fabric e do seu respectivo consumo de recursos.

- Pré-requisitos: Conhecimento sobre operações de manutenção e cálculo do custo energético de um equipamento;
- Entradas: Dados da conta de luz, preço de servidores em lojas do Brasil e do exterior, estimativas de preço para diversos tipos de manutenção;
- Ações: Avaliação de um conjunto de custos em torno de um único servidor;
- Produtos: Lista com os custos de aquisição, manutenção e energético de um servidor;
- Pós-requisitos: Resposta para o seguinte questionamento: *Quanto custa hospedar um serviço em um servidor?*

Extração de fórmula fechada

A extração de fórmula fechada corresponde à obtenção de uma expressão matemática capaz de representar de modo equivalente a um dos modelos para avaliação de disponibilidade proposto. A fórmula fechada é a principal saída desta atividade e é a base para as atividades posteriores.

- Pré-requisitos: Ferramenta Mercury, Pacote State Diagrams, Wolfram Mathematica;
- Entradas: Modelo CTMC;
- Ações: Extração de fórmula fechada através do Wolfram Mathematica;
- Produtos: Fórmula fechada;
- Pós-requisitos: Fórmula fechada para avaliação de disponibilidade de infraestruturas genérica.

Desenvolvimento da Ferramenta

A Ferramenta proposta foi intitulada de Blockchain Provisioning Planning Tool (BPPT).

- Pré-requisitos: Extração de fórmula fechada através do Wolfram Mathematica;
- Entradas: Fórmula fechada para avaliação de disponibilidade e equações para o cálculo de custo energético e de operação;
- Ações: Implementação de uma ferramenta que englobe a avaliação de disponibilidade e custo;
- Produtos: Ferramenta Web;
- Pós-requisitos: Implantação da ferramenta.

Avaliação de Custo x Benefício

Esta macro-atividade trata da avaliação de custo × benefício ao relacionar o downtime anual do sistema com o custo total necessário à sua operação. Esta avaliação é feita através da ferramenta proposta, e requer um conjunto de valores de entrada para as métricas que serão avaliadas. É importante salientar que a avaliação de custo × benefício é apenas uma dentre as muitas que podem ser feitas através da ferramenta proposta.

- Pré-requisitos: Desenvolvimento de Ferramenta Web;
- Entradas: MTTFs e MTTRs dos componentes e o seu respectivo custo;
- Ações: Avaliação de custo × benefício;
- Produtos: Resultados numéricos para métricas de interesse;
- Pós-requisitos: Estabelecimento da melhor relação custo × benefício.

Análise dos resultados obtidos

Em termos de custo × benefício, a análise dos resultados obtidos compreende a relação entre downtime anual e os custos necessários a implantação e operação de uma infraestrutura computacional capaz de hospedar aplicações baseadas no Hyperledger Fabric. Os resultados obtidos precisaram ser normalizados para que estejam na mesma grandeza, um valor entre zero e um.

Apresentação de Resultados e Recomendações

Os dados da relação custo × benefício são apresentados em um formato capaz de indicar aos tomadores de decisão um conjunto de opções para auxiliá-la no processo de tomada de decisão. O que inclui, mas não limita-se, a possibilidade de hospedar o serviço em uma infraestrutura de nuvem pública ou privada, ou ainda a escolha de uma política de endossamento que detenha menor impacto financeiro para o provimento do serviço.

4.6 CONSIDERAÇÕES FINAIS

Esta seção apresentou as metodologias que seguimos para o desenvolvimento desta tese. Através delas é possível obter auxílio no processo de compreensão de cada contribuição apresentada no âmbito deste documento, bem como, na replicação do que foi proposto no presente trabalho.

5 ARQUITETURA E MODELOS DE DISPONIBILIDADE

> *My goal is so far away that I'm not even on the starting line yet.*
>
> — Emiya Shirou, *Fate/Stay Night*, 2004

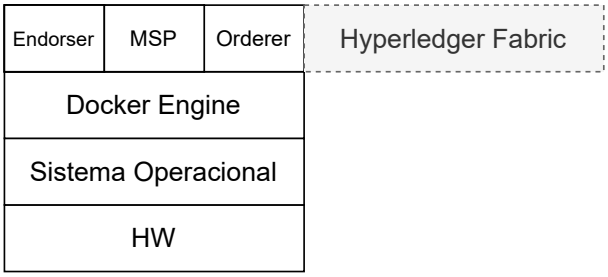
Esta seção apresenta uma visão de alto nível do ambiente avaliado, bem como, os modelos que representam seu funcionamento. Além disso, apresentamos uma série de outros modelos estendidos para avaliação de disponibilidade. Estes que são complementares e tentam sanar algumas das limitações relacionadas ao processo de reparo de componentes em estado de falha.

5.1 ARQUITETURA

No âmbito da presente tese, fizemos uso do Hyperledger Fabric (HLF)[1]. Outras plataformas que fazem uso de smart contracts e Docker para sua implantação teriam pequena ou nenhuma variação quanto à sua representação em modelos de alto nível.

A Figura 16 denota uma visão de alto nível do Hyperledger Fabric, e como a mesma se relaciona com os demais componentes da infraestrutura. Através dessa visão fomos capazes de estabelecer e determinar o comportamento do sistema e traduzi-lo em modelos formais organizados hierarquicamente.

Figura 16 – Layout do Hyperledger Fabric



Fonte: (GROUP et al., 2017)

A partir da camada inferior da pilha temos o Hardware (HW) e o sistema operacional (OS) do servidor. Acima deles encontram-se o Docker Engine, responsável pela criação dos contêineres que irão hospedar os contratos inteligentes e a aplicação baseada em Hyperledger Fabric que será executada. Esses contêineres assumem papéis distintos na plataforma: (i) endossamento de transações; (ii) controle de acesso e; (iii) organização das transações em blocos.

[1] Hyperledger Fabric: https://www.hyperledger.org/projects/fabric

5.2 MODELOS DE DISPONIBILIDADE

Esta subseção apresenta os modos operacionais de arquiteturas especializadas no provimento de aplicações baseadas na plataforma Hyperledger Fabric. Para uma melhor representação optamos pela utilização de modelagem hierárquica, subdividida em dois diferentes níveis de hierarquia. Deste modo, temos um melhor apelo visual e uma leitura mais simplificada do problema que estamos solucionando. O primeiro nível da hierarquia é chamado de Modelo de Infraestrutura, já o segundo nível remete aos Modelos de Provimento de Serviço.

5.2.1 Modelo de Infraestrutura

O primeiro nível hierárquico de nossa modelagem trata da utilização de um modelo RBD para representação dos componentes primários do sistema: Hardware, Sistema Operacional e Docker Engine. É importante destacar que o modo operacional é definido visando o processo de provimento do serviço e não a relação individual entre os componentes. Deste modo, podemos utilizar um RBD em série nos atendo a um nível aceitável de granularidade do sistema, sem representar complexidades com baixo impacto sobre a métrica de interesse.

A Figura 17 é uma representação gráfica deste RBD, enquanto as Equações 5.1 e 5.2 mostram o processo de avaliação de disponibilidade ou da confiabilidade deste modelo em série.

Figura 17 – Modelo de Infraestrutura - RBD

Fonte: Autoria Própria

$$R_{spc}(t) = R_{hw}(t) \times R_{os}(t) \times R_{de}(t), \tag{5.1}$$

Se, e somente se, o tempo até a falha (TTF) do HW, OS e DE forem exponencialmente distribuídos então essa expressão equivale a,

$$R_{spc}(t) = e^{-\lambda_{hw}t} \times e^{-\lambda_{os}t} \times e^{-\lambda_{de}t}, \tag{5.2}$$

As Expressões 5.3 e 5.2 apresentam o cálculo do MTTF para o presente modelo. Esse cálculo permitiu o estabelecimento de um MTTR, visando a obtenção de um valor específico para os índices de disponibilidade.

$$MTTF = \int_0^\infty e^{-(\lambda_{hw}+\lambda_{os}+\lambda de)\,t}\,dt, \tag{5.3}$$

O que nos leva a

$$MTTF = \frac{1}{\lambda_{hw} + \lambda_{os} + \lambda_{de}} \tag{5.4}$$

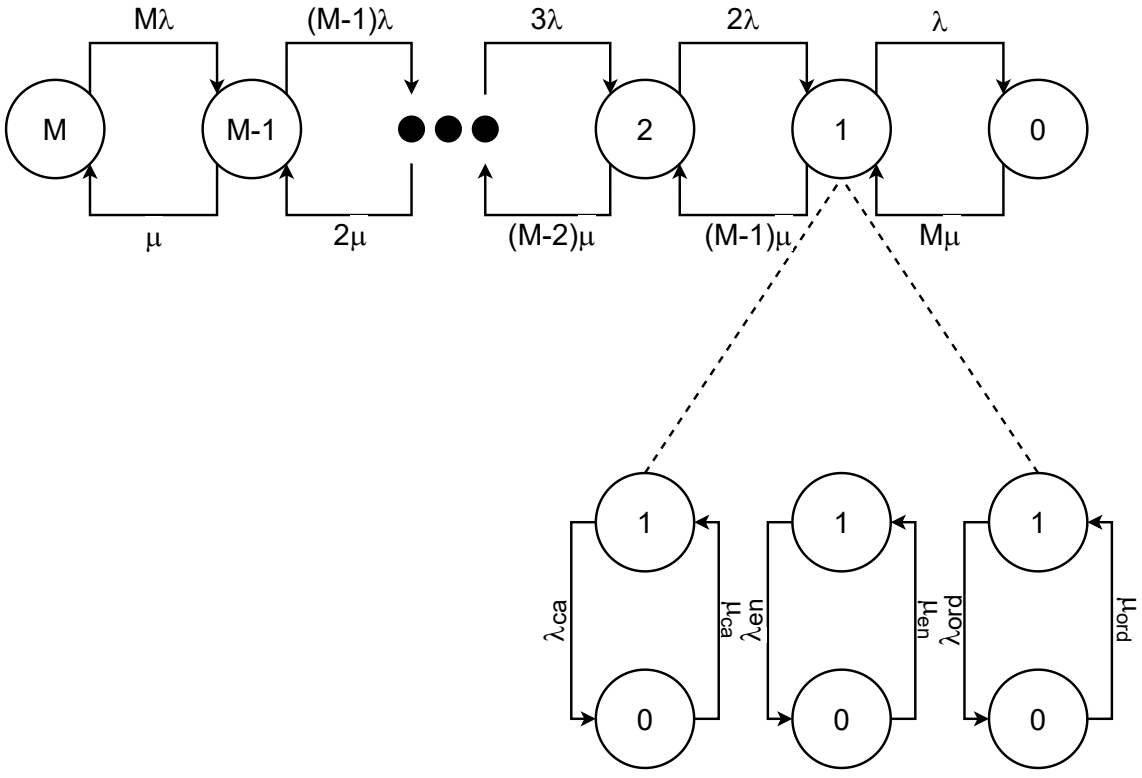
Salientando que os valores para as taxas são inversamente proporcionais ao tempo t, ou seja, 1/t.

Após a avaliação do RBD da Figura 17, obtemos os respectivos MTTFs e MTTRs para o modelo de infraestrutura, ou seja, a base para o modelo de provimento de serviço.

5.2.2 Modelo de Provimento de Serviço

A próxima etapa é caracterizada pela produção de um segundo modelo capaz de conectar o modelo de infraestrutura aos contêineres que proveem o Hyperledger Fabric. Para este novo modelo, fizemos uso de uma CTMC que nos permitiu uma generalização da disponibilidade obtida para ambientes maiores através da extração de uma fórmula fechada. A Figura 18 apresenta uma CTMC multinível e denota o nosso modelo de provimento de serviço.

Figura 18 – Modelo de Provimento de Serviço



Fonte: Autoria Própria

Esta CTMC tem em seu primeiro nível a generalização do modelo de infraestrutura. Deste modo, para uma infraestrutura de tamanho M, ou seja, com um número M de máquinas, cada máquina terá seu respectivo Hardware (HW), Sistema Operacional (OS)

e Docker Engine (DE). Além disso, cada máquina hospeda os três contêineres (Endorser, MSP e Orderer), que são visíveis ao longo do segundo nível de nossa CTMC.

As máquinas podem entrar em estado de falha ou serem reparadas seguindo, respectivamente, taxas λ e μ que correspondem ao inverso dos valores de MTTF e MTTR obtidos através da avaliação do modelo de infraestrutura. Quanto aos contêineres do Hyperledger Fabric, esses têm suas próprias taxas de falha e reparo associadas. Essas que são respectivamente λ_{ca} e μ_{ca} para o contêiner MSP, λ_{en} e μ_{en} para o contêiner responsável pelo endossamento e, λ_{ord} e μ_{ord} para o contêiner responsável pela ordenação das transações em blocos.

Utilizamos o pacote State Diagrams, uma extensão para o Wolfram Mathematica, para extrair um conjunto de expressões a partir do modelo de provimento de serviço. As expressões extraídas são capazes de avaliar a disponibilidade geral do sistema com base nas diversas políticas de endossamento utilizáveis.

A Equação 5.5 representa o provimento de infraestrutura em um único servidor. Este servidor corresponde a uma única máquina física, com seus três respectivos contêineres a ela associados.

$$A_{\text{Server}} = \left(\frac{\mu}{\mu+\lambda}\right) \times \left(\frac{\mu_{ca}}{\mu_{ca}+\lambda_{ca}}\right) \times \left(\frac{\mu_{en}}{\mu_{en}+\lambda_{en}}\right) \times \left(\frac{\mu_{ord}}{\mu_{ord}+\lambda_{ord}}\right) \tag{5.5}$$

Através da equação 5.5 calculamos a disponibilidade (A) de um único servidor. Isto é um fator limitante, já que almejamos uma avaliação genérica e que atenda as necessidades de empresas e infraestruturas dos mais variados tamanhos. Sanamos esta limitação através da utilização de um binomial, similar àquele de blocos KooN em modelos RBDs. Todavia, capaz de relacionar servidores e contêineres, denotando uma relação de dependência entre ambos. Na falha dos componentes primários (HW, OS, DE) haverá falha dos contêineres hospedados naquele servidor.

A Equação 5.6 representa uma política de endossamento do tipo K-out-of-N, onde alguém, administrador ou desenvolvedor, definirá um valor mínimo (K) de um total (M) de componentes que precisam estar em operação.

$$A_{\text{KooN}} = \sum_{i=k}^{M} \binom{M}{k} A_{\text{Server}}^{k} (1 - A_{\text{Server}})^{M-k} \tag{5.6}$$

Alguns cenários específicos podem ser derivados a partir da expressão que representa uma política de endossamento do tipo K-out-of-N. A Equação 5.7, apresenta como é calculada a disponibilidade do serviço sob a perspectiva de uma política de endossamento to tipo AND. Esta política de endossamento requer que todos os componentes da infraestrutura estejam operacionais em ambos os níveis da nossa CTMC. Deste modo, esse cenário

específico poderia ser representado através de uma relação N-out-of-N.

$$A_{\text{NooN}} = \left(\frac{\mu}{\mu+\lambda}\right)^M \times \left(\frac{\mu_{ca}}{\mu_{ca}+\lambda_{ca}}\right)^M \times \left(\frac{\mu_{en}}{\mu_{en}+\lambda_{en}}\right)^M \times \left(\frac{\mu_{ord}}{\mu_{ord}+\lambda_{ord}}\right)^M \tag{5.7}$$

Por fim, a Equação 5.8 avalia a disponibilidade de um ambiente configurado para atender uma política de endossamento do tipo OR, ou que requer que pelo menos um dos servidores e seus respectivos contêineres associados esteja operante. Esse novo cenário pode ser tratado como uma política do tipo 1-out-of-N.

$$A_{\text{1ooN}} = \left(1-\left(\frac{\lambda}{\lambda+\mu}\right)^M\right) \times \left(1-\left(\frac{\lambda_{en}}{\lambda_{en}+\mu_{en}}\right)^M\right) \times \left(1-\left(\frac{\lambda_{ord}}{\lambda_{ord}+\mu_{ord}}\right)^M\right) \times \left(1-\left(\frac{\lambda_{ca}}{\lambda_{ca}+\mu_{ca}}\right)^M\right) \tag{5.8}$$

Para o modelo de provimento de serviço apresentado podemos apontar como uma limitação a existência de N equipes de manutenção dentro do provedor de serviço. Na falha de qualquer componente, de acordo com o presente modelo, sempre teremos uma equipe ou funcionário apto a realização da referida manutenção. Na presença de 100, 1000, 10000 ou mais servidores teríamos 100, 1000, 10000 ou mais equipes de manutenção a disposição, o que foge a realidade de qualquer empreendimento. Tentamos contornar essa limitação ao propor um conjunto de modelos parametrizáveis distintos.

5.2.3 Modelo de Provimento de Serviço com Restrições

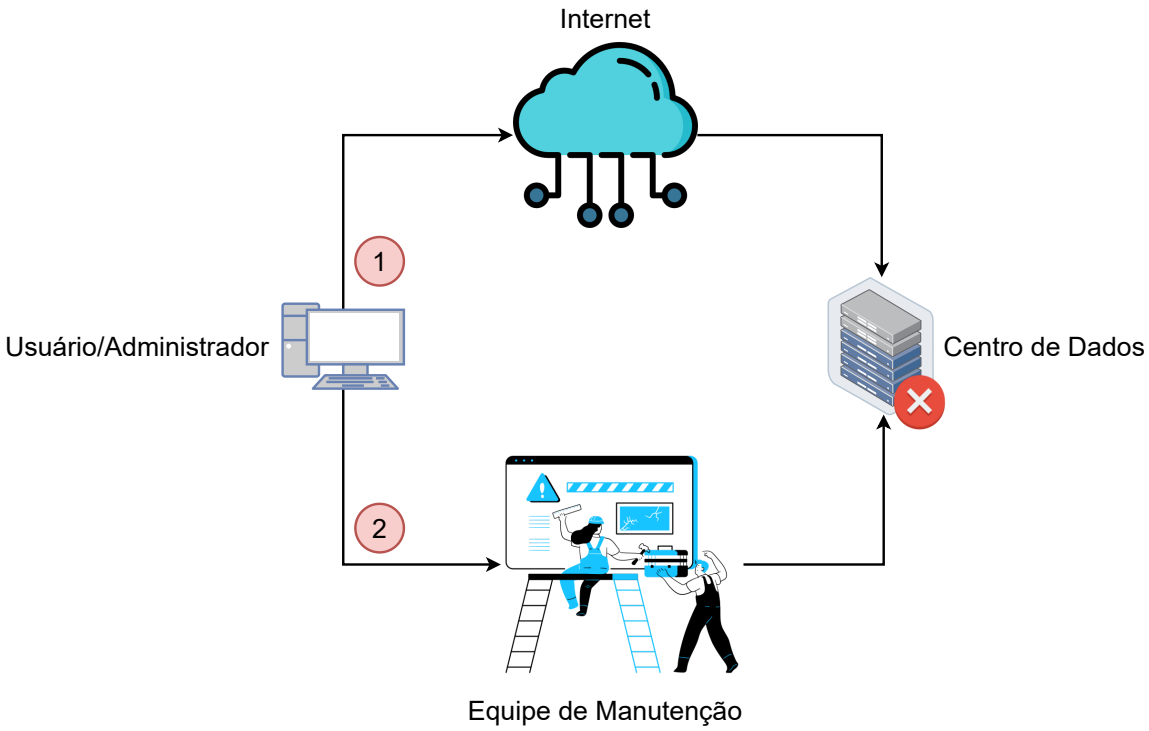
O primeiro modelo de provimento de serviço estendido considera a presença de restrições no número de equipes de manutenção, tempo até a detecção de uma falha e tempo de locomoção da equipe ao referido centro de dados.

As manutenções em nosso modelo seguirão o tipo reativa, esta que é uma das mais comuns e ocorrem se, e somente se, o sistema em questão estiver sob um estado de falha. Deste modo, (1) após a detecção da ocorrência de uma falha no provimento do serviço; (2) uma equipe de manutenção será chamada pela equipe administrativa.

A Figura 20 apresenta o modelo de provimento de serviços com restrições. Esta SPN possui um conjunto de transições (MTTF, DTC, M2S, MTTR_S e MTTR), todas com políticas de disparo do tipo *single server* indicando a impossibilidade de dois eventos do mesmo tipo ocorrerem simultaneamente.

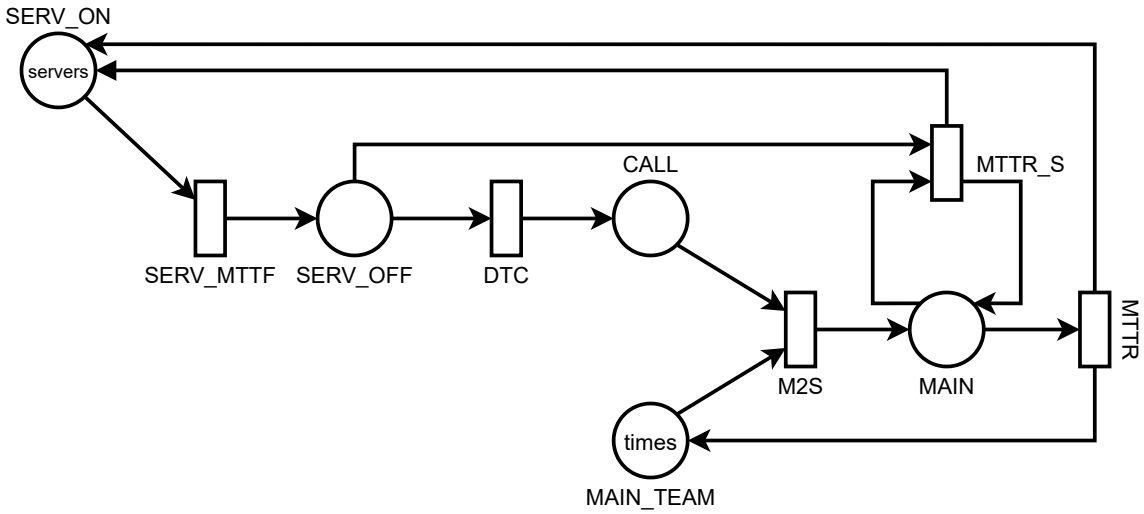
O presente modelo possui um conjunto de marcações (*tokens*) iniciais nos lugares SERV_ON e MAIN_TEAM. A presença de uma marcação no lugar SERV_ON é um indicativo de que pelo menos um servidor está operacional. Deste modo, SERV_ON corresponde ao número de servidores disponíveis e em operação, cada qual com seu respectivo hardware, sistema operacional, docker engine e os respectivos contêineres responsáveis por executar uma aplicação. Após a passagem do tempo médio entre falhas esperado para o

Figura 19 – Visão de alto nível do provimento de serviço com restrições



Fonte: Autoria Própria

Figura 20 – Modelo de Provimento de Serviço Reativo com Restrições



Fonte: Autoria Própria

servidor, representado pela transição SERV_MTTF, uma marcação será movida do lugar SERV_ON para um estado de falha do servidor, este que é representado pelo lugar SERV_OFF.

Como as falhas, tradicionalmente, não são imediatamente diagnosticadas e catalogadas pelo provedor do serviço, há um período de tempo até sua detecção, este representado pela transição DTC (*detection*). O DTC é um tempo administrativo, onde o administrador do sistema, após se deparar com inoperabilidade do serviço, deverá contactar uma equipe de manutenção, o que ocorre através da transição CALL. Se houver uma equipe de manutenção disponível (algo que é indicado pela presença de marcação em MAIN_TEAM), esta deverá mover-se ao local do servidor (M2S ou *move-to-site*) e realizar a manutenção em tempo pré-estabelecido em contrato (MTTR).

Caso a equipe de manutenção já esteja no centro de dados e se depare com a ocorrência de novas falhas de serviço, esta poderá realizar um outro tipo de reparo, representado pela transição MTTR_S. Para o disparo desta transição, é necessário a presença de ao menos uma marcação no lugar SERV_OFF, indicando a inoperabilidade de um ou mais servidores.

Por fim, uma política de prioridades foi estabelecida para duas das transições presentes no modelo, a transição MTTR_S tem prioridade sobre a transição MTTR, visando a garantia de que todos os componentes em falha no centro de dados serão reparados antes da equipe de manutenção deixar o local. A Tabela 5 apresenta detalhes sobre cada uma das transições do modelo de provimento de serviço com restrições.

Tabela 5 – Transições do modelo de provimento de serviço com restrições

Transição	**Política de Disparo**	**Prioridade**
MTTF	Single Server	1
DTC	Single Server	1
M2S	Single Server	1
MTTR	Single Server	1
MTTR_S	Single Server	2

Fonte: Autoria Própria

A Tabela 6 descreve como as marcações se movem de um lugar para o outro dentro do modelo de manutenção do tipo reativo. A movimentação dessas marcações ocorre de acordo com os respectivos tempos necessários para que cada transição seja disparada.

5.2.4 Modelo de Provimento de Serviço *Self-Healing* + Reativo com Restrições

O segundo modelo estendido com restrições considera um mecanismo de *self-healing* além da já tradicional manutenção reativa. A Figura 21 apresenta tal mecanismo, que trata a possibilidade de autocorreção através de um monitor de aplicação que: (1) verifica se os

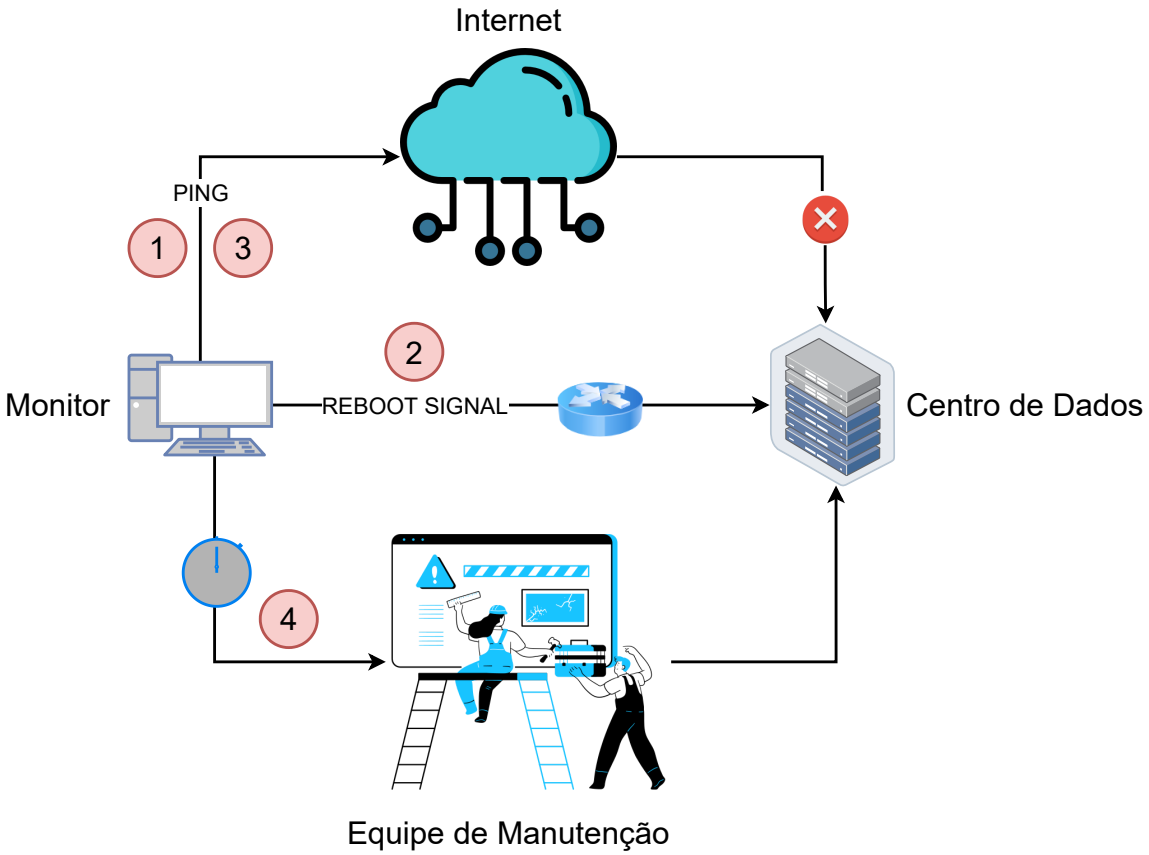
Tabela 6 – Função-δ para cada estado do modelo de provimento de serviço com restrições

Lugar/Trans.	**SERV_MTTF**	**DTC**	**M2S**	**MTTR**	**MTTR_S**
SERV_ON	SERV_OFF	-	-	-	-
SERV_OFF	-	CALL	-	-	SERV_ON
CALL	-	-	MAIN.	-	-
MAIN_TEAM	-	-		-	-
MAIN.	-	-	-	SERV_ON /MAIN_T	SERV_ON /MAIN

Fonte: Autoria Própria

servidores estão operacionais, caso um ou mais servidores não esteja sob essas condições; (2) o mecanismo tentará reiniciá-lo localmente. O *reboot* pode ou não ter sucesso no reparo do sistema, deste modo; (3) uma nova checagem é realizada e, caso o mesmo continue em falha; (4) aguarda-se a passagem do tempo administrativo e realiza-se o chamado da equipe de manutenção.

Figura 21 – Visão de alto nível do provimento de serviço com mecanismo de Self-Healing



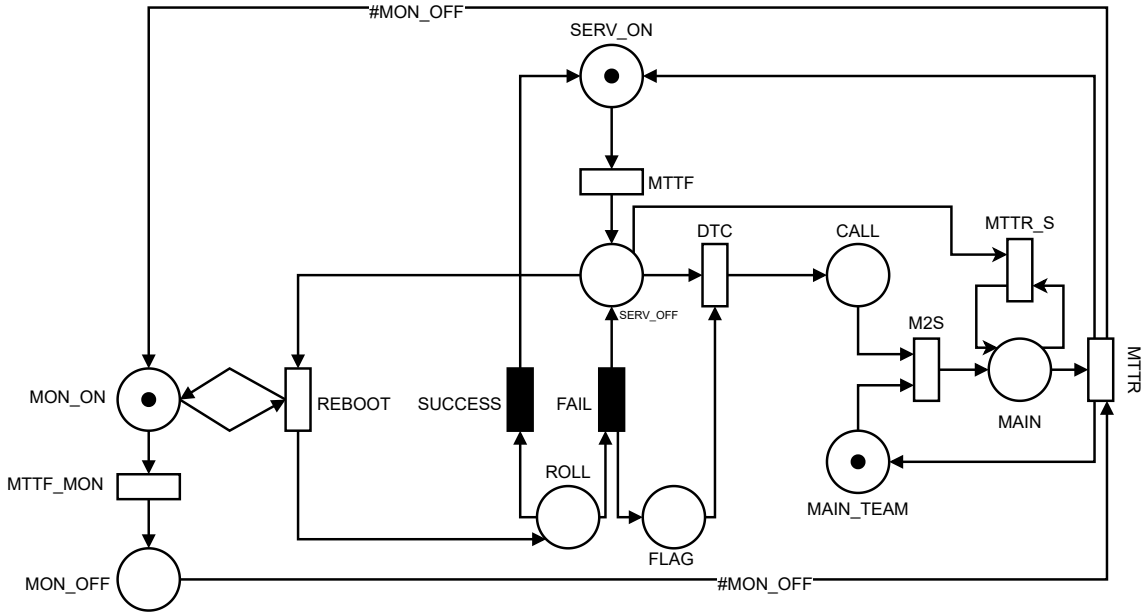
Fonte: Autoria Própria

A Figura 22 apresenta o segundo modelo estendido com restrições, nele a probabilidade do reparo automático está associada ao lugar ROLL e suas duas transições de saída,

SUCCESS e FAIL, que significam respectivamente a probabilidade de sucesso ou de falha no reparo através de *reboot*. Em caso de ocorrência de falha no reparo automático, os servidores permanecerão inoperantes a espera da passagem do tempo administrativo para sua detecção, indicado pela transição DTC. Após a passagem de tempo um administrador do sistema irá efetuar um chamado a equipe de manutenção, esta que deverá se locomover ao local, indicado pela transição M2S, e realizar os devidos reparos através das transições MTTR ou MTTR_S. É importante ressaltar que o monitor também poderá entrar em estado de falha, saindo do estado MON_ON para MON_OFF, e que é um trabalho da equipe mantenedora realizar o seu reparo em conjunto com o dos servidores em falha.

Além disso, a política de prioridades foi novamente estabelecida para duas das transições presentes no modelo, a transição MTTR_S tem prioridade sobre a transição MTTR. Deste modo, MTTR não dispara enquanto MTTR_S puder disparar.

Figura 22 – Modelo de manutenção Self-Healing + Reativo



Fonte: Autoria Própria

Um outro importante lugar recebe o nome de FLAG, e armazena a marcação que alerta ao monitor que o reparo através de *reboot* já tentou ser realizado, mas não obteve sucesso. Deste modo, se houver ao menos uma marcação em FLAG o monitor irá aguardar a passagem do tempo administrativo e não tentará reiniciar o servidor e suas aplicações infinitamente. Uma possível extensão a esta característica é fazer com que o monitor faça o chamado da equipe de manutenção automaticamente. Isso aumentará consideravelmente as chances de detecção de falha por parte da equipe de manutenção. A Tabela 7 apresenta detalhes acerca de cada transição do presente modelo.

A Tabela 8 apresenta a função de próximo estado para este modelo. É importante salientar o dinamismo na escolha dos pesos associados às transições imediatas SUCCESS

Tabela 7 – Transições do modelo de provimento de serviço com restrições e self-healing

Transição	**Política de Disparo**	**Peso**	**Guarda**	**Prioridade**
MTTF	Single Server	-	-	1
DTC	Single Server	-	-	1
M2S	Single Server	-	-	1
REBOOT	Single Server	-	#FLAG = 0	1
SUCCESS	-	WS	-	1
FAIL	-	WF	-	1
MTTR	Single Server	-	-	1
MTTR_S	Single Server	-	-	2

Fonte: Autoria Própria

e FAIL e representados por WS e WF. A soma de ambos (WS+WF) não deve ser superior a 1.

Tabela 8 – Função-δ para cada estado no modelo de provimento de serviço com restrições e self-healing

Place/Trans.	**MTTF**	**DTC**	**M2S**	**REBOOT**	**SUCCESS**	**FAIL**	**MTTR**	**MTTR_S**
SERV_ON	SERV_OFF	-	-	-	-	-	-	-
SERV_OFF	-	CALL	-	ROLL	-	-	- SERV_ON	
CALL	-	-	MAIN.	-	-	-	-	-
MAIN_TEAM	-	-		-	-	-	-	-
MAIN.	-	-	-	-	-	-	SERV_ON /MON_ON	SERV_ON /MAIN_T.
MON_ON	-	-	-	ROLL /MON_ON	-	-	-	-
MON_OFF	-	-	-	-	-	SERV_OFF	-	-
ROLL	-	-	-	-	SERV_ON	SERV_OFF /FLAG	-	-
FLAG	-	-	-	-	-	CALL	-	-

Fonte: Autoria Própria

5.3 CONSIDERAÇÕES FINAIS

Esta seção apresentou a arquitetura do Hyperledger Fabric, bem como, os respectivos componentes necessários ao seu funcionamento e suas políticas de endossamento. Também apresentamos modelos de disponibilidade que serão avaliados por esta tese. É importante salientar que os artefatos aqui mostrados podem ser readaptados a realidade de cada empreendimento mediante parametrização, podendo ser readequados a outras infraestruturas de tamanho menor ou maior do que as aqui apresentadas.

6 BLOCKCHAIN PROVISIONING PLANNING TOOL

> *I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched C-beams glitter in the dark near the Tannhauser gate. All those moments will be lost in time, like tears in rain...*
>
> — Roy Batty, *Blade Runner*, 1982

Blockchain Provisioning Planning Tool (BPPT) é o nome dado a ferramenta proposta por esta tese. O BPPT pode ser utilizado no planejamento de infraestruturas capazes de prover ambientes baseados na tecnologia blockchain e que se utilizam da plataforma Hyperledger Fabric. Para uso e consulta da presente ferramenta siga o presente link <https://www.cmelo.com.br/bppt>.

Através do BPPT é possível realizar o planejamento de um serviço baseado em blockchain orientando-o à métricas de disponibilidade e custos. Os custos por ela considerados são aqueles relacionados à manutenção e implantação de infraestruturas de blockchain privadas. Além disso, o BPPT também possibilita a realização de uma comparação entre os índices obtidos através de infraestruturas privadas com àqueles que podem ser alcançados pela utilização das principais plataformas de nuvem pública.

6.1 REQUISITOS DE PROJETO E IMPLEMENTAÇÃO

O BPPT foi inteiramente desenvolvido através da linguagem de programação Python, em sua versão 3.7, com o apoio do framework Django, um framework que é voltado ao desenvolvimento rápido para web. A decisão pelo desenvolvimento web teve como base a abrangência e o alcance esperado para a ferramenta proposta, além disso, optamos por código aberto e hospedagem em repositório público da mesma no GitHub, garantindo, assim, a extensão de sua vida útil, algo que é alcançado através da sua contínua manutenção por parte da sempre ativa comunidade de software livre.

Para a *dashboard*, uma interface unificada que apresenta e compõe todo o conjunto de gráficos e tabelas auto-geráveis com base nos valores de entrada fornecidos usou-se o Plotly[1] em combinação com o Dash Bootstrap[2]. Já para operação e implantação da ferramenta proposta, fizemos uso da plataforma Heroku[3] e de uma de suas instâncias gratuitas. Todavia, a principal base para a construção da ferramenta é um dos modelos de disponibilidade apresentados na seção anterior.

Como anteriormente mencionado, a partir da CTMC multinível proposta (Figura 18), fomos capazes de extrair um conjunto de expressões matemáticas equivalentes, e que

[1] Plotly: https://plotly.com/
[2] Dash Bootstrap: https://dash-bootstrap-components.opensource.faculty.ai
[3] Heroku: https://www.heroku.com

também representasse o cálculo de disponibilidade em infraestrutura genérica baseada no Hyperledger Fabric. Essas expressões representam ambientes dispostos sob um conjunto de políticas de endossamento distintas (Equações 5.6, 5.8, 5.7). Levamos estas expressões ao ambiente do Python, e a partir delas, o BPPT foi construído.

Todo o processo realizado para implementação do BPPT se resume a construção do modelo base através da ferramenta Mercury, passando por sua respectiva exportação para o pacote State Diagrams do Wolfram Mathematica, a extração de fórmula fecha, e a sua respectiva exportação para o ambiente Python.

A Tabela 9 apresenta os requisitos funcionais considerados no processo de desenvolvimento do BPPT. Ao longo das próximas seções iremos destacar os presentes requisitos funcionais listados e como os mesmos foram implementados.

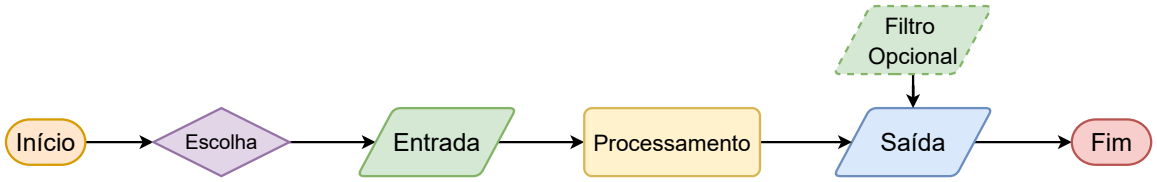
Tabela 9 – Requisitos Funcionais do BPPT

Functional Requirements	
FR01	Availability Evaluation
FR02	Expected Annual Downtime
FR03	Cost Evaluation
FR04	Simple Experiment
FR05	General Experiment

Fonte: Autoria Própria

Através de um conjunto de fluxogramas apontamos como se dá o funcionamento de cada componente presente na ferramenta proposta, desde a respectiva avaliação de disponibilidade à realização de experimentos gerais e específicos. Todos os fluxogramas aqui apresentados seguem o mesmo padrão, este que é mostrado de forma genérica através da Figura 23.

Figura 23 – Legenda dos Fluxogramas



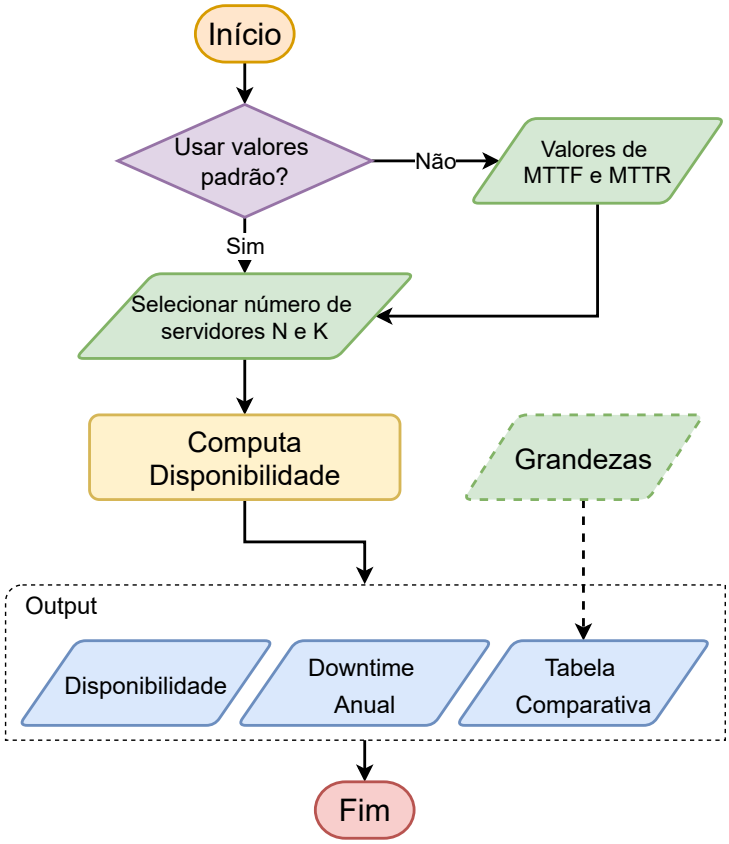
Fonte: Autoria Própria

6.2 PLANEJAMENTO DE DISPONIBILIDADE

A principal funcionalidade promovida pelo BPPT é a do cálculo da disponibilidade de infraestruturas computacionais com base no conjunto de políticas de endossamento

pré-estabelecidas (AND, OR e KooN). O planejador ou avaliador, quando interessado na disponibilidade da infraestrutura, utilizará equações previamente obtidas na avaliação da nossa CTMC multinível para determinar os índices de disponibilidade do sistema e o seu respectivo downtime anual. A Figura 24 apresenta o fluxograma de utilização para a função de planejamento de disponibilidade.

Figura 24 – Fluxograma do Planejamento de Disponibilidade



Fonte: Autoria Própria

O usuário de nossa ferramenta tem como opção inicial utilizar valores pré-definidos para os MTTFs e MTTRs dos componentes pertencentes à infraestrutura. Caso opte por fazer uso de valores personalizados, este deverá fornecê-lo para cada componente ou combiná-los com um conjunto de valores pré-definidos. Após essa etapa inicial, é necessário estabelecer o número de servidores total (N) e o número de servidores que se espera que estejam operacionais, este último quando considerando uma política de endossamento do tipo KooN.

A partir dos valores inseridos como entrada à ferramenta fornecerá uma saída. Esta saída se dá em forma de gráficos de barra e de linha, que representma a respectiva disponibilidade em porcentagem, bem como o respectivo downtime anual em horas, obtidos para cada política de endossamento. Além disso, a BPPT apresenta uma tabela comparativa entre as políticas de endossamento avaliadas. Essa tabela garante ao avaliador a possibilidade de aplicar filtros e determinar a grandeza ou intervalo para os resultados obtidos, ou

seja, para o cálculo da disponibilidade ele/ela poderá optar pelo número de noves ou pela disponibilidade em porcentagem, já para o cálculo do downtime anual, escolherá entre horas, minutos e segundos. Por fim, também é possível definir o número de casas decimais para os mais variados cenários.

6.3 PLANEJAMENTO DE CUSTOS

O planejamento de custos opera de modo similar à funcionalidade de planejamento de disponibilidade, já que este também estende as métricas de disponibilidade na tentativa de estabelecimento de uma relação entre custo e benefício de possíveis infraestruturas. O benefício aqui considerado diz respeito a disponibilidade e o custo aos respectivos valores relacionados a aquisição, manutenção e operação da infraestrutura.

A Figura 25 apresenta o fluxograma para utilização do planejador de custos. Primeiramente, o usuário definirá se utilizará ou não valores padrões de MTTF e MTTR para os componentes da infraestrutura. Depois, irá determinar o número de servidores (N e K), e a partir daí estabelecer o conjunto de valores relacionadas a aquisição, manutenção e operação do ambiente.

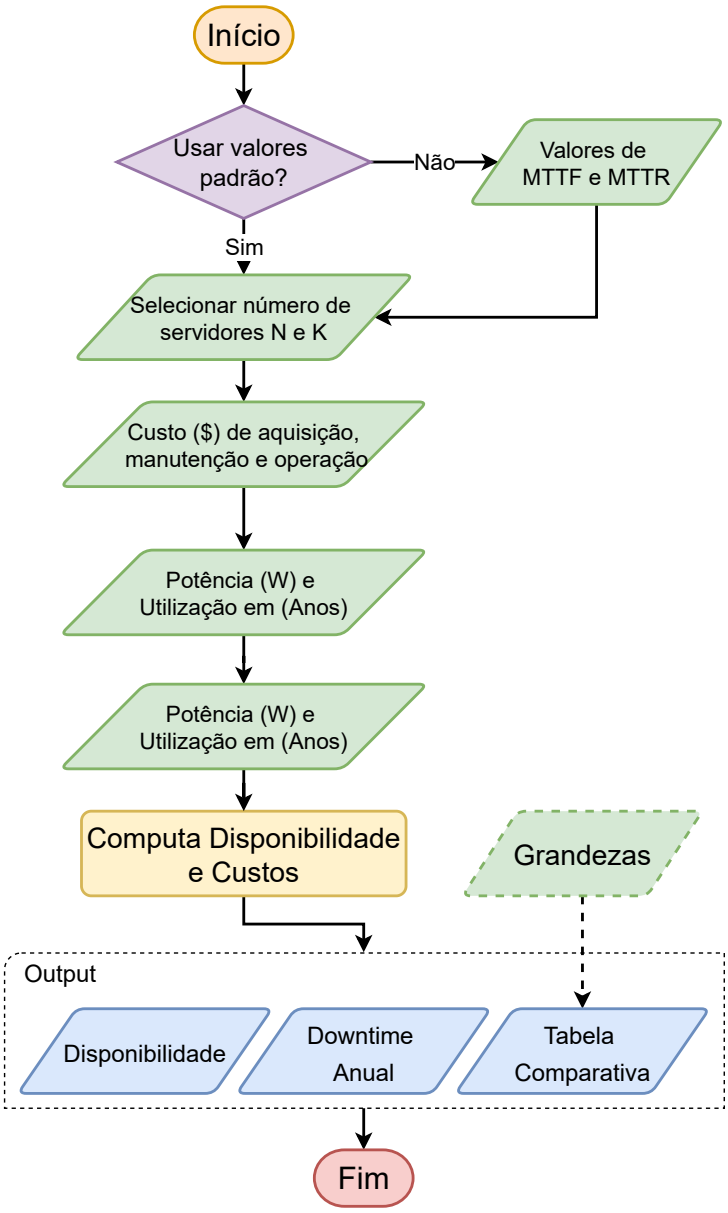
No BPPT, o processo de aquisição diz respeito a compra dos servidores e apenas deles, desconsiderando então outros componentes como racks, switches, e demais equipamentos.

Já a manutenção considera os custos das operações de reparo realizadas quando um servidor entra em estado de falha. Para tanto, o BPPT estima o número total de falhas ocorridas no decorrer de um ano na infraestrutura avaliada. Esta análise considera as várias políticas de endossamento, como por exemplo, se a minha infraestrutura é constituída de duas máquinas e está sob uma política de endorsamento do tipo AND, então a ocorrência de uma falha em qualquer um desses servidores moverá o meu serviço para um estado de falha. Deste modo, uma equipe de manutenção deverá prestar o devido reparo a um certo custo.

Há também os custos operacionais, e que dizem respeito ao custo por kilowatt hora na região de implantação da infraestrutura. Para tanto, a potência em Watts do servidor deverá ser fornecida. Outro importante item a se especificar é o tempo de utilização esperado em anos para aquele dispositivo, assim o BPPT será capaz de calcular a sua depreciação anual.

Assim como o planejador de disponibilidade, o planejador de custos também fornece três diferentes saídas. Uma das saídas fornecidas inclui um gráfico de barras, que representa o custo da adoção de infraestruturas públicas na Amazon, Microsoft e Google e os compara com o custo da aquisição, manutenção e operação de uma infraestrutura privada para a hospedagem de aplicações baseadas no Hyperledger Fabric. Outra importante saída é um gráfico do tipo radar, esta saída está relacionada aos possíveis custos e benefícios providos pelas infraestruturas avaliadas com base em sua política de endossamento.

Figura 25 – Fluxograma do Planejamento de Custos



Fonte: Autoria Própria

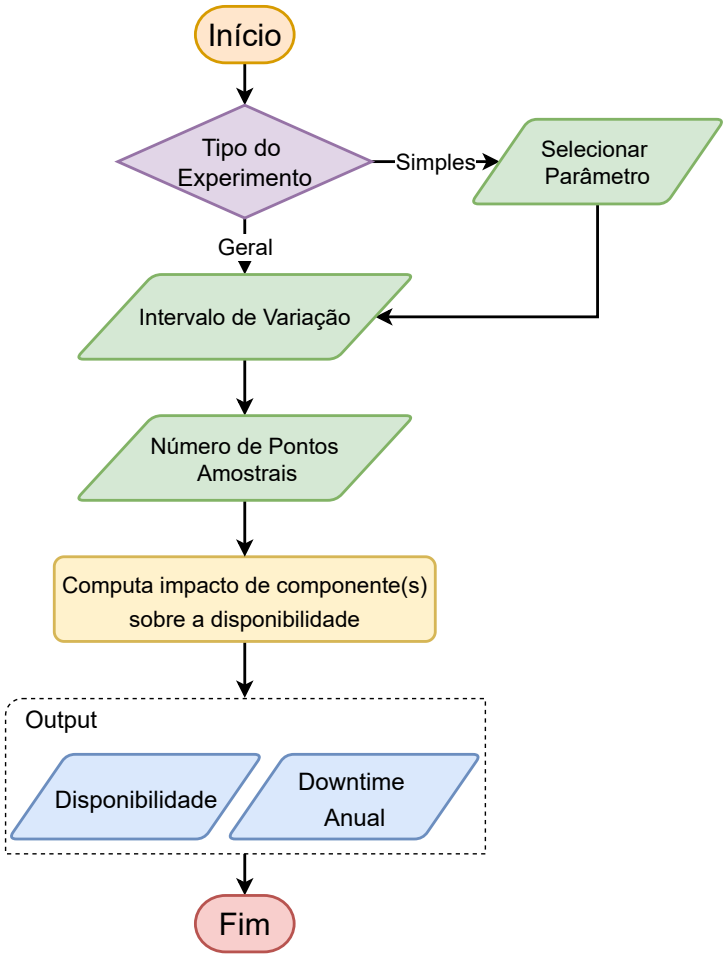
Por fim, o planejador de custos apresenta uma tabela comparativa entre as políticas de endossamento adotáveis e as demais métricas mencionadas. Essa tabela de custo também apresenta opções de filtro ao usuário da ferramenta, este que, mais uma vez, poderá alterar a grandeza da disponibilidade e do downtime anual, bem como o número total de casas decimais a direita.

6.4 PLANEJAMENTO DE EXPERIMENTOS

O planejador de experimentos permite ao interessado avaliar o impacto de um único componente (Experimento Simples) ou de um conjunto de componentes (Experimento Geral) sobre a disponibilidade geral ou o downtime anual da infraestrutura avaliada. Primeiramente, o usuário deverá selecionar o tipo de experimento desejado e depois estará apto a definir um número de pontos amostrais a serem exibidos em gráficos de linha. Quanto maior for o número de pontos amostrais definido pelo usuário, mais visível será a curva de impacto daquele componente sobre as métricas de interesse.

A Figura 26 apresenta o fluxograma do planejador de experimentos. É importante salientar que a etapa de definição do intervalo de variação diz repeito a escolha de valores mínimos e máximos a cada componente pertencente a infraestrutura no que diz respeito ao seu respectivo MTTF e MTTR. O número de servidores não é utilizado nesta análise, o que limita o cenário de experimentos a uma infraestrutura do tipo *baseline* ou com o número mínimo de componentes necessários para provimento do serviço.

Figura 26 – Fluxograma do Planejamento de Experimento



Fonte: Autoria Própria

6.5 CONSIDERAÇÕES FINAIS

Esta seção apresentou uma visão geral do Blockchain Provisioning Planning Tool (BPPT), uma ferramenta capaz de auxiliar no planejamento e na implantação de aplicações e infraestruturas baseadas em blockchain. O BPPT considera um conjunto de parâmetros associados a disponibilidades e custos gerais, e os compara ao das grandes provedoras de serviços de computação em nuvem. Mais detalhes da ferramenta proposta são apresentados no Apêndice C.1. A seguir, apresentamos um conjunto de estudos de caso que demonstram a viabilidade dos modelos e da ferramenta proposta.

7 ESTUDOS DE CASO

> *And Can You Offer Me Proof Of Your Existence? How Can You, When Neither Modern Science Nor Philosophy Can Explain What Life Is?*
>
> — Puppet Master, *Ghost in the Shell*, 1995

Esta seção apresenta cinco estudos de caso que tentam demonstrar a viabilidade do processo de avaliação de desempenho, da utilização dos modelos propostos e, do uso do BPPT como um mecanismo capaz de auxiliar o processo de planejamento de infraestruturas computacionais capazes de prover e hospedar serviços e aplicações baseados em blockchain.

O primeiro estudo de caso é caracterizado pela avaliação de desempenho através de medição, tanto de métricas inerentes à plataforma em si, como latência e vazão, quanto métricas relacionadas ao consumo de recursos (RAM, Disco e CPU).

No segundo estudo de caso, estendemos o primeiro estudo através de uma avaliação de performabilidade do sistema, mostrando o impacto do consumo de recursos sobre a disponibilidade geral da plataforma.

O terceiro estudo de caso apresenta a avaliação de dependabilidade sem o impacto causado pelo consumo de recursos, primeiramente obtemos a confiabilidade do sistema através de alterações no modelo de performabilidade proposto e depois a avaliação dos modelos de disponibilidade apresentados na seção 5, visando um conjunto de infraestruturas com diferentes características, como o número total de servidores a disposição, políticas de endossamento utilizadas, e número mínimo de componentes necessários a operação do sistema.

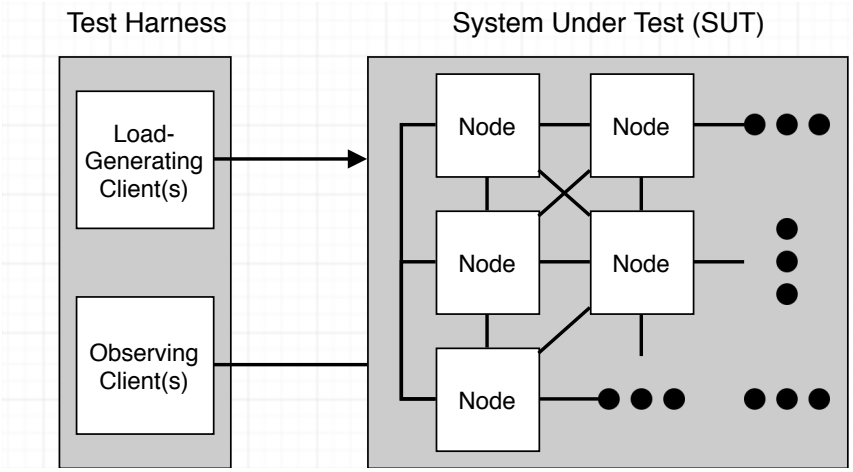
Já o quarto estudo de caso lida com os custos associados a manutenção da infraestrutura, determinando um conjunto de cenários estabelecidos através de acordo de nível de serviço.

Por fim, o quinto e último estudo de caso apresenta um comparativo entre os custos necessários à implantação e operação de uma série de infraestruturas. Além disso, realizamos um comparativo com as principais plataformas públicas que acobertam o uso do Hyperledger Fabric.

7.1 ESTUDO DE CASO I - AVALIAÇÃO DE DESEMPENHO

A Figura 27 apresenta uma visão de alto nível do processo de avaliação de desempenho aplicado ao Hyperledger Fabric. Esta figura mostra uma rede blockchain, composta de 1 a N nós (SUT), que recebe a carga de trabalho submetida por um ou mais clientes (Load Generating Client), podendo ser monitorada pelo mesmo cliente ou por um outro dispositivo (Observing Client).

Figura 27 – Benchmark aplicado a uma rede Blockchain



Fonte: (HYPERLEDGER, 2018a)

O **Test Harness** corresponde a todo o conjunto de software e hardware utilizado para geração e monitoramento da carga de trabalho (HYPERLEDGER, 2018a). Dentro do Test Harness há o **observer client**, um tipo de nó que recebe notificações sobre o SUT e pode ser utilizado para submissão de transações.

Nesta tese utilizamos do Caliper Benchmark para configuração e aplicação de nossa carga de trabalho ao SUT, bem como para realização do monitoraramento das transações submetidas ao sistema. Já o processo de monitoramento de recursos como CPU, RAM e Disco, foi realizado através da utilização de scripts bash para melhor gerenciamento dos recursos.

A Tabela 10 apresenta as configurações do nosso servidor, ou seja, do SUT responsável pelo atendimento das transações submetidas. Já a Tabela 11 apresenta o nosso Test Harness, composto por uma única máquina responsável pela submissão e monitoramento de transações e recursos do SUT.

Tabela 10 – System Under Test - Configurações

System Under Test (SUT)	
OS	Ubuntu 18.04 LTS
PROCESSOR	Xeon E3-1220V3
MEMORY	32 GB
STORAGE	500 GB HDD
HYPERLEDGER FABRIC	v1.4.1
DOCKER ENGINE	v18.09
DOCKER COMPOSE	v3.7

Fonte: Autoria Própria

Tabela 11 – Load Generation Client - Configurações

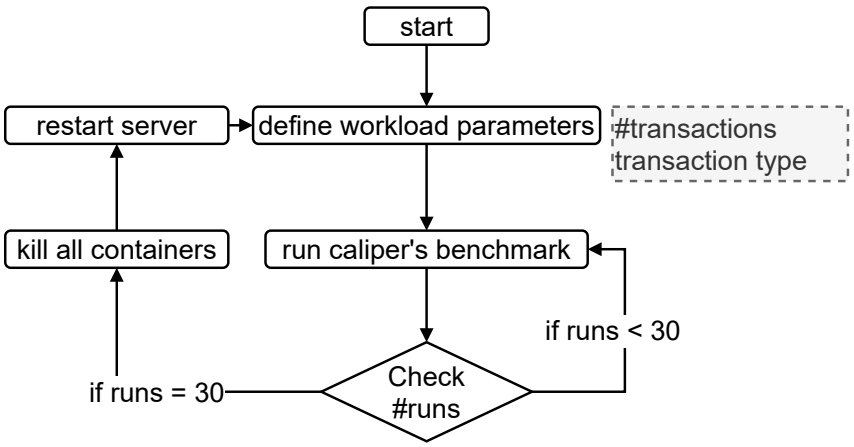
Load Generator Client	
OS	Ubuntu 18.04 LTS
PROCESSOR	Intel Core i5-3570K
MEMORY	8 GB
STORAGE	500 GB HDD
CALIPER	v0.2.0
NODE.JS	v10.24

Fonte: Autoria Própria

7.1.1 Resultados Obtidos

Para a carga de trabalho, escolhemos uma aplicação básica, esta presente no âmbito do próprio Hyperledger Caliper, tal aplicação consistia em um micro-ambiente bancário, com abertura de contas, consultas e transferência de valores. Para maiores informações sobre o benchmark utilizado, consultar o seu repositório no GitHub[1]. A versão do Hyperledger Fabric utilizada foi a 1.4.1. A Figura 28 mostra uma visão geral da carga de trabalho aplicada ao SUT.

Figura 28 – Visão Geral do Experimento



Fonte: Autoria Própria

O primeiro passo da carga de trabalho consiste na definição dos parâmetros. Para a nossa avaliação, trabalhamos com três tipos diferentes de transação: (i) Transferência entre Contas; (ii) Consulta e; (iii) Criação de Contas. Algumas informações adicionais sobre o experimento são listadas a seguir:

- Um número fixo de transações para cada rodada de transações foi definido (10 mil transações) por ser considerado suficiente;

[1] <https://github.com/hyperledger/caliper-benchmarks/tree/master/benchmarks/scenario/simple>

- Por simplicidade a taxa de envio de transações ao sistema foi constante para cada rodada de experimento.

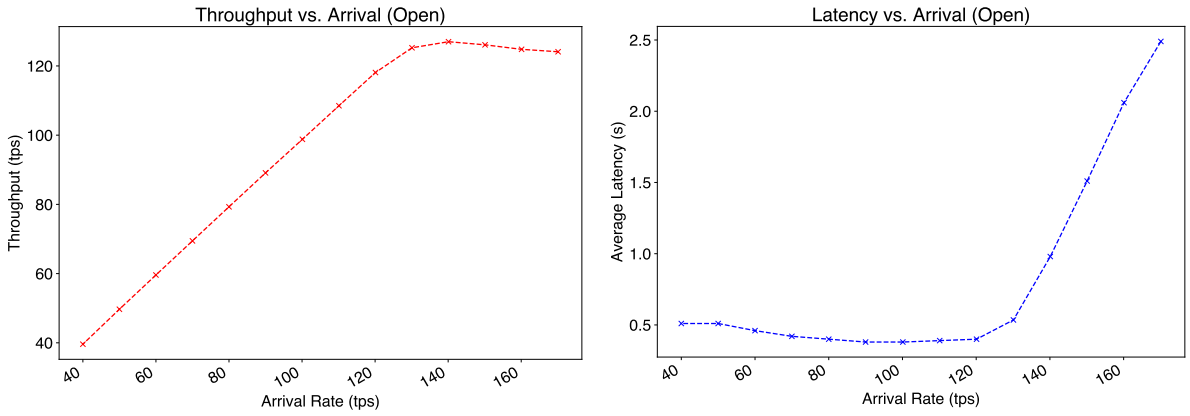
O segundo passo da nossa avaliação de desempenho foi a execução do Caliper benchmark por trinta rodadas, para cada combinação entre tipo de transação e taxa de submissão. Após completar as 30 execuções, o próprio Hyperledger Caliper encerrava os contêineres e então o SUT era reiniciado.

7.1.1.1 Vazão e Latência

Um total de 390 rodadas foram executadas para a transação do tipo abertura de contas, a cada trinta rodadas de execuções nós aumentamos a taxa de submissão por parte do Caliper em um total de 10 transações por unidade de tempo. Desse modo, o que foi iniciado a partir de 30 transações submetidas por segundo, depois 40, 50, e assim sucessivamente, encerrou-se com uma taxa de submissão na ordem de 170 transações por segundo (TPS).

As Figuras 29a e 29b apresentam, respectivamente, a vazão e a latência média das transações em segundos que foram obtidas através do monitoramento das transações do tipo abertura de contas.

Figura 29 – Abertura de Contas



(a) Vazão Média (tps)

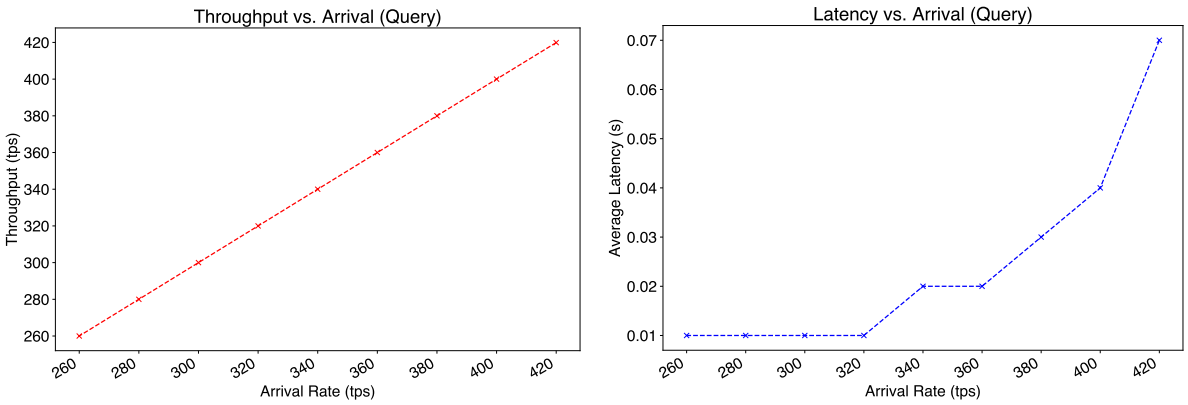
(b) Latência Média (s)

Fonte: Autoria Própria

No cenário de abertura de contas, a vazão cresce conforme aumentamos a taxa de envio de transações ao sistema, ao menos até atingirmos uma taxa de envio na ordem de 130 transações por segundo. A partir desse ponto, o SUT se mostra sobrecarregado e não há mais crescimento. Do mesmo modo, a latência em segundos a partir do momento que passamos a barreira das 130 transações por segundo começa a crescer quase que exponencialmente, o que é mais um indicativo de sobrecarga do nosso servidor.

Para as transações do tipo consulta, aproximadamente 180 execuções foram realizadas. Neste cenário, incrementamos a taxa de submissão em 20 transações por unidade de tempo a cada 30 rodadas, pois notou-se que o seu impacto sobre os recursos do sistema eram mínimos para os valores iniciais. O experimento que teve início com 260 transações chegou a um total de 420. As Figuras 30a e 30b apresentam, respectivamente, a vazão e a latência em segundos para as transações de consulta de contas.

Figura 30 – Consulta de Contas



(a) Vazão Média (tps)

(b) Latência Média (s)

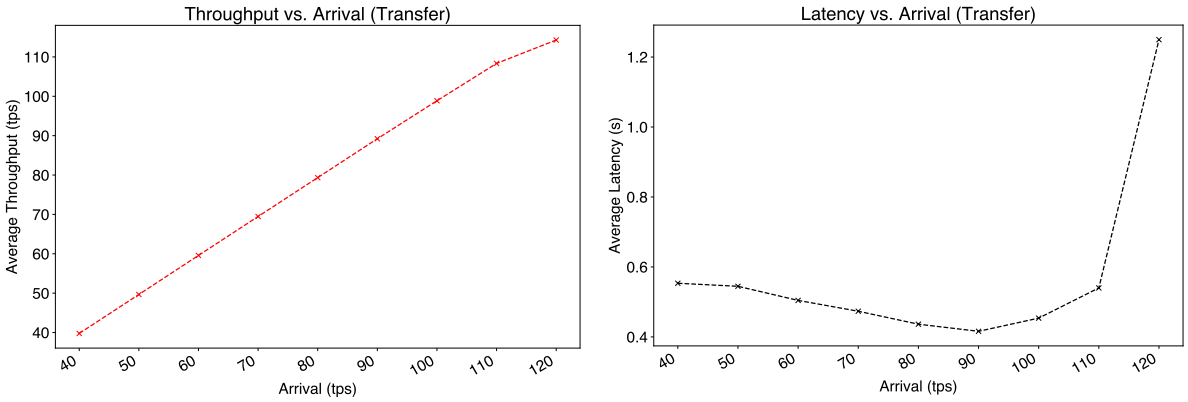
Fonte: Autoria Própria

A transação de consulta de contas apresenta uma relação de 1:1, constante ao longo de todo o processo de experimentação. Já a latência para este tipo de transação, por mais que apresente uma alta de 700% ao longo dos cenários avaliados, ainda é de menos de 1 segundo em seu tempo total.

O último tipo de transação avaliado foi o de transferência entre contas. Este cenário é caracterizado por incremento de 10 operações a taxa de submissão a cada 30 rodadas e estende um experimento iniciado com submissão de 40 transações por segundo a um total de 120. Identificamos neste cenário um aumento considerável no consumo de recursos e, a mesma abordagem utilizada no cenário anterior não poderia ser aplicada quando lidando com a transação de transferência entre contas. As Figura 31a e 31b apresentam, respectivamente, a vazão e a latência em segundos para a operação de transferência entre contas.

Em um comportamento similar ao apresentado pelas transações do tipo abertura de contras, as transações do tipo transferência entre contas começam a estagnar a partir de 120 transações submetidas por segundo ao SUT, antes disso, também apresentavam relação 1:1, alertando para possível sobrecarga do servidor. A latência em segundos também apresenta um claro crescimento. A seguir apresentamos os resultados relacionados ao monitoramento de recursos do SUT com base nos tipos de transação a ele submetidos.

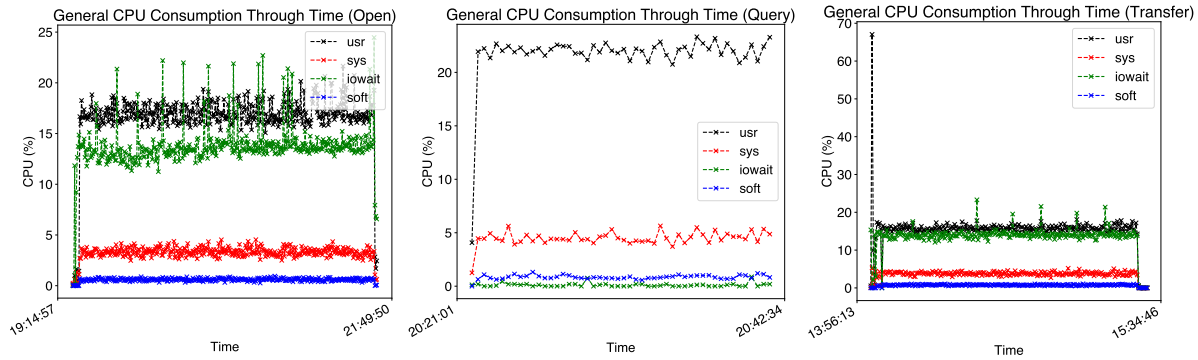
Figura 31 – Consulta de Contas



(a) Vazão Média (tps) (b) Latência Média (s)

Fonte: Autoria Própria

Figura 32 – Consumo de CPU por tipo de transação



(a) Abertura - CPU (b) Consulta - CPU (c) Transferência - CPU

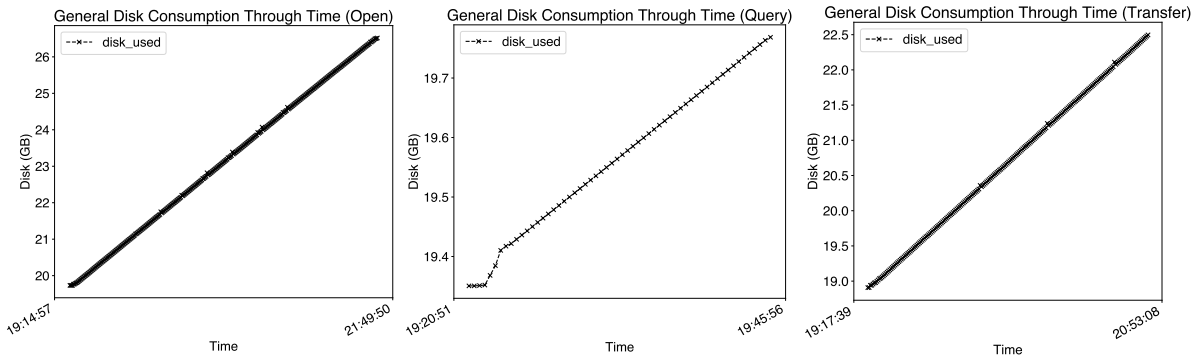
Fonte: Autoria Própria

7.1.1.2 Consumo de Recursos

O monitoramento de recursos foi realizado através de scripts bash. Os recursos monitorados foram CPU, RAM, Disco e Memória em Cache, esta que é alocada pelo Linux para operações de caching. Como resultados iniciais podemos apontar indícios de sobrecarga do servidor.

Para o consumo de recursos resolvemos apresentar os resultados no pico de transações submetidas e assim apresentamos a real ocupação dos recursos presentes em nosso SUT. Iniciamos pela CPU, que é significativamente estressada no processo de realização de transações. Porém, há espaço para mais, dando um indicativo de que o gargalo do sistema está em outro recurso e provavelmente em recurso de armazenamento volátil. As Figuras 32 apresentam o consumo de CPU por cada tipo de transação submetida ao SUT.

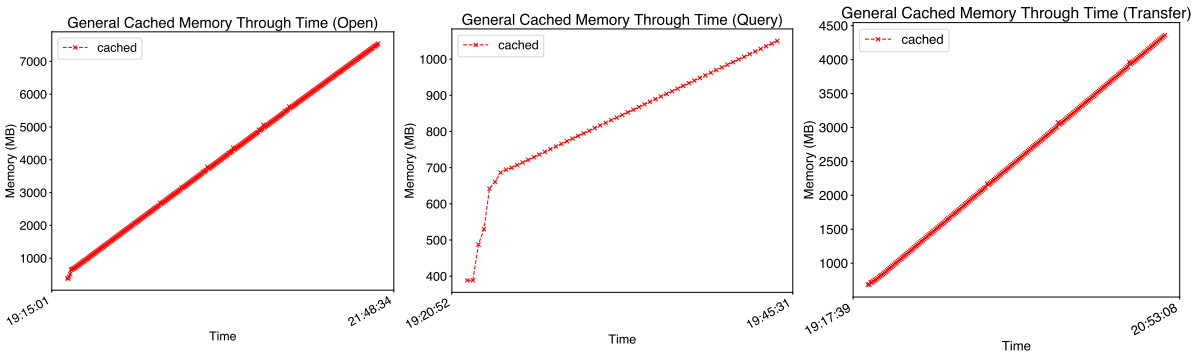
O consumo de disco apresentou uma alta considerável para o período de realização do



(a) Abertura - Disco (b) Consulta - Disco (c) Transferência - Disco

Figura 33 – Consumo de Disco por tipo de Transação

Figura 34 – Recursos em Cache para cada tipo de transação



(a) Abertura - Cached (b) Consulta - Cached (c) Transferência - Cached

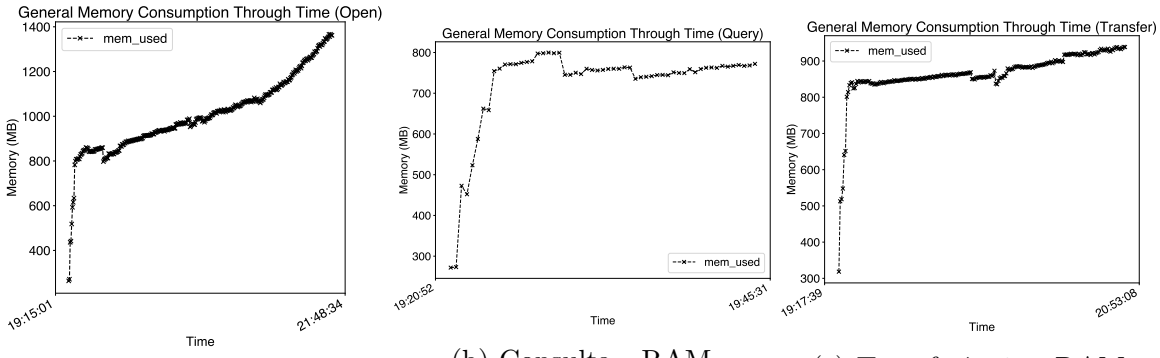
Fonte: Autoria Própria

experimento. Tal acréscimo diz respeito a memória em cache alocada pelo Linux para as operações que lidam sobre arquivos. Por ser o dispositivo de maior lentidão na hierarquia de memória, podemos caracterizar este componente como o provável gargalo do sistema. A exaustão de recursos de disco a longo prazo acaba sendo uma realidade do ambiente avaliado. Tomemos por exemplo a operação de abertura de contas, que em menos de 2 horas e 30 minutos apresentou um acréscimo de mais de 7 GB no consumo desse recurso. A Figura 33 apresenta o consumo de disco por cada tipo de transação.

Como extensão ao disco, a memória em cache que é alocada pelo sistema operacional, apresenta um alto consumo. Os gráficos das Figuras 34 apresentam como principal diferença entre os previamente apresentados o início da contagem a partir do ponto zero, indo ao respectivo pico para cada tipo de transação realizada.

O último recurso monitorado foi a memória principal. Mais uma vez, a transação do tipo abertura de contas apresentou o maior consumo de recurso. O consumo de RAM por este tipo de transação cresceu em 80% em menos de três horas, o que poderia caracterizar vazamento de recursos proveniente de um processo de envelhecimento de software, este acelerado pela alta carga de trabalho aplicada. As Figuras 35 apresentam o consumo de

Figura 35 – Consumo de RAM por tipo de transação



(a) Abertura - RAM (b) Consulta - RAM (c) Transferência - RAM

Fonte: Autoria Própria

RAM para cada tipo de transação realizada.

7.1.2 Limitações

O indicativo de envelhecimento não pode ser negado, porém não possuímos insumos suficientes para alegarmos que a plataforma em si passa por este fenômeno. Há trabalhos que apontam a presença de envelhecimento de software tanto na plataforma Docker que executa e gerência os contêineres do Hyperledger Fabric, quanto no kernel de sistemas baseados em Linux. Porém, resolvemos estender o presente estudo de caso para avaliarmos o impacto do crescimento no consumo de recursos sobre a disponibilidade do serviço prestado.

7.2 ESTUDO DE CASO II - PERFORMABILIDADE

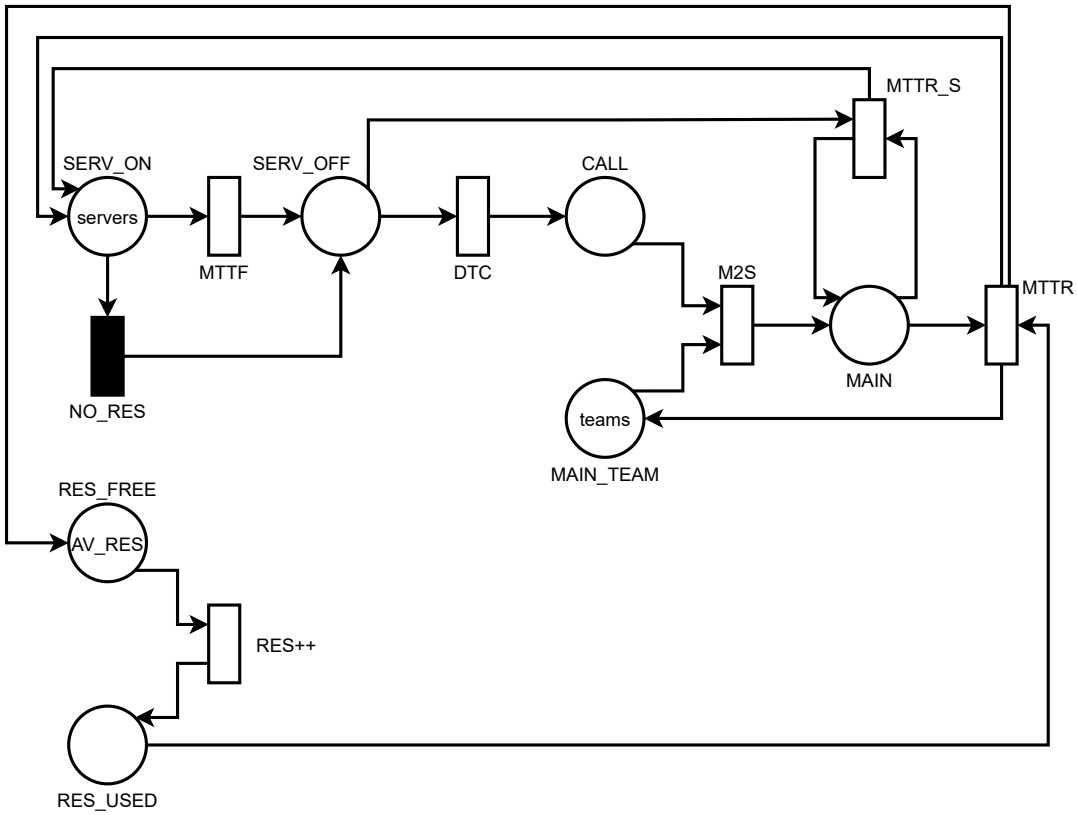
O segundo estudo de caso apresenta a avaliação do impacto causado pelo crescimento no consumo de recursos à disponibilidade geral da infraestrutura básica. Os recursos considerados de alto impacto foram o disco e a RAM, tal conjectura é resultante do estudo de caso anterior. Para esta avaliação desenvolvemos um modelo de performabilidade que estende o modelo de provimento de serviço com restrições.

7.2.1 Modelo de Performabilidade

Como mencionado, o modelo proposto representa um ambiente com um número finito de recursos e que está sob estresse provocado por alta carga de trabalho. Fizemos uso de uma rede de Petri estocástica para modelagem deste sistema, apresentado através da Figura 36.

A operacionalidade do sistema é dada pela presença de uma marcação no lugar SERV_ON. É importante salientar ainda que com base no estudo de caso anterior, onde

Figura 36 – Modelo de Performabilidade



Fonte: Autoria Própria

diagnosticou-se um aumento no consumo de recursos, resolvemos adicionar uma transição imediata, que corrobora com a possibilidade de falha do serviço por exaustão de recursos, o que é indicado pela transição NO_RES.

Na presença de vazamento de recursos, nós estimamos o tempo que levaria para que os recursos sejam totalmente exauridos com base nos experimentos realizados. Distribuímos esse tempo na transição temporizada (RES++).

A falha natural do provimento do serviço, ou seja, ocasionada por falha do servidor é representada pela transição MTTF. Outro importante lugar do modelo é o RES_FREE que apresenta o número de recursos disponível. Quando não há mais recursos disponíveis a transição imediata NO_RES é disparada, movendo a marcação de SERV_ON para SERV_OFF.

7.2.2 Resultados Obtidos

Para avaliação da disponibilidade através do modelo de performabilidade, parâmetros de entrada são necessários. Alguns dos valores para os respectivos parâmetros foram obtidos através de revisão da literatura, em trabalhos como (MELO et al., 2021; PEREIRA et al., 2021; SEBASTIO; GHOSH; MUKHERJEE, 2018), enquanto que outros são resultados de

nosso estudo de caso anterior. A Tabela 12 apresenta os valores de entrada utilizados para avaliação de disponibilidade do modelo de performabilidade.

Tabela 12 – Parâmetros de Entrada para Avaliação de Disponibilidade

Transição	**Tempo (h) ou Condição**
MTTF	308.45
MTTR_S	1
MTTR	1
DTC	0.5
M2S	0.5
RES++	Número de Recursos
NO_RES	RES_FREE = 0

Fonte: Autoria Própria

A métrica relacionada ao consumo de recursos é dada em GB/h para ambos RAM e disco. Esse cenário é uma estimativa e toma como base o monitoramento dos recursos previamente realizado para operações de transferência entre contas, abertura e consulta de contas. É importante destacar que o SUT ao qual aplicamos a carga de trabalho possui 500 GB de disco e 32 GB de RAM e que nossa previsão consiste em iniciá-los a partir de 0GB, ou seja, abstraímos uma complexidade e desconsideramos o consumo por parte do sistema operacional, kernel e componentes associados. A Tabela 13 apresenta o consumo por hora para cada tipo de transação.

Tabela 13 – Consumo de Recursos por Hora

Cenário	**Consumo por Hora**
Opening Accounts - Disk	3 GB
Opening Accounts - RAM	0,24 GB
Query - Disk	2 GB
Query - RAM	0,1 GB
Transfer - Disk	1,67 GB
Transfer - RAM	0,2 GB

Fonte: Autoria Própria

Com base nos parâmetros de entrada e do consumo de recursos por hora previamente apresentados, fomos capazes de avaliar o modelo proposto. A Tabela 14 apresenta a disponibilidade geral alcançada por um conjunto de cenários considerando a presença de vazamento de recursos na infraestrutura.

A avaliação do modelo considerando a possibilidade de exaustão de recursos provê uma disponibilidade abaixo de dois noves para todos os cenários avaliados. A transação

Tabela 14 – Resultados Gerais de Disponibilidade

Cenário	**Disponibilidade (%)**	**Downtime Anual (h)**
Abertura - Disco	98,48	133,58
Abertura - RAM	98,17	160,30
Consulta - Disco	98,88	98,12
Consulta - RAM	98,99	88,47
Transferência - Disco	98,57	125,26
Transferência - RAM	98,41	139,28

Fonte: Autoria Própria

do tipo abertura de contas provê o menor valor de disponibilidade registrado (99,17%), mostrando o impacto da alta carga de transações sobre o consumo da memória RAM. É importante ressaltar que no âmbito do modelo proposto, no processo de ocorrência de uma manutenção e consequentemente de um reparo do sistema. Todavia, o recurso utilizado é devolvido ao sistema, reiniciando o processo de consumo.

As transações do tipo consulta apresentaram o menor impacto geral sobre a disponibilidade do sistema dentre as operações avaliadas. Em segundo lugar, em termos de impacto causado estão as operações de transferência entre contas, que para o caso de vazamento de recursos de RAM pode apresentar um downtime anual de aproximadamente 140 horas, ou quase 6 dias.

7.2.3 Limitações

A principal limitação do presente modelo é a sua não generalização. O modelo não é capaz de lidar com ambientes maiores, onde os recursos estariam presentes individual para cada máquina, levando a uma explosão de estados onde a avaliação seria possível apenas através de simulação. Deste modo, o presente estudo de caso acaba sendo aplicável apenas ao cenário básico, onde um único servidor hospeda a aplicação e os componentes da plataforma Hyperledger Fabric.

Uma outra possibilidade de análise é a instância de um novo bloco para o outro tipo de recurso avaliado, um para a RAM e outro para disco. Atualmente, nossa análise considera um único recurso por vez, no passo que ambos são consumidos simultaneamente pelo sistema.

Por fim, a não combinação entre tipos distintos de transação no processo de avaliação do modelo pode enviesá-lo para determinado tipo de transação. Sabemos que na maioria dos sistemas convencionais consultas ocorrem com mais frequência do que operações de cadastro e transferência de recursos.

7.3 ESTUDO DE CASO III - AVALIAÇÃO DE DISPONIBILIDADE E CONFIBIABILIDADE

Esta subseção apresenta o terceiro estudo de caso, onde será avaliado um conjunto de cenários distintos. Cada cenário, também está disposto sobre políticas distintas de endossamento. Tal avaliação é de suma importância para estabelecermos uma visão macro do funcionamento de nossa plataforma, bem como da interação entre os componentes que a compõe.

Nossas principais métricas de interesse são a confiabilidade e a disponibilidade do sistema (infraestrutura). É importante salientarmos que não consideramos a inclusão de uma aplicação baseada em blockchain no contexto avaliativo, apenas o provimento da plataforma Hyperledger Fabric como serviço.

7.3.1 Parâmetros de Entrada

O primeiro passo para avaliação das métricas de interesse foi o estabelecimento de valores de entrada para os modelos propostos. A disponibilidade do sistema foi avaliada através do BPPT, e os valores utilizados foram extraídos através de revisão da literatura. Todavia, outros valores foram obtidos diretamente pela avaliação de outros modelos, sendo assim, são resultados derivados da modelagem hierárquica. Este é o caso do RBD que representa o nível superior da infraestrutura computacional (Ver Seção 5), composto essencialmente por Hardware, Sistema Operacional e Docker Engine. A Tabela 15 apresenta os valores de MTTF e MTTR em horas que foram utilizados como entrada em nossas avaliações.

Tabela 15 – Parâmetros de Entrada para Avaliação de Dependabilidade

Component	**MTTF (h)**	**MTTR (h)**
Hardware (HW)	8760	1.66
Operating System (OS)	2893	0.15
Docker Engine (DE)	2516	0.15
MSP-END-ORD	1258	0.15

Fonte: Autoria Própria

Os tempos de falha e reparo para o hardware e o sistema operacional foram extraídos de (DANTAS, 2013; MELO et al., 2016). No âmbito do reparo de componentes de hardware consideramos a possibilidade da troca ou substituição desse equipamento caso seja detectada uma falha fatal do mesmo. Para o sistema operacional, consideramos como reparo o tempo necessário a detecção de falha e o reinício do servidor. É uma tendência de que a maioria das falhas de sistema operacional sejam intermitentes e que sempre que ocorram sejam automaticamente armazenadas as causas em seu respectivo log. Todavia, quando essas falhas tornam-se frequentes podem estar associadas a problemas de *drivers*

ou até mesmo do próprio hardware. Deste modo, o tempo de reparo definido aqui é uma estimativa.

Para os contêineres e o Docker Engine os tempos de falha foram extraídos de (SEBASTIO; GHOSH; MUKHERJEE, 2018), já os reparos foram estimados. Todavia, em (SEBASTIO; GHOSH; MUKHERJEE, 2018) os autores consideram o valor de reparo de contêineres como o tempo necessário para o seu reinício, este valor tende a ser pequeno o suficiente para um grande centro de dados, mas foge a realidade de uma infraestrutura menor e privada e que requer detecção de problemas de forma usualmente reativa.

Já para os cenários de avaliação de confiabilidade, não utilizamos tempos associados ao reparo, apenas dos respectivos tempos esperados até a ocorrência de uma falha em cada componente da infraestrutura.

7.3.2 Cenários de Avaliação

Construímos dez diferentes cenários para avaliação. Esses cenários são apresentados na Tabela 16.

Tabela 16 – Cenários Propostos

Cenário	**Política de endossamento**	**#Min. de Servidores**	**#Servidores**
1	Baseline	1	1
2	AND	2	2
3	AND	3	3
4	AND	4	4
5	OR	1	2
6	OR	1	3
7	OR	1	4
8	KooN	2	3
9	KooN	2	4
10	KooN	3	4

Fonte: Autoria Própria

Cada cenário considera uma possível combinação de política de endossamento e quantidade de servidores. O primeiro cenário é dito *baseline*, haja vista que uma infraestrutura contendo um único servidor pode atender todas as diferentes políticas de endossamento, seja esta AND, OR ou K-out-of-N.

Do segundo cenário em diante, variamos o número total de servidores dentro da infraestrutura de 1 a 4. Outro fator, este dependente da política de endossamento aplicada, é o número mínimo (#Min. de Servidores) de servidores requeridos para realização do endossamento. Note que, para políticas de endossamento do tipo AND, esperamos que todos os servidores estejam operacionais, já as do tipo OR que ao menos um deles esteja.

Já para os cenários sob a política de endossamento do tipo K-out-of-N, variamos esse número mínimo em combinação ao número de servidores providos. Sendo importante ressaltar, que um cenário disposto de 4 servidores em operação dentre 4 disponíveis, ou mesmo para 3 de 3, 2 de 2 e 1 de 1, equivale a políticas do tipo AND, e já foram contemplados por cenários anteriores.

Uma outra importante característica a se destacar é que para cada servidor operacional devemos ter em operação os três contêineres inerentes ao provimento de uma infraestrutura baseada em Hyperledger Fabric (Endorsers, Orderers e MSP).

7.3.3 Resultados Obtidos

Após listarmos os valores de entrada utilizados e definirmos o conjunto de cenários para avaliação, poderemos avaliar a disponibilidade e a confiabilidade da infraestrutura. Vale ressaltar, que para esta etapa, a ferramenta Mercury foi utilizada em conjunto do planejador de disponibilidade fornecido pelo BPPT.

7.3.3.1 Avaliação de Disponibilidade

O planejador de disponibilidade apresenta os resultados de disponibilidade e downtime anual para as políticas de endossamento avaliadas (OR, AND ou KooN). Para cada cenário avaliado, capturamos os valores providos na tabela resumo do BPPT. Esses valores são apresentados na Tabela 17, que resume os índices de disponibilidade e downtime anual obtidos.

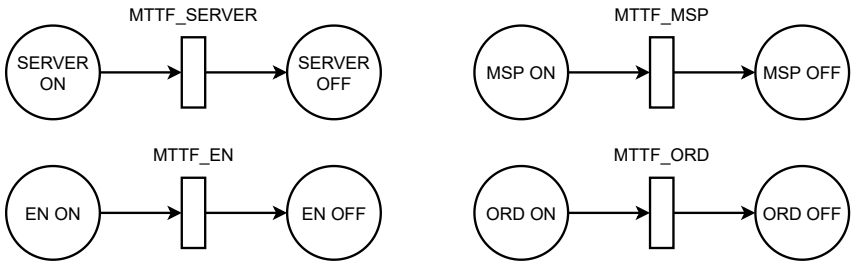
Tabela 17 – Resultados Gerais de Disponibilidade

Cenário	**Disponibilidade(%)**	**Disponibilidade(#9s)**	**Downtime Anual(h)**
1	99,9341	3,18	5,77
2	99,8683	2,88	11,53
3	99,8026	2,70	17,29
4	99,7369	2,58	23,05
5	99,9998	6,36	0,0038
6	99,9999999	9,54	0,000003
7	99,9999999999	12,73	0,00000002
8	99,9998	5,89	0,011
9	99,9999	8,94	0,000009
10	99,9997	5,59	0,022

Fonte: Autoria Própria

Como podemos ver, a disponibilidade de uma infraestrutura sob a política de endossamento do tipo AND (cenários 2 a 4) é reduzida conforme novos recursos (servidores) são adicionados ao ambiente. Tal comportamento é visto com mais facilidade através da

Figura 37 – Modelo de Confiabilidade



Fonte: Autoria Própria

coluna de downtime anual, onde iniciamos no cenário base com 5.77 horas de downtime anual e chegamos a mais de 23 horas no cenário com 4 servidores sobre a política de endossamento do tipo AND. Isto ocorre devido ao requisito de operacionalidade estabelecido para esta política, que demanda, ou obriga, que todos os servidores daquela infraestrutura estejam em pleno estado de funcionamento e de posse de todos os seus respectivos contêineres para que uma determinada transação seja completamente endorsada.

Quando consideramos uma política de endossamento do tipo OR (cenários 5, 6 e 7) visualizamos um comportamento inversamente proporcional aquele obtido de uma política de endossamento baseada em AND. Olhando para o downtime anual do cenário 5, já iniciamos com alguns poucos segundos de indisponibilidade e que tende a zero conforme novos servidores são acrescidos a nossa infraestrutura. Esse comportamento é esperado, haja vista que a dependência para esta nova política é de que pelo menos um dentre os servidores esteja operacional e realize o endossamento das transações.

Por último e não menos importante, a avaliação de cenários sob a política de endossamento do tipo K-out-of-N. Esta política de endossamento apresenta valores de disponibilidade considerados um *meio termo* se comparados aos obtidos pelas demais políticas avaliadas. É um fato que sua disponibilidade cresce conforme aumenta o número de servidores a disposição. Todavia, ela diminui conforme cresça a dependência de que mais servidores precisem estar operacionais para prestar o respectivo endossamento.

7.3.3.2 Avaliação de Confiabilidade

Para avaliação de confiabilidade das infraestruturas presentes nos cenários apresentados, reutilizamos o modelo de performabilidade proposto, ao excluir o consumo de recursos bem como a possibilidade de reparo do ambiente através da remoção da marcação presente no lugar MAIN_TEAM. Na falta de uma equipe de manutenção não haverá reparo do sistema em questão. Deste modo, a Figura 37 apresenta o modelo de confiabilidade proposto.

Para o cálculo de confiabilidade do sistema sob a perspectiva da política de endossamento estabelecemos a probabilidade para cada condição relacionada a cada política

ser verdadeira. Por exemplo, em um sistema com três nós, n = 3, sob uma política de endossamento do tipo AND, é necessário que todos os seus componentes estejam operacionais, possuindo 3 marcações nos lugares SERVER_ON, EN_ON, MSP_ON e ORD_ON, incluindo assim, todos os servidores e os contêineres a eles associados, deste modo a confiabilidade poderá ser dada pela Expressão 7.1, onde #P é indicativo de Probabilidade associada.

$$\#P_{\mathrm{AND}} = (\#P_{\mathrm{SERVER_{ON}}} = n) \times (\#P_{\mathrm{ORD_{ON}}} = n) \times (\#P_{\mathrm{EN_{ON}}} = n) \times (\#P_{\mathrm{MSP_{ON}}} = n) \tag{7.1}$$

Mas, no caso de uma política de endossamento do tipo OR, precisaríamos que apenas um estivesse operante, como indicado pela Equação 7.2. Enquanto que em um cenário mais característico a uma política de endossamento do tipo KooN, poderíamos precisar que ao menos 2 dos 3 componentes esteja operando, ou seja 2oo3, como pode ser visto pela Expressão 7.3.

$$\#P_{\mathrm{1oo3}} = (\#P_{\mathrm{SERVER_{ON}}} >= 1) \times (\#P_{\mathrm{ORD_{ON}}} >= 1) \times (\#P_{\mathrm{EN_{ON}}} >= 1) \times (\#P_{\mathrm{MSP_{ON}}} >= 1) \tag{7.2}$$

$$\#P_{\mathrm{2oo3}} = (\#P_{\mathrm{SERVER_{ON}}} >= 2) \times (\#P_{\mathrm{ORD_{ON}}} >= 2) \times (\#P_{\mathrm{EN_{ON}}} >= 2) \times (\#P_{\mathrm{MSP_{ON}}} >= 2) \tag{7.3}$$

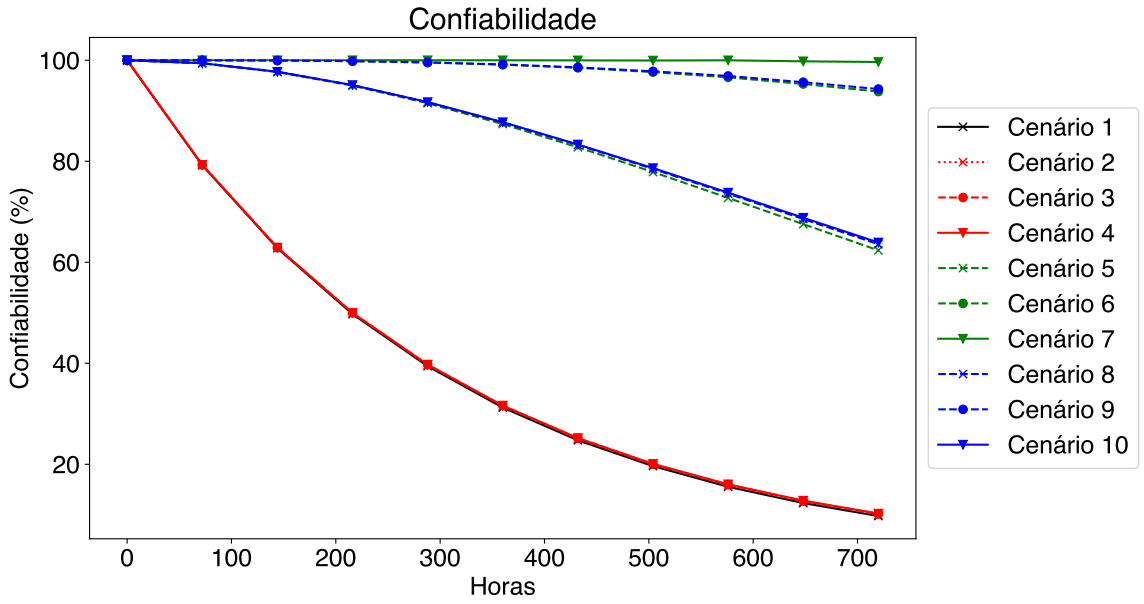
A Figura 38 apresenta os resultados obtidos através da avaliação de confiabilidade para os dez diferentes cenários apresentados na Tabela 16. Consideramos 11 pontos amostrais em um intervalo de 30 dias ou 720 horas, o primeiro deles inicia-se no ponto zero, onde a confiabilidade do sistema é igual a 1 ou 100%.

Os resultados obtidos apontam que o cenário de número 7, sob uma política de endossamento do tipo OR, apresenta o melhor valor para confiabilidade do sistema, muito próximo a 100%, superando a marca de seis noves. Enquanto que o cenário 1, baseline, e os demais cenários sob uma política de endossamento do tipo AND, apresentam os piores resultados, com tendência de falha para o intervalo de 30 dias. Os cenários KooN apresentaram-se, mais uma vez, como alternativa as duas outras políticas avaliadas.

7.3.4 Limitações

As principais limitações do presente estudo de caso giram em torno da possibilidade de combinações entre políticas de endossamento e da falta de uma aplicação específica em execução sobre o ambiente avaliado. Outro problema importante a ser destacado está na CTMC multinível, e está atrelado ao processo de manutenção que considera a existência de uma equipe de manutenção para cada servidor da infraestrutura. Deste modo, se dois servidores falham o meu serviço poderá contar com dois responsáveis pelo reparo imediato a uma taxa de reparo 2μ.

Figura 38 – Resultado da Análise de Confiabilidade para os Cenários Avaliados



Fonte: Autoria Própria

Em ambientes reais é possível definir níveis distintos e políticas variadas como, por exemplo, em um ambiente de três servidores um deles obrigatoriamente deverá fazer o endossamento prioritariamente antes da transação ser encaminhada aos outros.

7.4 ESTUDO DE CASO IV - AVALIAÇÃO DE DISPONIBILIDADE DOS MODELOS DE PROVIMENTO DE SERVIÇO COM RESTRIÇÕES

Para contornar as limitações relacionadas ao processo de manutenção apresentados por nossa CTMC multinível, propomos redes de Petri Estocásticas que consideram a presença de uma única ou várias equipes de manutenção que podem estar presentes no local do servidor ou se locomover até lá para realização dos devidos reparos. Além disso, propomos a possibilidade da adoção de mecanismos de autocorreção ou *self-healing* que são responsáveis pelo monitoramento e reparo do ambiente através de seu reinício.

7.4.1 Modelos de Provimento de Serviço com Restrições

Dois modelos de provimento de serviço com restrições serão avaliados, o primeiro deles lida com a manutenção do tipo reativa, enquanto que o segundo lida ainda com um mecanismo de autocorreção adicional.

A **manutenção do tipo reativa** (Ver Figura 20) depende da ocorrência de falha no provimento do serviço para que se possa chamar uma equipe de manutenção que irá corrigir ou mitigar o problema. Isto pode ser feito através da troca de um equipamento defeituoso, correção de bugs e problemas a nível de software. Nossa CTMC multinível lida, também, com esse mesmo tipo de problema (manutenção reativa), mas por limitação estrutural trabalhamos com n-equipes de manutenção para n-servidores.

Já no modelo com mecanismo de autocorreção ou **self-healing** combinado a uma manutenção do tipo reativa (Ver Figura 22) um software monitor é acrescido ao contexto do ambiente em uma máquina a parte e tenta detectar falhas nos servidores e aplicações através de consultas recorrentes. Em caso de detecção de falha o mesmo tentará reanimar o dispositivo através de um comando de *reboot*, se o resultado obtido for não satisfatório aguardamos o tempo administrativo e chamamos a equipe de manutenção.

7.4.2 Parâmetros de Entrada

Assim como no cenário de avaliação de disponibilidade através do BPPT (que inclui nossa CTMC multinível em seu código latente) um conjunto de valores de entrada é necessário para que possamos realizar a avaliação dos modelos com restrição. Os valores utilizados são estimativas e podem variar consideravelmente entre a localidade do centro de dados, tipo de manutenção e número de equipes de manutenção a disposição. A Tabela 18 apresenta todos os valores utilizados para avaliação dos dois modelos com restrição propostos.

O valor de MTTF apresentado corresponde ao tempo médio entre falhas do servidor em operação com seus respectivos contêineres para provimento de serviço em execução. Esse valor foi extraído do cenário de avaliação de disponibilidade através do BPPT e considerou um único servidor. Já o tempo administrativo foi estimado, assim como o

Tabela 18 – Parâmetros de Entrada para Avaliação dos Modelos de Manutenção

Transições	**Tempo (h)**
MTTF	308,45
DTC	0,5
M2S	0,5
MTTR	1
MTTF_MON	788,4
REBOOT	0,08
Transições	**Peso**
SUCCESS	0,8
FAIL	0,2

Fonte: Autoria Própria

respectivo MTTR do sistema e o tempo para a equipe de manutenção chegar ao local do serviço (M2S).

Já o tempo entre falhas para o monitor (MTTF_MON) foi extraído da literatura (DANTAS, 2018) e considera o tempo entre falhas de uma aplicação externa ao servidor, que pode estar alocada em um dispositivo móvel ou outro computador pessoal dentro da mesma rede local do centro de dados.

7.4.3 Cenários

Para o presente estudo de caso a construção de cenários consistiu no estabelecimento de acordos de nível de serviço (SLA) para a contínua operação da infraestrutura. Os acordos foram montados considerando um número variável de equipes de manutenção e de recursos (servidores em operação). A Tabela 19 apresenta os cenários avaliados pelo presente estudo de caso.

Tabela 19 – Cenários Avaliados

Cenário	**Modelo com Restrições**	**#Recursos**	**#Equipes de Manutenção**
1	Reativo	1	1
2	Reativo	2	1
3	Reativo	3	1
4	Reativo	4	1
5	Reativo	2	2
6	Reativo	3	2
7	Reativo	4	2
8	Reativo	3	3
9	Reativo	4	3
10	Reativo	4	4
11	Self Healing + Reativo	1	1
12	Self Healing + Reativo	2	1
13	Self Healing + Reativo	3	1
14	Self Healing + Reativo	4	1
15	Self Healing + Reativo	2	2
16	Self Healing + Reativo	3	2
17	Self Healing + Reativo	4	2
18	Self Healing + Reativo	3	3
19	Self Healing + Reativo	4	3
20	Self Healing + Reativo	4	4

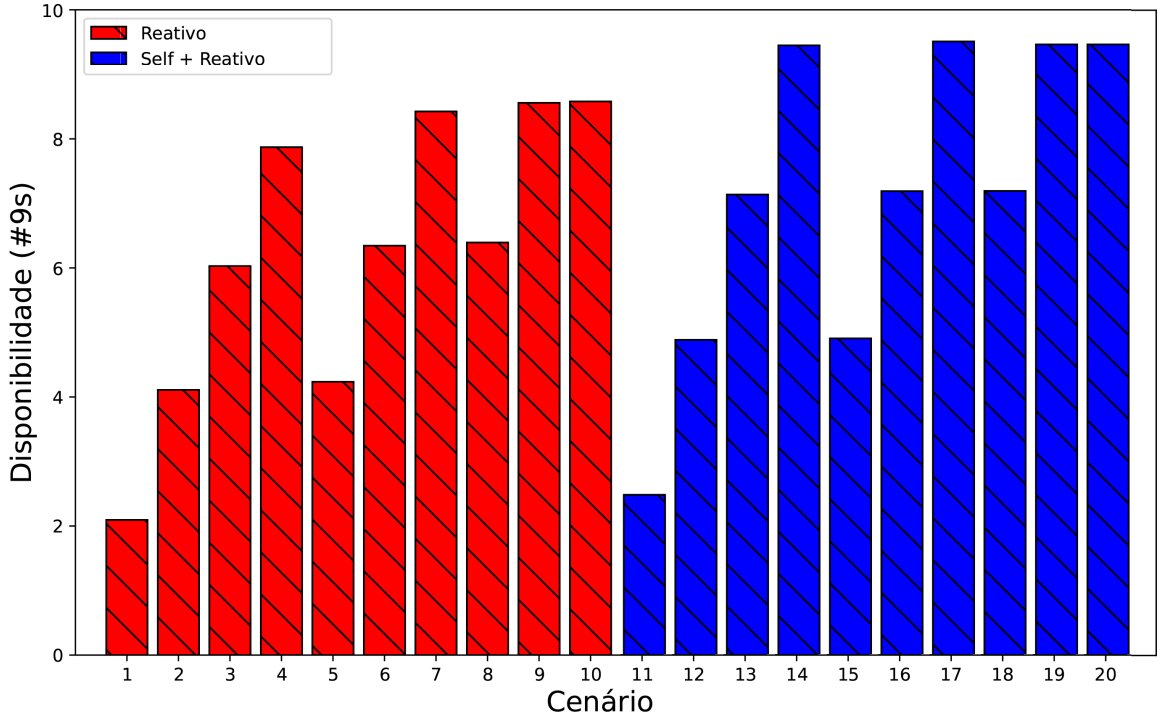
Fonte: Autoria Própria

7.4.4 Resultados Obtidos

A disponibilidade dos cenários apresentados foi avaliada considerando as políticas de endossamento do tipo AND e OR. Deste modo, a avaliação dos modelos com restrição considerou a avaliação de disponibilidade sob duas perspectivas: (i) a probabilidade de que ao menos um servidor está em operação (OR) e; (ii) a probabilidade de que todos os servidores estivessem em operação (AND). A Figura 39 apresenta os resultados gerais de disponibilidade em número de noves para a política de endossamento do tipo OR. Enquanto que a Figura 40 mostra os resultados para os mesmos cenários, mas considerando a adoção de política de endossamento do tipo AND.

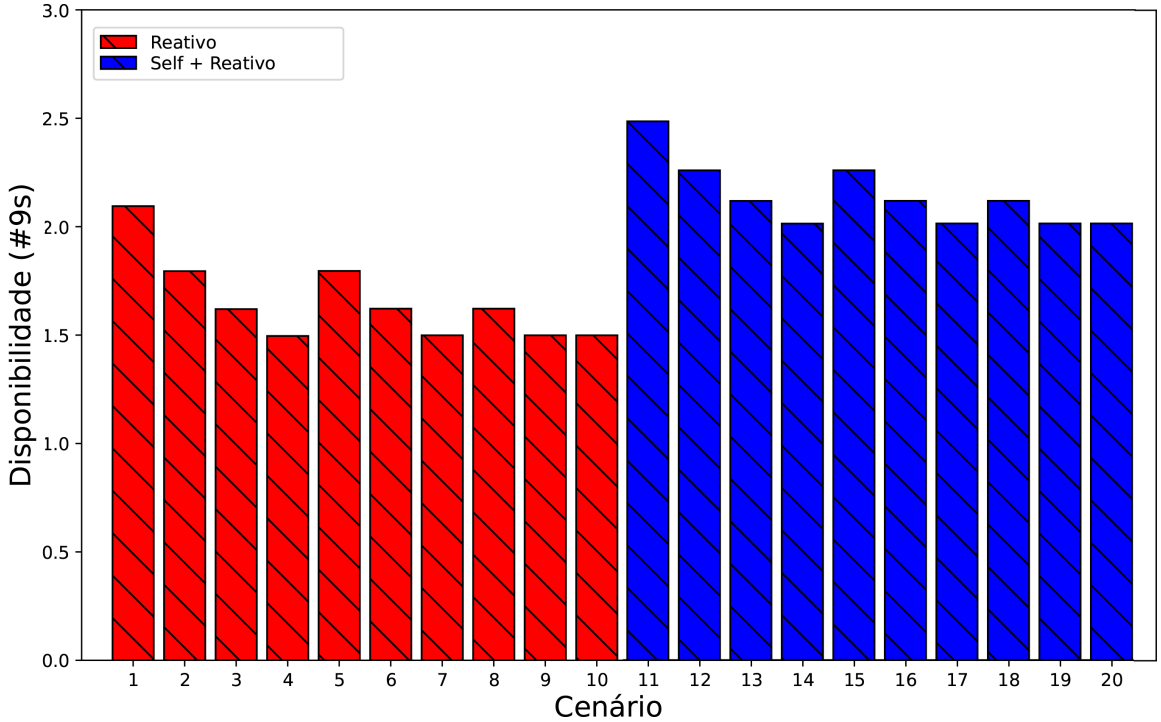
As figuras resumem o melhor e o pior caso em termos de disponibilidade. O modelo com restrições que apresentou os melhores resultados foi o self-healing+reativo. O monitor tenta reiniciar apenas as máquinas cujo serviço não está funcional, o que pode proporcionar um grande ganho em termos de disponibilidade. Todavia, ressaltamos que os valores definidos para sucesso e falha no reparo através de reboot são estimativas (80% e 20%) respectivamente. Além disso, o uso de um monitor sob uma política de endossamento do tipo AND apresentou ganhos significativos em comparação ao de uma política de endossamento do tipo AND, que naturalmente já apresenta índices consideráveis para a métrica

Figura 39 – Política de Endossamento do Tipo OR



Fonte: Autoria Própria

Figura 40 – Política de Endossamento do Tipo AND



Fonte: Autoria Própria

de interesse.

Por fim, a adoção de um modelo com restrições exclusivamente reativo mostrou-se de grande valia. Tal modelo toma como base os valores utilizados para locomoção, tempo médio de reparo, tempo de detecção e número de equipes de manutenção disponíveis. Todavia, é tendência que o seu custo seja mais elevado que o do cenário com mecanismo de autocorreção.

Através da linguagem de scripts presente na ferramenta Mercury fomos capazes de determinar os gargalos do sistema com respeito aos dois modelos com restrição avaliados. Para tanto, a técnica de análise de sensibilidade baseada em diferença percentual foi aplicada. Deste modo, fomos capazes de extrair o impacto de cada componente presente em nossos modelos sobre a disponibilidade geral do sistema. A Tabela 20 apresenta os novos valores de entrada utilizados para a análise de sensibilidade e estendem os valores apresentados na Tabela 18.

Tabela 20 – Parâmetros de Entrada para Análise de Sensibilidade - Modelos de provimento de serviço com restrições

Parâmetro	**Intervalo de Variação (h)**
#Servidores	[1, 4]
#Times	[1, 4]
MTTR	[0.5, 1.5]
MTTF	[154, 462]
M2S	[0.5, 1.5]
MTTR_S	[0.5, 1.5]
DTC	[0.25, 0.75]
REBOOT	[0.04, 0.12]
Transição	**Peso**
SUCCESS	[0.4, 1.0]
FAIL	[0.0, 0.6]

Fonte: Autoria Própria

7.4.4.1 Análise de Sensibilidade - Modelo de provimento de serviço com restrições

Os resultados que correspondem ao ranking da análise de sensibilidade aplicada a este modelo são apresentados no conjunto de Tabelas de número 21. A Tabela 21a mostra que o componente de maior impacto sobre uma política de endossamento do tipo AND é o número de servidores. Já para a política de endossamento do tipo OR (Ver Tabela 21b) o tempo de falha dos servidores é mais importante que o número de servidores, haja vista que apenas um deles é necessário para que a aplicação hospedada realize o endossamento de suas transações.

Por fim, para a política de endossamento KooN fixamos a 2 e a 3 o número de recursos mínimos que deveriam estar operacionais. Deste modo, duas tabelas são apresentadas,

por mais que os índices variem o ranking e a ordem de importância dos componentes é mantido.

Tabela 21 – Ranking de Sensibilidade para Modelo de provimento de serviço com restrições

AND	
Componente	**Índice de Sensibilidade**
#Servidores	0.024071
MTTF	0.010223
MTTR	0.002855
M2S	0.002855
DTC	0.001428
MTTR_S	0
#Times	0

(a)

OR	
Componente	**Índice de Sensibilidade**
MTTF	0.010223
#Servidores	0.008039
MTTR	0.002855
M2S	0.002855
DTC	0.001428
MTTR_S	0
#Times	0

(b)

2oo4 (K=2)	
Componente	**Índice de Sensibilidade**
#Servidores	0,975846
MTTF	0,020425
M2S	0,005730
MTTR	0,005718
DTC	0,002840
#Times	$3{,}75 \times 10^{-5}$
MTTR_S	$1{,}39 \times 10^{-5}$

(c)

3oo4 (K=3)	
Componente	**Índice de Sensibilidade**
#Servidores	0,967772
MTTF	0,030607
M2S	0,008626
MTTR	0,008590
DTC	0,004235
#Times	$1{,}12 \times 10^{-4}$
MTTR_S	$4{,}18 \times 10^{-5}$

(d)

Fonte: Autoria Própria

7.4.4.2 Análise de Sensibilidade - Modelo de provimento de serviço Self-Healing + Reativo com restrições

Os parâmetros de entrada são os mesmos utilizados anteriormente e apresentados na Tabela 20. Os resultados gerais dessa análise de sensibilidade são apresentados nas Tabelas em 22.

Em todos os cenários o número de servidores é o item de maior significância no ranking de sensibilidade. Isso vai na contra-mão do modelo de provimento de serviço com restrições e puramente reativo, onde na política de endossamento do tipo OR o MTTF do servidor sempre teve maior impacto sobre a disponibilidade geral dos sistema. Outro item que passou a ser de grande importância é o tempo de falha do monitor (MTTF_MON). Além disso, temos as probabilidades de falha e reparo através de reboot (FAIL e SUCCESS). O número de times voltou a ter baixo impacto geral, assim como o MTTR_S, ou seja, o

reparo de componente em falha quando a equipe de manutenção já encontra-se no local de provimento do serviço.

Tabela 22 – Ranking de Sensibilidade para Modelo de provimento de serviço com restrições e self-healing

AND	
Componente	**Índice de Sensibilidade**
#Servidores	0,008080
MTTR	0.001456
M2S	0.001456
MTTF_MON	0,001282
FAIL	0,001134
SUCCESS	$8{,}27 \times 10^{-4}$
reboot	$5{,}0 \times 10^{-4}$
DTC	$1{,}70 \times 10^{-4}$
MTTF	0
MTTR_S	0
#Times	0

(a)

OR	
Componente	**Índice de Sensibilidade**
#Servidores	0,004098
MTTR	0,001456
M2S	0,001456
MTTF_MON	0,001282
FAIL	0,001134
SUCCESS	$8{,}27 \times 10^{-4}$
reboot	$5{,}0 \times 10^{-4}$
DTC	$1{,}70 \times 10^{-4}$
MTTF	0
MTTR_S	0
#Times	0

(b)

2oo4 (K=2)	
Componente	**Índice de Sensibilidade**
#Servidores	0,990438
FAIL	0,002641
M2S	0,002460
MTTR	0,002458
SUCCESS	0,001927
MTTF_MON	0,001838
reboot	0,001165
DTC	$3{,}59 \times 10^{-4}$
#Times	$7{,}17 \times 10^{-6}$
MTTF	0
MTTR_S	0

(c)

3oo4 (K=3)	
Componente	**Índice de Sensibilidade**
#Servidores	0,987830
FAIL	0,004194
M2S	0,003414
MTTR	0,003409
SUCCESS	0,003060
MTTF_MON	0,002117
reboot	0,001850
DTC	$7{,}21 \times 10^{-4}$
#Times	$1{,}78 \times 10^{-5}$
MTTF	0
MTTR_S	0

(d)

Fonte: Autoria Própria

7.4.5 Limitações

O uso de redes de Petri requer um conhecimento prévio desse formalismo por parte do avaliador. Deste modo, estamos adicionando uma camada extra de complexidade que até então não era cabível mediante o uso do BPPT para avaliação do provimento de serviços. Porém, observamos que os modelos avaliados nesta subseção são gerais e podem ser readequados a outros tipos de serviços que possam ser providos através da Internet, não limitando-se apenas a blockchain como serviço.

Outra limitação que deve ser destacada está na apresentação de cenários que atendam a uma política de endossamento do tipo KooN. Isso demandaria uma análise combinatória e um conjunto de dezenas de novos cenários que não poderiam ser comparados aos das políticas apresentadas graficamente (OR e AND). Todavia, apresentamos uma análise de sensibilidade para cada modelo de manutenção proposto. Deste modo, estabelecemos quais os gargalos para todas as três políticas de endossamento avaliadas nesta tese, provendo artefatos que podem contribuir para melhorias futuras tanto no processo de manutenção quanto de otimização do ambiente ou infraestrutura a ser implantado.

É importante mencionar, também, que os parâmetros de entrada utilizados podem ser considerados estimativas e dependem de inúmeros fatores, como região em que se encontra o centro de dados, distância entre o local de ocorrência da falha e a equipe de manutenção, custo de troca de componente em falha, tempo de espera por peça de reposição, entre outros. Todavia, tais valores são parametrizáveis em nosso modelo, podendo ser adequados a realidade da empresa que planeja prover algum tipo de serviço.

7.5 ESTUDO DE CASO V - AVALIAÇÃO DE CUSTOS

No quinto estudo de caso avaliamos os custos relacionados a implantação e manutenção de infraestruturas capazes de hospedar as aplicações baseadas em Hyperledger Fabric. Também determinarmos as configurações que apresentam a melhor relação entre custo e benefício com base nas políticas de endossamento anteriormente apresentadas. Todavia, precisamos primeiramente definir o benefício que estamos buscando. Para a presente tese consideramos como benefício o downtime anual de cada infraestrutura computacional avaliada. Quanto menor for o downtime anual de uma infraestrutura, melhor o é, tanto para quem provê serviços quanto para quem consome os recursos ofertados por este ambiente.

A análise de custo leva em conta o custo para aquisição de equipamento, o seu custo energético com base em sua potência em Watts, os custos para manutenção de infraestruturas privadas e o custo do uso de infraestruturas de nuvem pública para hospedagem de serviço baseado em Hyperledger Fabric.

7.5.1 Parâmetros de Entrada

Assim como fizemos nos estudos de caso anteriores ((i) planejamento de disponibilidade através do BPPT e; (ii) avaliação do impacto de modelos de manutenção distintos sobre a disponibilidade do sistema), precisaremos de um conjunto de parâmetros de entrada referentes ao custo de aquisição, implantação e manutenção de infraestruturas computacionais. A Tabela 23 apresenta o conjunto de custos considerados pelo presente estudo de caso. Vale destacar que esses valores serão associados ao BPPT.

O levantamento de custos relacionados a aquisição de servidores para a infraestrutura avaliada decorreu do monitoramento de preços em sites de compras no Brasil e no exte-

Tabela 23 – Parâmetros de Entrada para Análise de Custos

Parâmetros	**Valor**
Custo por Servidor (US$)	1.100,00
Custo por Manutenção (US$)	100,00
Preço do kwh (US$)	0,10
Potência do Servidor (W)	120

Fonte: Autoria Própria

rior pelo período que se estendeu de Janeiro a Março de 2021. O menor valor encontrado para este componente no período observado foi o utilizado em nossos cálculos, aproximadamente R$6.399,00 na cotação do dia 09 de Março de 2021. O servidor escolhido para provimento dos serviços em nosso ambientes foi o Dell PowerEdge T320, com processador Intel Xeon E5-2420, seis núcleos físicos e, 12 threads, quanto a unidade de armazenamento esta conta com 1 TB e, 32 GB de RAM[2].

Realizamos, também, análise da especificação técnica do servidor em busca dos valores de potência (W), mirando o cálculo do custo energético para cada item de acordo com período pré-determinado em nossa ferramenta. Tal avaliação, considera que os componentes estarão operacionais 24 horas por dia os sete dias da semana.

Já os custos de manutenção foram estimados, mas derivam de valores obtidos através de revisão da literatura (CALLOU et al., 2014) e levam em conta a manutenção do tipo reativa. O custo do quilowatt-hora em Recife, Pernambuco, em Março de 2021 era de 54,93 centavos de real [3], que foi convertido para 10 centavos de dólar para nossa avaliação. De posse desses dados, calculamos a quantidade de kWh para cada servidor e operamos sobre o custo de cada kWh para região, desconsiderando encargos tarifários.

7.5.2 Cenários

Utilizamos dos mesmos dez cenários apresentados na Tabela 16, onde variamos as políticas de endossamento (AND, OR e KooN) e o número de servidores. Deste modo, estabelecendo uma quantidade mínima de componentes essenciais ao provimento do serviço que é resultante do número de servidores presentes em nossa infraestrutura.

Estabelecidos os cenários de avaliação, somos capazes de saber quando o serviço está, de fato, operacional, e realizar uma manutenção reativa com quantas equipes de manutenção se queira apenas quando as transações submetidas ao sistema não puderem mais ser realizadas dentro do nosso ambiente.

Como discutido previamente, a inoperabilidade do serviço é mais frequente quando tratamos de uma política de endossamento do tipo AND que demande que todos os

[2] Servidor Specs.: https://goo.gl/XmTbwF
[3] Tarifas Celpe: <https://bit.ly/3v9cvNz>

servidores e contêineres estejam em completa operação, o que difere das políticas OR e KooN, o que nos faz concluir que políticas de endossamento do tipo AND vão demandar um número maior de reparos ao longo do ano.

7.5.3 Resultados Obtidos

Dividimos em três grupos os resultados obtidos em nossa análise de custos. Através do BPPT fomos capazes de determinar os custos de aquisição, manutenção, energético e total em dólares americanos para cada cenário, categorizando assim os custos gerais de implantação e operação de infraestrutura. Também estabelecemos uma relação de custo x benefício sob a perspectiva do downtime anual alcançando em cada um dos cenários avaliados. Por fim, definimos uma relação entre a implantação de serviços considerando infraestruturas de nuvem pública e infraestruturas privadas.

7.5.3.1 Custos Gerais

A Tabela 24 apresenta os custos gerais relacionados a implantação de cada cenário considerando o primeiro ano da infraestrutura, deste modo incluímos o custo de aquisição dos servidores como um dos fatores determinantes a precificação total. Note mais uma vez que o custo para outros componentes não foi acrescido em nenhuma das colunas desta tabela.

Tabela 24 – Custos Gerais - 1º Ano

Cenário	**Aquisição (US$)**	**Manutenção (US$)**	**Energético (US$)**	**Custo Total (US$)**
1	1.100,00	2.839,70	105,12	4.044,81
2	2.200,00	5.679,39	210,24	8.089,63
3	3.300,00	8.519,09	315,36	12.134,45
4	4.400,00	11.357,79	420,48	16.179,27
5	2.200,00	2.839,70	210,24	5.249,73
6	3.300,00	2.839,70	315,36	6.455,05
7	4.400,00	2.839,70	315,00	7.660,17
8	3.300,00	5.679,39	315,36	9.294,75
9	4.400,00	5.679,39	420,48	10.499,87
10	4.400,00	8.519,09	420,48	13.339,57

Fonte: Autoria Própria

O custo de manutenção é o mais elevado dentre os custos avaliados, para estabelecermos o custo total de manutenção para o período de um ano, contabilizamos o total de falhas ocorridas naquela infraestrutura no decorrer do mesmo período. Para tanto, dividimos o número de horas em um ano não bissexto (8760h) pelo tempo médio entre falhas (MTBF), onde $MTBF = MTTF + MTTR$. Para cada nó, com base nos valores de entrada utilizados, alcançamos aproximadamente 28,39 falhas por ano. O valor encontrado

é fixo para infraestruturas sob a política de endossamento do tipo OR; é multiplicado pelo número total de servidores em uma política de endossamento do tipo AND e; multiplicado pelo número mínimo de recursos (K) que devem estar em operação sob uma política KooN.

Outro custo essencial ao provimento do serviço é o custo energético, ressaltamos que o custo energético avaliado engloba, também, o custo com refrigeração, pois foi considerado o fato de que para cada dólar gasto com operação de servidores foi gasto um outro dólar para refrigerá-lo (CHOPRA, 2017). Deste modo, quanto maior o número de máquinas maior o custo energético.

Por fim, o custo total, que corresponde a soma dos outros três custos considerados. A arquitetura do cenário 1, baseline, foi a que apresentou o menor custo total e é a única com um único servidor, não fugindo ao esperado. Já entre as infraestruturas com dois servidores o cenário 5 sob política de endossamento do tipo OR apresentou melhor resultado que o cenário 2 em uma política de endossamento do tipo AND, isso se repete para os demais cenários com 3 e 4 servidores. O cenários KooN (8, 9 e 10) se mostraram, uma vez mais, alternativas para o analista que podem ir além do conteúdo avaliado por esta tese e acrescentar parâmetros relacionados a segurança das aplicações em futuros trabalhos.

7.5.3.2 Custo x Benefício

Sabendo dos custos associados a cada infraestrutura miramos a identificação da melhor relação custo x benefício. Como já mencionado, o benefício que consideraremos é o *downtime anual*. Mas como custo e downtime são grandezas distintas precisamos estabelecer um processo de normalização para colocá-los em um mesmo intervalo, algo entre 0 e 1. O BPPT provê uma visão geral da relação custo x benefício que tentamos destrinchar melhor para o leitor com o foco no custo total. A Expressão 7.4 apresenta o processo de normalização conduzido pela presente tese.

$$\text{NormalizarXY} = \frac{\text{NumX - MinNumX}}{\text{MaxNumX - MinNumX}} \tag{7.4}$$

onde X é o custo total ou o downtime anual, e Y o respectivo cenário ou infraestrutura avaliada, deste modo, MinNumX é o menor valor para downtime ou custo para a infraestrutura e MaxNumX sua contraparte, ou seja, o valor máximo para o parâmetro de interesse.

Já temos posse do custo e downtime anual normalizados, mas devemos relacioná-los de alguma forma, para isso usaremos da distância euclidiana. A infraestrutura com menor distância para a origem em um plano cartesiano tende a ser a que apresenta a melhor relação custo x benefício. O cálculo da distância euclidiana se dá através da raiz quadra dos quadrados dos termos somados, deste modo, $\text{DistânciaZ} = \sqrt{\text{NMCost}^2 + \text{NMDowntime}^2}$,

onde Z representa a infraestrutura e NM as métricas em avaliação, ou seja, custo normalizado e downtime normalizado. A Tabela 25 apresenta um ranking das infraestruturas com base na distância euclidiana, além dos seus respectivos custos e downtime normalizados.

Tabela 25 – Ranking de Infraestruturas

Cenário	**Colocação**	**Downtime Normalizado**	**Custo Normalizado**	**Distância Euclidiana**
1	3º	0,25	0	0,25
2	7º	0,50	0,33	0,60
3	9º	0,75	0,67	1,00
4	10º	1	1	1,41
5	1º	0,00016	0,1	0,09
6	2º	0,00000001	0,2	0,19
7	4º	0	0,3	0,29
8	5º	0,0004	0,43	0,43
9	6º	0,00000004	0,53	0,53
10	8º	0,0009	0,77	0,76

Fonte: Autoria Própria

Nosso ranking aponta o cenário 5 como o cenário de melhor custo x benefício, a política de endossamento nele utilizada é a do tipo OR e o número total de servidores na infraestrutura representa é 2 (dois). Esta política de endossamento também se destaca ao ocupar o segundo e o quarto lugar em nosso ranking, para cenários com três ou quatro servidores, respectivamente. Nota-se, porém, que o aumento do custo não justifica a redução do downtime anual associado, algo que alcança uma marca de segundos por ano quando temos quatro servidores a nossa disposição.

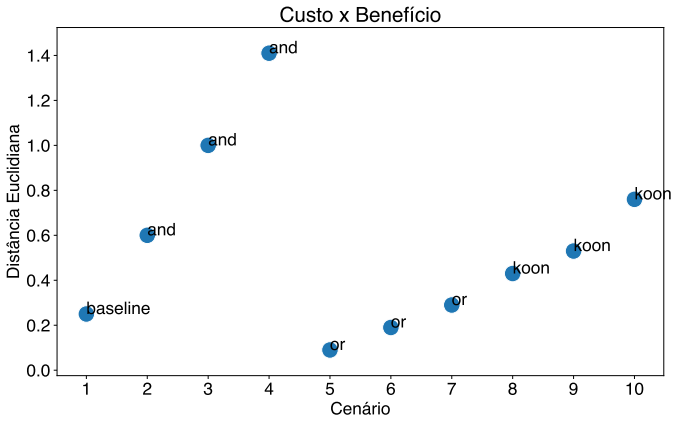
O terceiro lugar vai para o cenário baseline, que apresenta o menor custo dentre todos os cenários avaliados, afinal de contas possui um único servidor em operação. Já os piores resultados são alcançados pela política AND e agora os quantificamos através do nosso ranking, como mencionado anteriormente a opção a essa política é a do tipo KooN que representa um meio termo. A Figura 41 apresenta os mesmos resultados gerais previamente apresentados, porém de forma gráfica.

7.5.3.3 Infraestrutura Privada x Pública

Para comparativo entre infraestrutura privada e pública estimamos o tempo de vida útil para um servidor, a partir do momento de sua compra. O valor estipulado foi de três anos. Deste modo, seguimos os mesmos cenários anteriormente listados (Ver Tabela 16) e variamos o intervalo entre 1 e 3 anos.

Para cada servidor físico em infraestrutura privada comparamos com a utilização de uma instância virtual nas plataformas Amazon AWS, Google Cloud e Microsoft Azure. A Tabela 26 apresenta os custos relacionados as instâncias dessas plataformas.

Figura 41 – Custo x Benefício



Fonte: Autoria Própria

Tabela 26 – Despesas Anuais com Instâncias de Nuvem Pública

Platforma	**Tipos de Instância**	**Preço On-demand/ano (US$)**	**Disponibilidade (%)**
Amazon	n1-standard-1	291,24	99,9
Google	m3.medium	420,48	99,9
Microsoft	D1s	403,56	99,9

Fonte: Autoria Própria

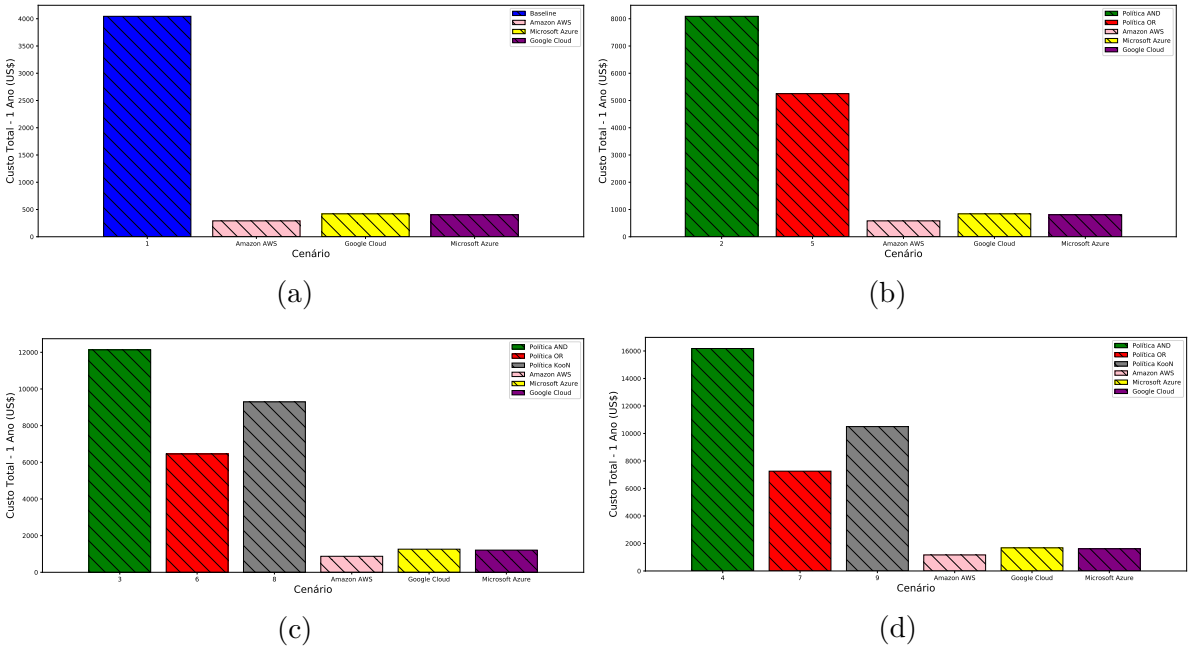
Para esta avaliação, mais uma vez, fizemos uso do BPPT, as Figuras 42, 43 e 44 apresentam os resultados de custos gerais obtidos comparando-se os dez cenários privados com os ambientes de infraestrutura pública equivalentes, ao longo de três anos.

A Figura 42a apresenta o comparativo do custo do primeiro ano entre o cenário baseline e as provedoras públicas, já a Figura 42b mostra um comparativo dos cenários 2 e 5, ambos com dois servidores, com suas versões equivalentes em ambientes de nuvem pública. A Figura 42c compara os cenários que possuem 3 servidores e a Figura 42d mostra os custos para quatro servidores. Para este e todos os cenários subsequentes a política de endossamento do tipo AND apresenta o maior custo e tem crescimento exponencial tendo como única aproximação cenários sob a política KooN.

Já para o segundo ano as Figuras em 43 começam a mostrar que para os tipos de instância escolhidas o custo nunca será maior que o de uma infraestrutura privada, mesmo com redução considerável pela não necessidade de aquisição de equipamentos o custo de manutenção é muito elevado e prejudicial a esse modelo de serviço.

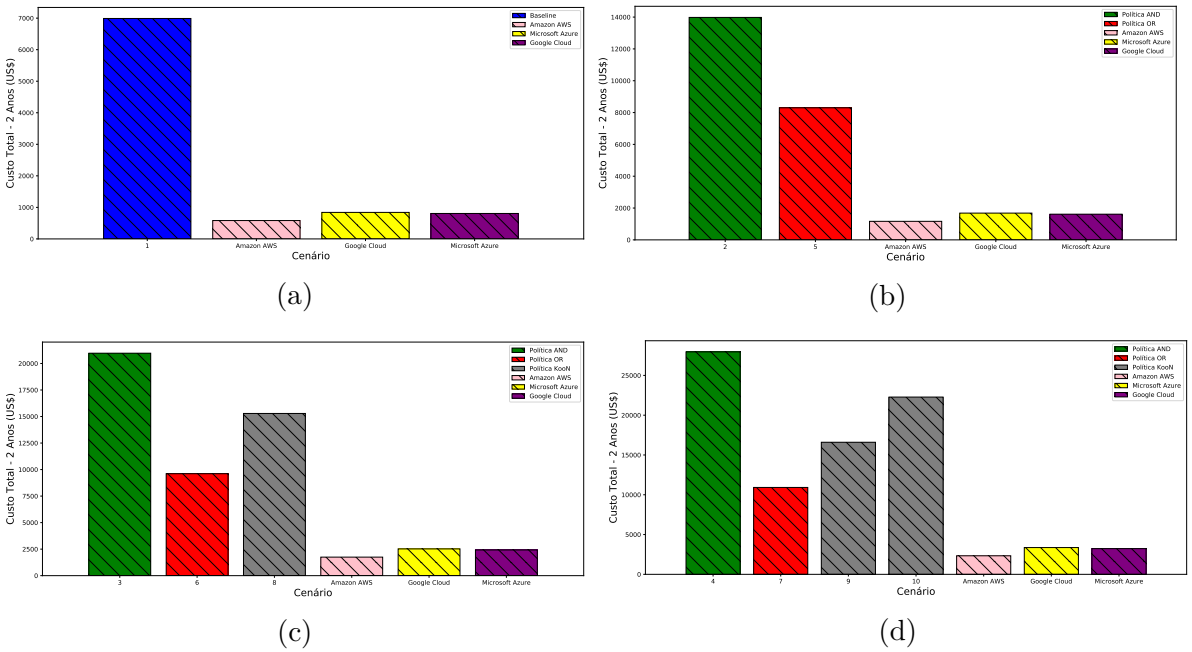
Por fim, para o terceiro ano apresentado pelas Figuras em 44 vemos que a relação se mantém e os gráficos passam a soar como repetitivos. Mas é sempre importante ressaltar a limitação das instâncias consideradas para ambientes públicos. Essas limitações vão desde o desempenho do processador propriamente dito a latência de conexão e outras métricas associadas. Outro importante fator que devemos mencionar é que em cenários sob a política de endossamento do tipo OR temos excelentes alternativas de preço ao uso

Figura 42 – Custos Gerais para 1 Ano



Fonte: Autoria Própria

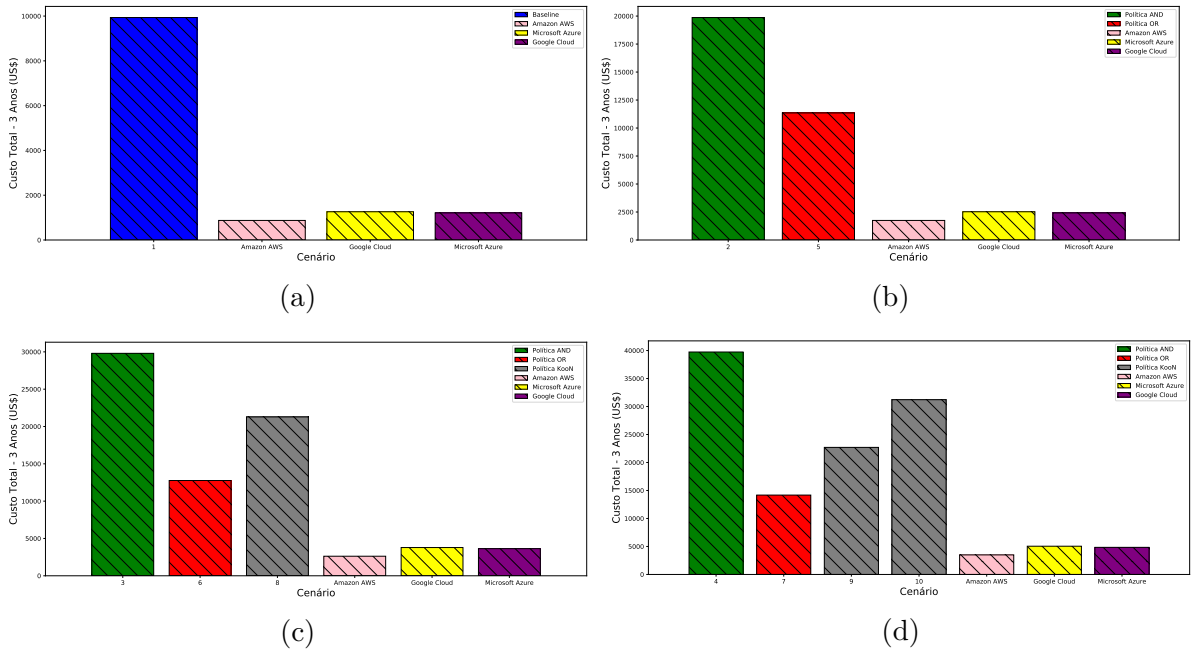
Figura 43 – Custos Gerais para 2 Anos



Fonte: Autoria Própria

de infraestruturas privadas e se desconsiderarmos o fator manutenção externa, ou seja, o provedor de serviço já conta com uma equipe de manutenção própria não precisando de expansão do seu quadro de funcionários, o custo será ainda menor.

Figura 44 – Custos Gerais para 3 Anos



Fonte: Autoria Própria

7.5.4 Limitações

As principais limitações do presente estudo de caso giram, mais uma vez, em torno do processo de manutenção das infraestruturas privadas. Os custos a elas associados são variantes e dependem do tipo de falha ocorrida, bem como do contrato estabelecido entre provedor de ambiente e do responsável pela realização serviço de manutenção, este que pode ser interno (funcionário ou equipe própria) ou externo a organização (terceirizado).

Outro importante fator desconsiderado por nosso estudo de caso gira em torno das taxas e encargos de operação do serviço, além do quilowatt-hora diversas taxas estão associadas a contratação do serviço de energia elétrica.

Outra limitação que deve ser destacada está na apresentação de cenários que atendam a uma política de endossamento do tipo KooN, o que demandaria uma análise combinatória e um conjunto de dezenas de novos cenários que não poderiam ser comparados aos das políticas apresentadas graficamente (OR e AND). Todavia, apresentamos uma análise de sensibilidade para cada modelo de manutenção proposto, deste modo, estabelecemos quais os gargalos para todas as três políticas de endossamento avaliadas nesta tese, provendo artefatos que podem contribuir para melhorias futuras tanto no processo de manutenção quanto de otimização do ambiente ou infraestrutura a ser implantado.

É importante mencionar, também, que os parâmetros de entrada utilizados podem ser considerados estimativas e dependem de inúmeros fatores, como região em que se encontra o centro de dados, distância entre o local de ocorrência da falha e a equipe de manutenção, custo de troca de componente em falha, tempo de espera por peça de reposição, entre outros. Todavia, tais valores são parametrizáveis em nosso modelo, podendo ser adequados a realidade da empresa que planeja prover algum tipo de serviço.

Por fim, os custos associados as instâncias no cenário avaliado são bem específicos, ou seja, apenas três tipos de instância foram avaliados, mas existem dezenas para cada provedor. O que limita consideravelmente o impacto desta avaliação, todavia ela é apenas demonstrativa, caso o analista tenha interesse em expandi-la poderá selecionar outras opções de instância e seus custos nos respectivos sites das provedoras.

7.6 CONSIDERAÇÕES FINAIS

Esta seção apresentou uma série de estudos de caso, e através deles fomos capazes de determinar a viabilidade do que propomos nesta tese. A começar pelo Blockchain Provisioning Planning Tool (BPPT) e o planejamento de serviços com base na disponibilidade e custo de implantação de infraestruturas e serviços baseados em Hyperledger fabric, depois avaliamos uma série de outros modelos que visam determinar o impacto da manutenção sobre a disponibilidade do sistema.

8 CONCLUSÕES E TRABALHOS FUTUROS

> *No matter the era, whichever the society, people always gaze at the stars when young. Then they stretch out their hands and try to catch them. And then one day, they realize their arms are not long enough to catch the stars, and that's when they grow up.*
>
> — Reinhard Von Lohengramm, *LOGH*, 1982

Blockchain é uma realidade em ascensão, sua presença vem se tornando ubíqua, porém, sua caracterização é desconhecida por muitos que a utilizam. Seja na academia ou no mercado seu uso vem sendo investigado com base unicamente em premissas de desempenho e segurança, dois dos principais fatores que a diferenciam de bancos de dados distribuídos tradicionais.

A presente tese de doutorado estende a avaliação de desempenho e apresenta, também, a avaliação de dois dos principais atributos de dependabilidade, além de uma análise de custos relacionados à implantação e manutenção de infraestruturas capazes de hospedar aplicações e serviços baseados em blockchain.

A avaliação de desempenho apontou limitações acerca do servidor utilizado para o provimento de uma aplicação baseada no Hyperledger Fabric. Essa avaliação contou com a utilização do Hyperledger Caliper, um Benchmark capaz de aplicar uma carga de trabalho sob um ambiente de testes. Além disso, utilizamos scripts bash para monitoramento dos recursos de hardware durante a aplicação da carga de trabalho. Através dos resultados obtidos pelo monitoramento, estabelecemos um modelo de performabilidade capaz de determinar o impacto da RAM e do disco sobre a disponibilidade geral do sistema.

Além do modelo de performabilidade, esta tese propõe uma série de outros modelos formais, que auxiliam o processo de tomada de decisão. A avaliação dos atributos de dependabilidade utilizou modelagem hierárquica. Esta abordagem consistiu no uso de ferramentas e formalismos matemáticos que possibilitam a conversão de um ambiente real em modelos comportamentais.

O modelo para avaliação de confiabilidade apresentado mostra a existência de uma tendência de falha para os primeiros trinta dias de sua operação. Já para estabelecimento de uma relação entre a disponibilidade e o custo de implantação de infraestruturas, propomos um outro conjunto de modelos, além de uma ferramenta web de alto nível.

É importante salientar que o processo de modelagem iniciou-se com Diagramas de Bloco Confiabilidade (RBDs) e Cadeias de Markov de Tempo Contínuo (CTMCs) para estabelecimento de um modelo genérico de disponibilidade. Tal modelo foi então abstraído por uma representação matemática que foi a base para a construção do Blockchain Provisioning Planning Tool (BPPT). O BPPT é uma ferramenta web capaz de auxiliar a alta

gerência no processo de tomada de decisão quanto à implantação e provimento de serviços baseados na plataforma Hyperledger Fabric.

Já para avaliação de performabilidade, confiabilidade e outros cenários de maior restrição, optamos por redes de Petri estocásticas, que são modelos isomórficos das Cadeias de Markov. Essa escolha teve como base o grande número de estados gerados e características que dificilmente poderiam ser replicadas em uma CTMC, aumentando a complexidade da avaliação.

Tanto através dos modelos como da ferramenta proposta, estabelecemos uma série de cenários para demonstração da viabilidade da presente tese, considerando a disponibilidade geral da infraestrutura e o seu respectivo downtime anual com base nas políticas de endossamento avaliadas.

Quanto à relação entre o downtime anual, custos e políticas de endossamento avaliadas, apontamos a política do tipo AND como a principal vilã, devido a necessidade de contínua operação de todos os servidores que compõem a infraestrutura. Já a política de endossamento do tipo OR apresenta a melhor relação geral entre downtime anual e custos, com a política KooN como um meio termo entre ambas. As próximas subseções descrevem as principais contribuições desta tese, bem como um conjunto de possíveis trabalhos futuros.

8.1 CONTRIBUIÇÕES

Através das atividades desenvolvidas no âmbito da presente tese identificamos um conjunto de contribuições, dentre as quais:

- Elaboração e avaliação de modelos de dependabilidade para o provimento de serviços baseados na plataforma de blockchain Hyperledger Fabric;
- Metodologias para avaliação de disponibilidade, confiabilidade, desempenho, performabilidade e custo × benefício;
- Criação de uma ferramenta que faz uso de um dos modelos de disponibilidade propostos e dos custos associados à implantação e manutenção para auxiliar o processo de planejamento de infraestruturas computacionais capazes de hospedar aplicações e serviços baseados em blockchain;
- Avaliação de Desempenho e elaboração de um modelo de performabilidade capaz de enumerar o impacto do consumo de recursos sobre a disponibilidade do sistema.

8.2 TRABALHOS FUTUROS

Embora a presente tese tenha alcançado e proporcionado uma série de avanços relacionados ao estudo de desempenho, dependabilidade e custos de implantação e manutenção

de serviços baseados em blockchain, há ainda uma série de oportunidades a serem exploradas por nós ou por outros pesquisadores interessados neste campo de pesquisa.

Primeiramente, esperamos conseguir adequar os modelos genéricos de confiabilidade e os modelos de provimento de serviço com restrições ao BPPT, não limitando-o apenas a avaliação de disponibilidade e as manutenções do tipo reativa com proporção 1:1, garantindo assim, uma maior aproximação a manutenção e operação de serviços e infraestruturas reais.

Outra possibilidade está no cálculo de outros atributos e métricas inerentes a dependabilidade de sistemas computacionais, como a segurança, que apresenta papel importante sobre as políticas de endossamento escolhidas.

É importante salientar que o BPPT é uma ferramenta de código aberto e que será mantida pela comunidade. Ela ainda possui muitas limitações em seu entorno, mas futuros avanços poderão vir de toda parte, entre eles está a inclusão de um monitor e gerador de carga de trabalho que faça uso da API do Hyperledger Caliper. Deste modo, poderemos prover além da dependabilidade, mas também estimativas correlatas ao desempenho da plataforma Hyperledger Fabric diretamente no BPPT. Em termos de desempenho, podemos dar ênfase as métricas de latência e vazão do respectivo sistema, ambas avaliadas no âmbito desta tese, é fato que a primeira poderá ter uma grande influência da localização do centro de dados, haja vista que na presente tese avaliamos um ambiente localmente, o que torna factível a inclusão de um simulador geográfico para novos estudos e geração de cenários.

REFERÊNCIAS

AGERWALA, T. Special feature: Putting petri nets to work. *Computer*, IEEE Computer Society, Los Alamitos, CA, USA, v. 12, n. 12, p. 85–94, 1979. ISSN 0018-9162.

ALASLANI, M.; NAWAB, F.; SHIHADA, B. Blockchain in iot systems: End-to-end delay evaluation. *IEEE Internet of Things Journal*, p. 1–1, 2019. ISSN 2327-4662.

ALTARAWNEH, A.; SUN, F.; BROOKS, R. R.; HAMBOLU, O.; YU, L.; SKJELLUM, A. Availability analysis of a permissioned blockchain with a lightweight consensus protocol. *Computers & Security*, Elsevier, v. 102, p. 102098, 2021.

ARAUJO, J.; MATOS, R.; MACIEL, P.; MATIAS, R.; BEICKER, I. Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure. In: *Proceedings of the middleware 2011 industry track workshop*. [S.l.: s.n.], 2011. p. 1–7.

ARUNDEL, J.; DOMINGUS, J. *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud*. [S.l.]: O'Reilly Media, 2019.

AVIŽIENIS, A.; LAPRIE, J.; RANDELL, B.; SCIENCE, U. of Newcastle upon T. C. *Fundamental Concepts of Dependability*. University of Newcastle upon Tyne, Computing Science, 2001. (Technical report series). Disponível em: <https://books.google.com.br/books?id=cDkmGwAACAAJ>.

AVIZIENIS, A.; LAPRIE, J.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, v. 1, p. 11–33, 2004.

BALBO, G. Lectures on formal methods and performanceanalysis: First eef/euro summer school on trends in computer science bergen dal, the netherlands, july 3–7, 2000 revised lectures. In: _____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. cap. Introduction to Stochastic Petri Nets, p. 84–155. ISBN 978-3-540-44667-5. Disponível em: <http://dx.doi.org/10.1007/3-540-44667-2_3>.

BARROSO, L. A.; HÖLZLE, U.; RANGANATHAN, P. The datacenter as a computer: Designing warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, Morgan & Claypool Publishers, v. 13, n. 3, p. i–189, 2018.

BOLCH, G.; GREINER, S.; MEER, H. de; TRIVEDI, K. Modeling and performance evaluation with computer science application. In: *Queuing Networks and Markov Chains*. [S.l.: s.n.], 2006.

BRINCKMAN, A.; LUC, D.; NABRZYSKI, J.; NEIDIG, G. L.; NEIDIG, J.; PUCKETT, T. A.; RADHA, S. K.; TAYLOR, I. J. A comparative evaluation of blockchain systems for application sharing using containers. In: *2017 IEEE 13th International Conference on e-Science (e-Science)*. [S.l.: s.n.], 2017. p. 490–497.

BUTERIN, V. Ethereum: Platform review. *Opportunities and Challenges for Private and Consortium Blockchains*, 2016.

CALLOU, G.; FERREIRA, J.; MACIEL, P.; TUTSCH, D.; SOUZA, R. An Integrated Modeling Approach to Evaluate and Optimize Data Center Sustainability, Dependability and Cost. *Energies*, v. 7, n. 1, p. 238–277, jan. 2014. ISSN 1996-1073. Disponível em: <http://www.mdpi.com/1996-1073/7/1/238>.

CASSANDRAS, C.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. Kluwer Academic, 1999. (Discrete event dynamic systems). ISBN 9780792386094. Disponível em: <https://books.google.com.br/books?id=IuGo4aRyb00C>.

CHOPRA, R. *Cloud Computing: An Introduction*. [S.l.]: Stylus Publishing, LLC, 2017.

CROMAN, K.; DECKER, C.; EYAL, I.; GENCER, A. E.; JUELS, A.; KOSBA, A.; MILLER, A.; SAXENA, P.; SHI, E.; SIRER, E. G. et al. On scaling decentralized blockchains. In: SPRINGER. *International Conference on Financial Cryptography and Data Security*. [S.l.], 2016. p. 106–125.

DANTAS, J. *Modelos para Analise de Dependabilidade de Arquiteturas de Computação em Nuvem*. Dissertação (Mestrado) — Centro de Informática - Universidade Federal de Pernambuco (Recife, Brasil), 2013.

DANTAS, J. *Planejamento de infraestrutura de nuvens computacionais para serviço de VoD streaming considerando desempenho, disponibilidade e custo*. Tese (Doutorado) — Federal University of Pernambuco, Recife, Brazil, 2018. AAI8201138.

EJLALI, A.; MIREMADI, S. G.; ZARANDI, H.; ASADI, G.; SARMADI, S. B. A hybrid fault injection approach based on simulation and emulation co-operation. In: IEEE COMPUTER SOCIETY. *2003 International Conference on Dependable Systems and Networks, 2003. Proceedings.* [S.l.], 2003. p. 479–479.

EKBLAW, A.; AZARIA, A.; HALAMKA, J. D.; LIPPMAN, A. A case study for blockchain in healthcare:"medrec" prototype for electronic health records and medical research data. In: *Proceedings of IEEE open & big data conference*. [S.l.: s.n.], 2016. v. 13, p. 13.

FRANK, P. M. *Introduction to Sensitivity Analysis*. [S.l.]: Academic, 1978.

GARG, S.; A, P.; M, T.; TRIVEDI, K. S. Analysis of software rejuvenation using markov regenerative stochastic petri net. In: *Proc. In: Sixth International Symposium on Software Reliability Engineering, (ISSRE'95)*. Paderborn: [s.n.], 1995. p. 180–187. ISBN 0-8186-8991-9.

GERMAN, R. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. New York, NY, USA: John Wiley & Sons, Inc., 2000. ISBN 0471492582.

GROUP, H. A. W. et al. Hyperledger architecture. *Hyperledger. org*, v. 1, p. 15, 2017.

GUPTA, M. *Blockchain for DUMMIES*. [S.l.]: John Wiley & Sons, Inc., 2017. ISBN 978-1-119-37123-6.

HERZOG, U. Lectures on formal methods and performance analysis. In: BRINKSMA, E.; HERMANNS, H.; KATOEN, J.-P. (Ed.). New York, NY, USA: Springer-Verlag New York, Inc., 2002. cap. Formal Methods for Performance Evaluation, p. 1–37. ISBN 3-540-42479-2. Disponível em: <http://dl.acm.org/citation.cfm?id=567305.567306>.

HYPERLEDGER. *Hyperledger Blockchain Performance Metrics White Paper*. [S.l.], 2018.

HYPERLEDGER. *An Introduction to Hyperledger*. [S.l.], 2018.

HYPERLEDGER. *Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus*. [S.l.], 2018.

HYPERLEDGER. *Smart Contracts*. [S.l.], 2018.

IEEE. Ieee standard glossary of software engineering terminology. In: *IEEE Std.* [S.l.]: IEEE Computer Society, 1990.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York: Wiley Computer Publishing, John Wiley & Sons, Inc., 1991. ISBN 0471503363.

JIANG, T.; FANG, H.; WANG, H. Blockchain-based internet of vehicles: distributed network architecture and performance analysis. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 4640–4649, 2018.

KANCHARLA, A.; KE, Z.; PARK, N.; KIM, H. Hybrid chain and dependability. In: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. [S.l.: s.n.], 2020. p. 204–209.

KANCHARLA, A.; SEOL, J.; PARK, N.; PARK, I.; PARK, N. Dependable industrial crypto computing. In: *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. [S.l.: s.n.], 2019. p. 1225–1232.

KANCHARLA, A.; SEOL, J.; PARK, N.; KIM, H. Slim chain and dependability. In: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*. [S.l.: s.n.], 2020. p. 180–185.

KASAHARA, S. Performance modeling of bitcoin blockchain: Mining mechanism and transaction-confirmation process. *IEICE Transactions on Communications*, The Institute of Electronics, Information and Communication Engineers, 2021.

KEESEE, W. *A method of determining a confidence interval for availability*. [S.l.], 1965.

KHAN, K. M.; ARSHAD, J.; KHAN, M. M. Investigating performance constraints for blockchain based secure e-voting system. *Future Generation Computer Systems*, Elsevier, v. 105, p. 13–26, 2020.

KOLMOGOROFF, A. Über die analytischen methoden in der wahrscheinlichkeitsrechnung. *Mathematische Annalen*, Springer, v. 104, n. 1, p. 415–458, 1931.

KOUSHIK, A.; JAIN, B.; MENON, N.; LOHIA, D.; CHAUDHARI, S. Performance analysis of blockchain-based medical records management system. In: IEEE. *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. [S.l.], 2019. p. 985–989.

KRISHNASWAMY, D. Performance considerations for edge blockchain systems in emerging 5g data networks. In: *Proceedings of the 21st International Conference on Distributed Computing and Networking*. [S.l.: s.n.], 2020. p. 1–6.

KUO, W.; ZUO, M. *Optimal Reliability Modeling: Principles and Applications*. Wiley, 2003. ISBN 9780471275459. Disponível em: <https://books.google.com.br/books?id=vdZ4Bm-LnHMC>.

KUROSE, J. F.; ROSS, K. W.; ZUCCHI, W. L. *Redes de Computadores ea Internet: uma abordagem top-down*. [S.l.]: Pearson Addison Wesley, 2007.

LI, Q.-L.; MA, J.-Y.; CHANG, Y.-X. Blockchain queueing theory. *arXiv preprint arXiv:1808.01795*, 2018.

LILJA, D. J. *Measuring Computer Performance: A Practitioner's Guide*. New York, NY, USA: Cambridge University Press, 2000. ISBN 0-521-64105-5.

LIU, Y.; ZHENG, K.; CRAIG, P.; LI, Y.; LUO, Y.; HUANG, X. Evaluating the reliability of blockchain based internet of things applications. In: IEEE. *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. [S.l.], 2018. p. 230–231.

MACIEL, P.; LINS, R.; CUNHA, P. *Uma Introducao as Redes de Petri e Aplicacoes*. [S.l.]: Sociedade Brasileira de Computacao, 1996. 213 p.

MACIEL, P.; MATOS, R.; SILVA, B.; FIGUEIREDO, J.; OLIVEIRA, D.; Fé, I.; MACIEL, R.; DANTAS, J. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*. [S.l.: s.n.], 2017. p. 50–57. ISSN 2473-3105.

MACIEL, P.; TRIVEDI, K.; MATIAS, R.; KIM, D. Dependability modeling. In: *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*. [S.l.: s.n.], 2011.

MARSAN, M. A.; BALBO, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. *Modelling with Generalized Stochastic Petri Nets*. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc., 1994. ISBN 0471930598.

MATOS, R.; ARAUJO, J.; OLIVEIRA, D.; MACIEL, P.; TRIVEDI, K. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory*, v. 50, p. 151 – 164, 2015. ISSN 1569-190X. Special Issue on Resource Management in Mobile Clouds. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1569190X14000616>.

MEGLIORINI, E. *Custos*. [S.l.]: Makron Books do Brasil, 2001.

MELO, C.; DANTAS, J.; ARAUJO, J.; MACIEL, P. Availability models for synchronization server infrastructure. In: *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'16)*. Budapest, Hungary: [s.n.], 2016.

MELO, C.; DANTAS, J.; PEREIRA, P.; MACIEL, P. Distributed application provisioning over ethereum-based private and permissioned blockchain: availability modeling, capacity, and costs planning. *The Journal of Supercomputing*, Springer, p. 1–27, 2021.

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

MISIC, J.; MISIC, V. B.; CHANG, X.; MOTLAGH, S. G.; ALI, Z. M. Modeling of bitcoin's blockchain delivery network. *IEEE Transactions on Network Science and Engineering*, IEEE, 2019.

MOLLOY, M. K. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. Tese (Doutorado) — University of California, Los Angeles, USA, 1981. AAI8201138.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. [S.l.]: John Wiley & Sons, 2006. ISBN 0470088109.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, Apr 1989. ISSN 0018-9219.

NAKAMOTO, S. et al. Bitcoin: A peer-to-peer electronic cash system. Working Paper, 2008.

ÖHMANN, D.; SIMSEK, M.; FETTWEIS, G. P. Achieving high availability in wireless networks by an optimal number of rayleigh-fading links. In: IEEE. *2014 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2014. p. 1402–1407.

ONIK, M. M. H.; MIRAZ, M. H. Performance analytical comparison of blockchain-as-a-service (baas) platforms. In: SPRINGER. *International Conference for Emerging Technologies in Computing*. [S.l.], 2019. p. 3–18.

PATEL, P.; RANABAHU, A. H.; SHETH, A. P. Service level agreement in cloud computing. 2009.

PEREIRA, P.; ARAUJO, J.; MELO, C.; SANTOS, V.; MACIEL, P. Analytical models for availability evaluation of edge and fog computing nodes. *The Journal of Supercomputing*, Springer, p. 1–29, 2021.

RODRIGUES, B.; BOCEK, T.; LAREIDA, A.; HAUSHEER, D.; RAFATI, S.; STILLER, B. A blockchain-based architecture for collaborative ddos mitigation with smart contracts. In: SPRINGER, CHAM. *IFIP International Conference on Autonomous Infrastructure, Management and Security*. [S.l.], 2017. p. 16–29.

ROEHRS, A.; COSTA, C. A. da; RIGHI, R. da R.; SILVA, V. F. da; GOLDIM, J. R.; SCHMIDT, D. C. Analyzing the performance of a blockchain-based personal health record implementation. *Journal of biomedical informatics*, Elsevier, v. 92, p. 103140, 2019.

SABERI, S.; KOUHIZADEH, M.; SARKIS, J.; SHEN, L. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, Taylor & Francis, v. 57, n. 7, p. 2117–2135, 2019.

SAHA, S.; SARKAR, J.; DWIVEDI, A.; DWIVEDI, N.; NARASIMHAMURTHY, A. M.; ROY, R. A novel revenue optimization model to address the operation and maintenance cost of a data center. *Journal of Cloud Computing*, SpringerOpen, v. 5, n. 1, p. 1–23, 2016.

SEBASTIO, S.; GHOSH, R.; MUKHERJEE, T. An availability analysis approach for deployment configurations of containers. *IEEE Transactions on Services Computing*, IEEE, 2018.

SILVA, B.; MATOS, R.; CALLOU, G.; FIGUEIREDO, J.; OLIVEIRA, D.; FERREIRA, J.; DANTAS, J.; JUNIOR, A. L.; ALVES, V.; MACIEL, P. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: *Proc. of Industrial Track at 45th Dependable Systems and Networks Conference (DSN-2015)*. Rio de Janeiro, Brazil: [s.n.], 2015.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Thomson Course Technology Boston, 2006. v. 2.

SOUSA, E.; MACIEL, P. R. M.; ARAúJO, C.; ALVES, G.; CHICOUT, F. Performance modeling for evaluation and planning of electronic funds transfer systems. In: *ISCC'09*. [S.l.: s.n.], 2009. p. 73–76.

SUKHWANI, H.; MARTÍNEZ, J. M.; CHANG, X.; TRIVEDI, K. S.; RINDOS, A. Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In: IEEE. *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. [S.l.], 2017. p. 253–255.

SUKHWANI, H.; WANG, N.; TRIVEDI, K. S.; RINDOS, A. Performance modeling of hyperledger fabric (permissioned blockchain network). In: IEEE. *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. [S.l.], 2018. p. 1–8.

SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: "O'Reilly Media, Inc.", 2015.

TAPSCOTT, D.; TAPSCOTT, A. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. [S.l.]: Penguin, 2016.

TRIVEDI, K. S. *Probability and statistics with reliability, queuing, and computer science applications*. [S.l.]: Wiley Online Library, 1982. v. 13.

TRIVEDI, K. S.; BOBBIO, A. *Reliability and availability engineering: modeling, analysis, and applications*. [S.l.]: Cambridge University Press, 2017.

TRIVEDI, K. S.; HUNTER, S.; GARG, S.; FRICKS, R. *Reliability Analysis Techniques Explored Through a Communication Network Example*. 1996.

WEBER, I.; GRAMOLI, V.; PONOMAREV, A.; STAPLES, M.; HOLZ, R.; TRAN, A. B.; RIMBA, P. On availability for blockchain-based systems. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. [S.l.: s.n.], 2017. p. 64–73.

WHITT, W. Sensitivity of performance in the erlang-a queueing model to changes in the model parameters. *Operations research*, INFORMS, v. 54, n. 2, p. 247–260, 2006.

XIONG, Z.; FENG, S.; WANG, W.; NIYATO, D.; WANG, P.; HAN, Z. Cloud/fog computing resource management and pricing for blockchain networks. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4585–4600, June 2019. ISSN 2327-4662.

ZHENG, K.; LIU, Y.; DAI, C.; DUAN, Y.; HUANG, X. Model checking pbft consensus mechanism in healthcare blockchain network. In: IEEE. *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. [S.l.], 2018. p. 877–881.

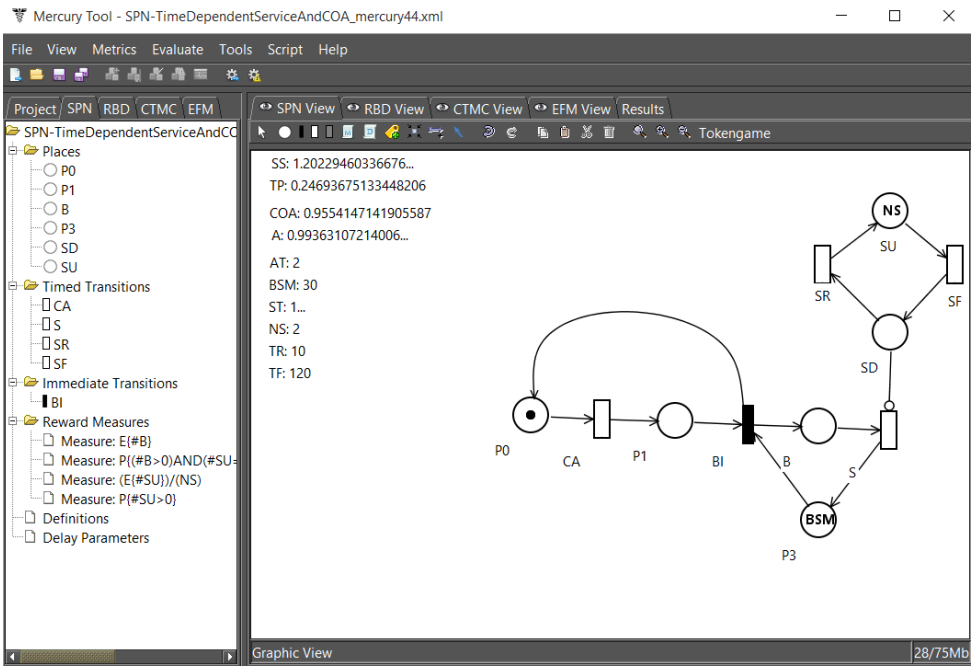
ZYSKIND, G.; NATHAN, O.; PENTLAND, A. . Decentralizing privacy: Using blockchain to protect personal data. In: *2015 IEEE Security and Privacy Workshops*. [S.l.: s.n.], 2015. p. 180–184.

APÊNDICE A – FERRAMENTA MERCURY

O Mercury[1] é uma ferramenta desenvolvida pelo grupo de pesquisa Modeling of Distributed Concurrent System (MoDCS) no Centro de Informática da Universidade Federal de Pernambuco (CIn - UFPE). Esta ferramenta, permite a criação e avaliação dos mais diversos modelos analíticos e de seus atributos, componentes e métricas relacionadas tanto a desempenho quanto a dependabilidade de sistemas computacionais (SILVA et al., 2015), tendo importante papel no processo de desenvolvimento desta proposta de tese.

Com este ambiente integrado de desenvolvimento e modelagem gráfica é possível representar os seguintes formalismos: Continuous Time Markov Chains (CTMC), Deterministic Time Markov Chains (DTMC), Stochastic Petri Nets (SPN), Fault Trees (FT), Energy Flow Model (EFM) e Reliability Block Diagrams (RBD). Através de técnicas de análise de sensibilidade implementadas no Mercury, é possível, determinar os componentes que possuem maior importância dentro dos modelos de sistemas criados, além da possibilidade de exportação dos resultados obtidos para análise em outras ferramentas matemáticas, como o Mathematica, R e Excel. A Figura 45 apresenta o exemplo de uma SPN no ambiente gráfico da ferramenta Mercury.

Figura 45 – Exemplo de uma SPN na ferramenta Mercury



Fonte: (SILVA et al., 2015)

Já por vias textuais, o Mercury proporciona um ambiente de *scripting* para escrita e automação no processo de modelagem, avaliação e análise de sensibilidade dos modelos

[1] Mercury: https://bit.ly/2kpuRu9

desenvolvidos, seja diretamente na plataforma ou através de suas APIs.

APÊNDICE B – VALIDAÇÃO

No âmbito desta tese fez-se necessária a realização da validação dos valores utilizados para alguns dos componentes presentes nos modelos propostos. A validação é caracterizada como um requisito primário para as conclusões tiradas a partir das abstrações do sistema real.

Optamos por validar os componentes que garantem a operacionalidade do Hyperledger Fabric, ou seja, os três contêineres base: Endorser, MSP e Orderer. O tempo até a falha (MTTF) utilizado para os três contêineres no processo de modelagem foi de 1258 horas, enquanto que o tempo até o reparo (MTTR) estipulado foi de 0.15 horas ou 9 minutos. A Tabela 27 apresenta o fator de redução utilizado para o MTTF.

Tabela 27 – Fator de Redução MTTF (Contêineres)

Componente	**MTTF Modelo (h)**	**MTTF Aplicado (h)**	**Reparo (h)**
Endorser-MSP-Orderer	1258	1,25	0,15

Fonte: Autoria Própria

O período de 1258 horas corresponde à aproximadamente 52 dias. Como não se dispõe de um intervalo tão grande de tempo até que o sistema falhe, pode-se utilizar de fatores de redução ou de aceleração de falha (ARAUJO et al., 2011). Para tanto, um método inicialmente proposto por (EJLALI et al., 2003) foi utilizado, este que consiste no uso de *scripts* bash com o único propósito de parar a execução do contêiner, ou seja, injetar falhas propositais no sistema.

O fator de aceleração utilizado é da ordem de 1000 vezes, ou seja, se o tempo estimado para a falha do componente era de 1258 horas, utilizando-se de 1,25 horas para cada execução do script, teremos então, um valor acelerado correspondente. O script foi executado 30 vezes para cada contêiner ao longo de aproximadamente 7 dias.

Após a coleta do *timestamp* correspondente às falhas e reparos dos componentes, os métodos estatísticos propostos em (KEESEE, 1965) foram utilizados. Tais métodos, tentam verificar o pertencimento dos valores utilizados em nossos modelos dentro do intervalo de confiança do experimento.

Para a nossa avaliação, a distribuição-F foi utilizada em conjunto da ferramenta Minitab, visando um intervalo de confiança na ordem de 95%. A Tabela 28 apresenta os valores mínimos e máximos providos pela distribuição-F.

A relação entre os tempos para falha e reparo resultou em um número ρ, obtido através da Equação $\rho = \frac{F_n}{R_n}$, onde F_n corresponde ao total que o componente esteve em estado de falha e R_n ao tempo total em funcionamento. É importante ressaltar que ρ permitirá encontrar valores mínimos e máximos que irão representar o intervalo de confiança.

Tabela 28 – Intervalos para a distribuição-F

Descrição	**Valor**
Valor mínimo	0,67
Valor máximo	1,43

Fonte: Autoria Própria

De posse do valor de ρ e dos valores providos pela distribuição-F, podemos calcular o valor mínimo (ρL) e máximo (ρU) de ρ. Além disso, precisamos encontrar os já mencionados valores mínimos e máximos para disponibilidade, ou seja, A_L e A_U respectivamente. Esses valores fornecerão a validação do modelo se o valor encontrado pertencer ao intervalo de disponibilidade resultante. As Expressões B.1 e B.2 apresenta como são calculados os valores mínimos e máximos que compõem o intervalo de confiança para a disponibilidade.

$$A_L = \frac{1}{\rho L} \tag{B.1}$$

$$A_U = \frac{1}{\rho U} \tag{B.2}$$

A Tabela 29 apresenta um resumo geral com os valores mínimos e máximos para os intervalos de confiança tanto de ρ quanto da disponibilidade dos componentes de acordo com o método proposto por (KEESEE, 1965).

Tabela 29 – Intervalo de Confiança para A e ρ

Intervalo de Confiança de 95%		
ρ	ρ_U	0,00010
	ρ_L	$7{,}49\times10^{-5}$
A	A_U	0,99981
	A_L	0,99961

Fonte: Autoria Própria

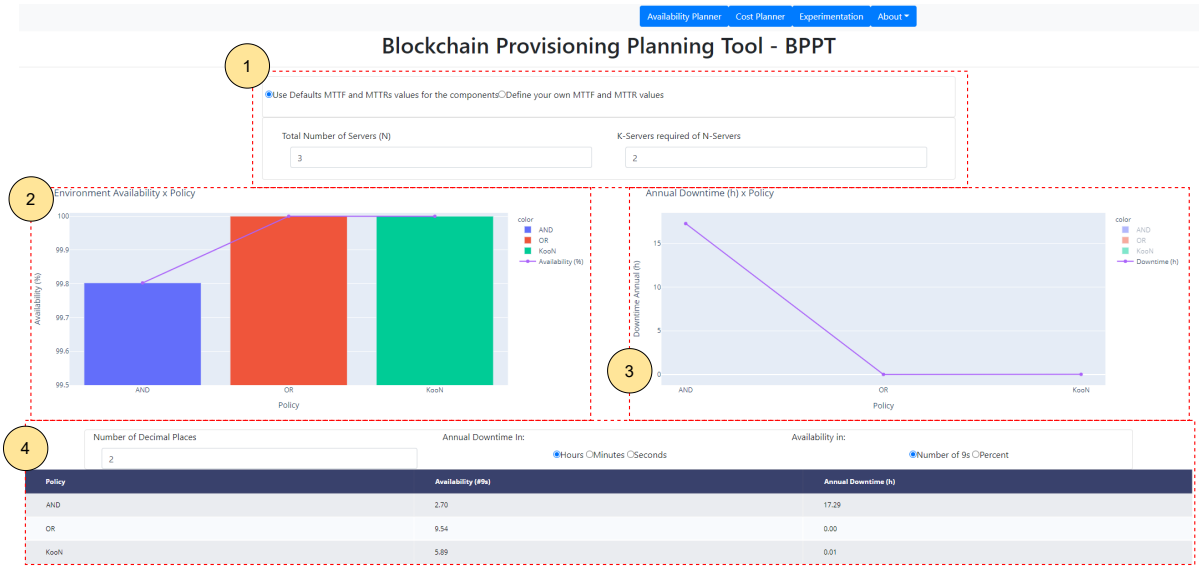
O valor encontrado na avaliação da disponibilidade através da ferramenta Mercury para os três contêineres que garantem a operação do Hyperledger Fabric foi de 0,99964. Este valor está dentro do intervalo de confiança estabelecido de [0,99961 - 0,99981], ou seja, consideramos que o modelo foi validado com 95% de confiança.

APÊNDICE C – O BPPT EM IMAGENS

C.1 PLANEJADOR DE DISPONIBILIDADE

O planejador de disponibilidade se subdivide em quatro partes, a saber (1) definição de parâmetros de entrada; (2) relação entre disponibilidade e política de endossamento; (3) relação entre o downtime anual e política de endossamento; (4) resumo geral com comparativo entre políticas e métricas através de uma tabela. A Figura 46 apresenta o planejador de disponibilidade.

Figura 46 – Planejador de Disponibilidade



Fonte: Autoria Própria

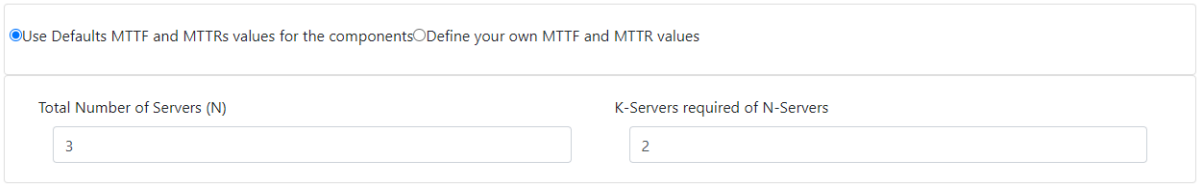
C.1.1 Definição de parâmetros de entrada

Através do planejador de disponibilidade o administrador ou interessado poderá alterar um conjunto de parâmetros relacionados a plataforma que está sendo planejada, dentre os principais parâmetros que podemos citar estão os respectivos tempos de falha ou reparo para cada componente da arquitetura. O avaliador poderá defini-los (Ver Figura 47b) ou simplesmente utilizar de valores-padrão pré-estabelecidos com base em levantamento bibliográfico (Ver Figura 47a).

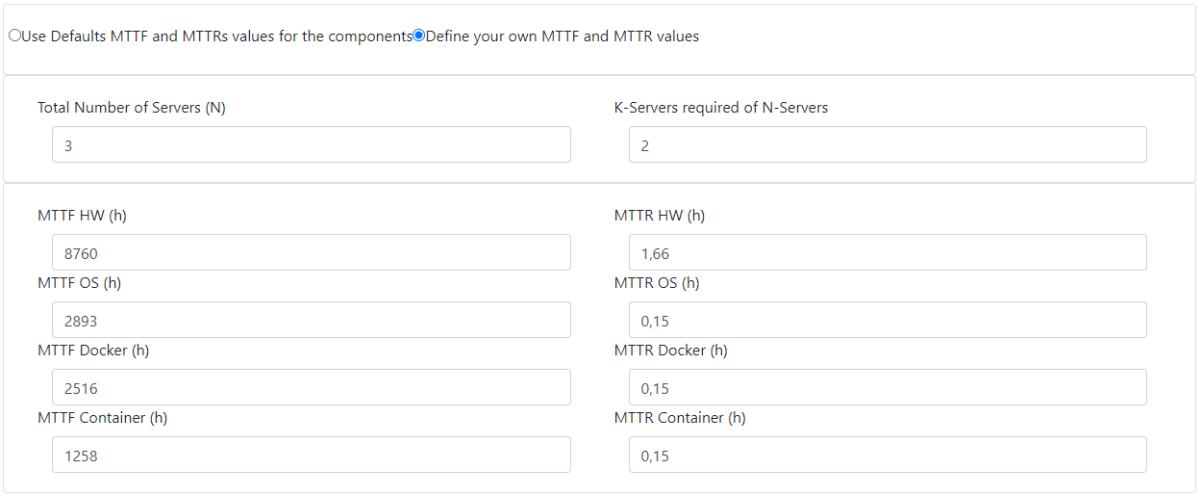
Um outro importante parâmetro condizente com a avaliação das métricas de disponibilidade é o número de servidores operacionais e as políticas de endorsamento de transações que serão utilizada na plataforma. Essa escolha se dá através da especificação de uma quantidade mínima de recursos necessários a operação do sistema através do valor K. É importante salientar que quando o tomador de decisão optar por uma política de endorsa-

Figura 47 – Parâmetros de Entrada

(a) Valores-padrão



(b) Definição de Valores de Entrada



Fonte: Autoria Própria

mento do tipo K-out-of-N esse valor deverá ser sempre menor ou igual ao número máximo de servidores disponível.

C.1.2 Relação entre disponibilidade e política de endossamento

Através dessa relação o avaliador poderá constatar os indices de disponibilidade alcançados pelo sistema com base nos valores fornecidos como entrada. É possível alterar a visão do gráfico de linhas para barras ou combiná-las. Aqui a disponibilidade é avaliada em porcentagem.

É possível, também, visualizar comparativos apenas entre políticas de interesse através da caixa de seleção ou legenda do gráfico, além de aplicar zoom ao gráfico caso os valores de disponibilidade sejam muito próximos. Ao passar o mouse seja pela barra ou ponto podemos ver até quatro casas decimais, o que pode ser determinante na escolha de um ambiente altamente disponível (Com cinco ou mais noves de disponibilidade).

C.1.3 Relação entre o downtime anual e política de endossamento

Essa relação apresenta o downtime anual em horas e sua relação com cada política de endossamento. É importante ressaltar que assim como na relação entre disponibilidade

Figura 48 – Disponibilidade x Política de endossamento

Fonte: Autoria Própria

e política de endossamento o avaliador também poderá optar por uma visualização em barras ou em linha.

Figura 49 – Downtime Anual x Política de endossamento

Fonte: Autoria Própria

C.1.4 Resumo geral

O resumo geral de disponibilidade consiste em uma tabela que apresenta os valores de disponibilidade e downtime anual para as diferentes políticas de endossamento. É através dessa nova visão que o avaliador poderá alternar o número de casas decimais apresentado para as métricas, se quer ver os valores de disponibilidade em porcentagem ou através do número de noves, e ainda se quer visualizar o downtime anual em horas, minutos ou segundos da forma que melhor atenda suas necessidades.

Figura 50 – Resumo Geral

Policy	Availability (#9s)	Annual Downtime (h)
AND	2.70	17.29
OR	9.54	0.00
KooN	5.89	0.01

Fonte: Autoria Própria

C.2 PLANEJADOR DE CUSTOS

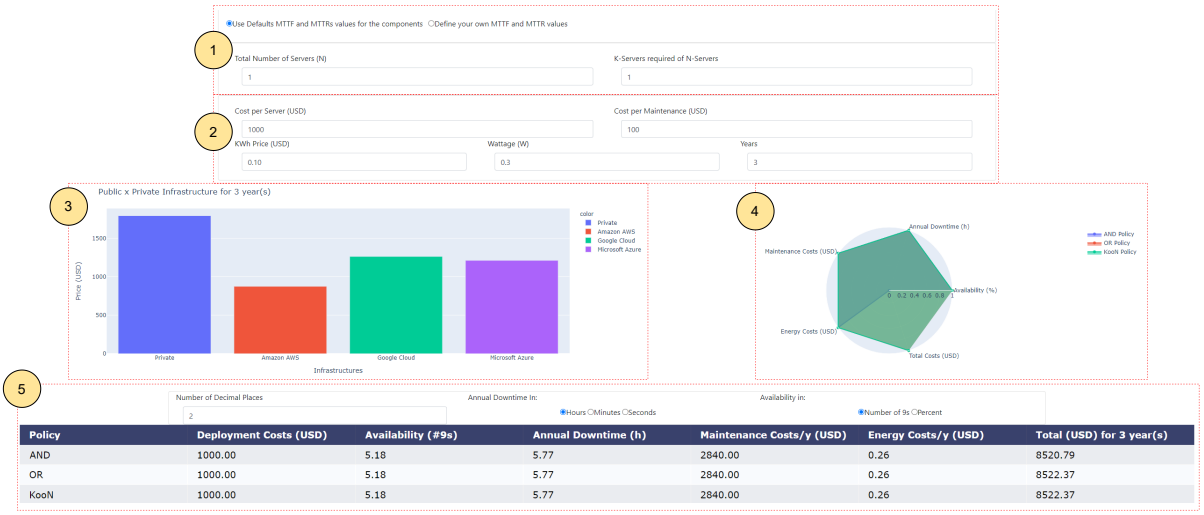
A Figura 51 apresenta a visão geral do planejador de custos do BPPT.

Subdividimos o planejador de custos em cinco partes, a saber (1) parâmetros de entrada de disponibilidade e (2) parâmetros de entrada de custo; (3) relação entre custo em dólares de infraestrutura pública e infraestrutura privada em período de tempo pré-determinado; (4) relação entre políticas de endossamento, métricas de disponibilidade e custos; (5) resumo geral de custos e disponibilidade.

C.2.1 Parâmetros de entrada

Assim como no planejador de disponibilidade, os parâmetros de entrada referentes aos tempos de falha e reparo podem, também, ser definidos aqui, o que possibilita um novo grau de comparação ao analista, este que trata-se da viabilidade entre custo e disponibilidade para implantação de sua respectiva infraestrutura (Ver Figura 52). Todavia, o usuário poderá, também, definir os custos de aquisição por servidor, os custos de manu-

Figura 51 – Planejador de Custos - Parte Superior

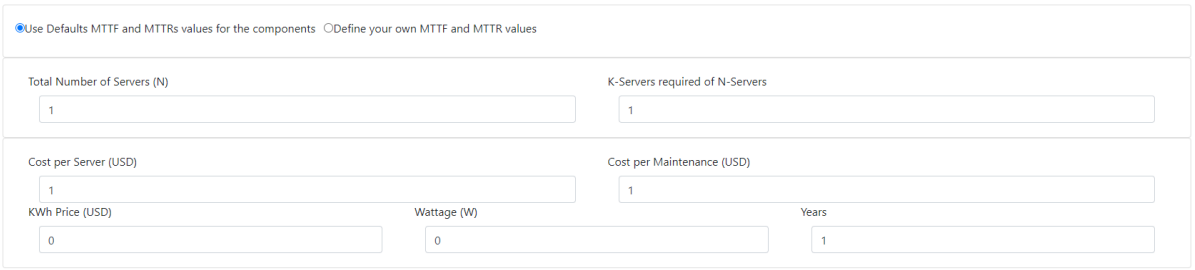


Policy	Deployment Costs (USD)	Availability (#9s)	Annual Downtime (h)	Maintenance Costs/y (USD)	Energy Costs/y (USD)	Total (USD) for 3 year(s)
AND	1000.00	5.18	5.77	2840.00	0.26	8520.79
OR	1000.00	5.18	5.77	2840.00	0.26	8522.37
KooN	1000.00	5.18	5.77	2840.00	0.26	8522.37

Fonte: Autoria Própria

tenção especificados em acordo de provimento de serviço, além do custo do quilowatt-hora na região, preferencialmente em dólar americano.

Figura 52 – Parâmetros de Entrada do Planejador de Custos



Fonte: Autoria Própria

Também é possível se estabelecer a potência em watts (W) dos servidores e o período esperado de utilização do mesmo, estabelecendo assim o consumo energético do ambiente como um todo para o período de um ano. É importante salientar uma vez mais, que o custo de refrigeração atribuído a todo ambiente é considerado igual ao custo energético da infraestrutura, ou seja, é igual ao custo de operação dos servidores.

C.2.2 Infraestrutura Pública x Privada

A nível de comparação, nós podemos escolher entre plataformas de computação em nuvem, como Google Cloud, Amazon AWS e Microsoft Azure. Essas três plataformas proveem ambientes que podem ser utilizados para hospedar aplicação que rodem em cima do Hyperledger Fabric. Nós escolhemos as instâncias mais básicas de cada plataforma e estipulamos sua utilização sob demanda (on-demand) pelo período de 365 dias. A Tabela 30 sumariza os resultados gerais para os ambientes selecionados.

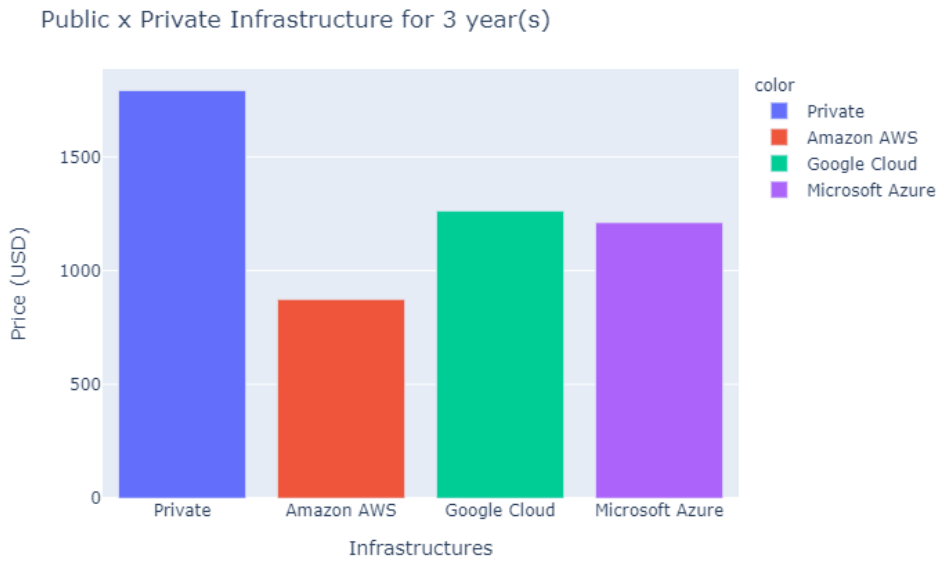
Tabela 30 – Despesas Anuais com Instâncias em Nuvens Públicas

Platforma	**Tipo de Instância**	**On-demand por ano (USD)**	**Disponibilidade (%)**
Google Cloud	n1-standard-1	291,24	99,99
Amazon AWS	m3.medium	420,48	99,99
Microsoft Azure	D1s	403,56	99,99

Fonte: Autoria Própria

Através dos parâmetros de entrada estabelecidos no planejador de custos poderemos realizar um comparativo de alto nível entre a implantação de infraestrutura privada e o custo de instâncias de nuvem pública. Esse comparativo se dá através de um gráfico de barras (Ver Figura 53).

Figura 53 – Infraestrutura Pública x Privada



Fonte: Autoria Própria

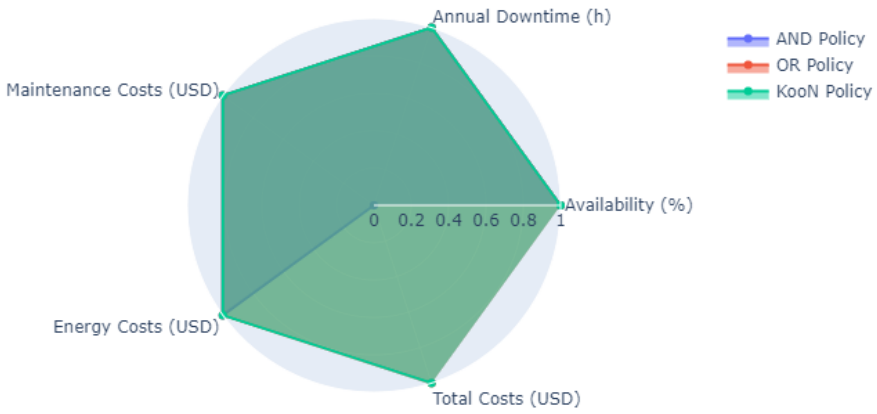
É sempre importante destacar que o custo de aquisição é relativo ao primeiro ano da infraestrutura, e isso deve sempre ser levado em consideração quando o avaliador se deparar com valores discrepantes entre nuvens públicas e infraestruturas privadas. Outro dado importante é o período de avaliação em anos, este que pode ser subentendido como a expectativa de vida que o avaliador tem sobre sua infraestrutura.

Outro importante fator é o custo de manutenção, para o comparativo com infraestruturas públicas avaliamos esse custo considerando apenas uma política de endossamento do tipo AND, que implica que ao se constatar a falha de um nó a equipe de manutenção deverá ser chamada e aplicar os devidos reparos. A subseção C.2.4 apresenta mais detalhes inerentes a manutenção de infraestruturas sob outros tipos de políticas de endossamento.

C.2.3 Relação entre políticas de endossamento, disponibilidade e custo

Essa relação é descrita através de um gráfico radar, que pode ser subentendido como uma relação custo x benefício entre as políticas de endossamento, o downtime anual em horas e o custo total de operação da infraestrutura pelo período previamente estabelecido (Ver Figura 54).

Figura 54 – Política de endossamento x Custos x Disponibilidade



Fonte: Autoria Própria

O analista pode alterar a visualização de acordo com a política de interesse e determinar se a mesma é viável ou não aos seus planos. Todavia, algumas características podem ser comuns a maioria das infraestruturas, como por exemplo, a política de endossamento do tipo AND, que geralmente apresenta um maior número de falhas já que é necessário que todas as máquinas encontrem-se operacionais para que o serviço seja devidamente provido. Deste modo, a manutenção será efetuada mais vezes do que seria em uma política de endossamento do tipo OR, ou ainda, por vezes, mais frequentemente do que em uma política de endossamento do tipo KooN.

C.2.4 Resumo Geral - endossamento x Custos x Disponibilidade

O resumo geral, mais uma vez, é apresentado em forma de tabela. Essa tabela engloba todas as informações pertinentes a custos de manutenção, energético e total; além de valores de disponibilidade, downtime anual e sua relação com as três políticas de endossamento contempladas pelo BPPT.

Figura 55 – Resumo Geral - endossamento x Custos x Disponibilidade

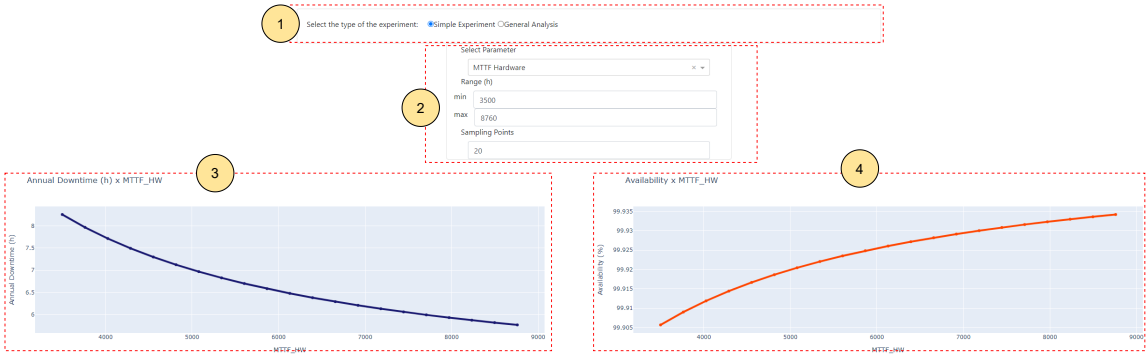
Policy	Deployment Costs (USD)	Availability (#9s)	Annual Downtime (h)	Maintenance Costs/y (USD)	Energy Costs/y (USD)	Total (USD) for 3 year(s)
AND	1000.00	5.18	5.77	2840.00	0.26	8520.79
OR	1000.00	5.18	5.77	2840.00	0.26	8522.37
KooN	1000.00	5.18	5.77	2840.00	0.26	8522.37

Fonte: Autoria Própria

C.3 PLANEJADOR DE EXPERIMENTOS

Podemos subdividi-lo em quatro partes, a saber (1) tipo de experimento; (2) intervalo de variação; (3) disponibilidade x parâmetro variado; (4) downtime anual x parâmetro variado. A Figura 56 provê uma visão geral do planejador de experimentos.

Figura 56 – Planejador de Experimentos



Fonte: Autoria Própria

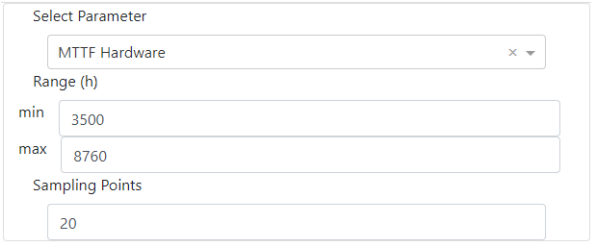
C.3.1 Tipo de experimento e Intervalo de variação

O intervalo de variação se caracteriza pelos tempos mínimos e máximos relacionados a falha e ao reparo dos componentes do sistema. Como visto anteriormente, a definição desses valores se dá com base no tipo de experimento escolhido pelo avaliador, podendo referenciar um único componente em específico (Ver Figuras 57a) ou todos os componentes da plataforma (Ver Figuras 57b).

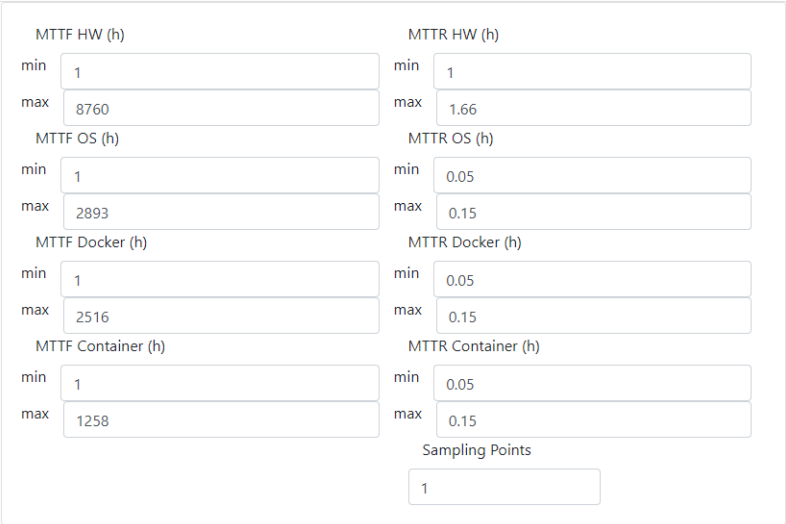
O usuário também poderá estabelecer o número de pontos amostrais que desejar em seu gráfico e um intervalo positivo de horas que o parâmetro em questão varia. Este tipo de avaliação pode ser considerado uma análise de sensibilidade do tipo paramétrico, assim, ao juntar-se todos os componentes e suas variações é possível analisar o impacto de cada um deles sobre a métrica de interesse, seja esta a disponibilidade geral ou o downtime anual do sistema.

Figura 57 – Parâmetros de Entrada

(a) Intervalo para único componente



(b) Intervalo para todos os componentes



Fonte: Autoria Própria

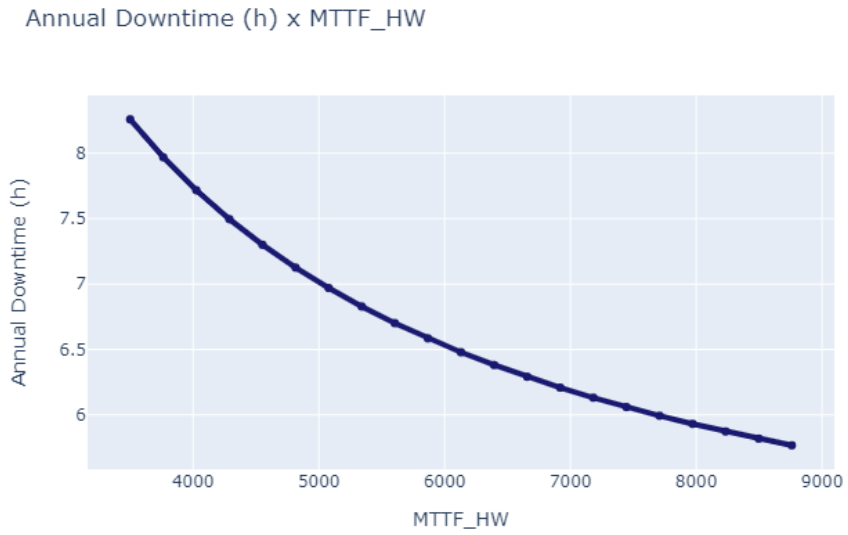
C.3.2 Métrica de Interesse x parâmetro(s) variado

Definindo-se um único parâmetro através da seleção de experimento do tipo simples, é possível obtermos dois gráficos complementares referentes ao downtime anual (Ver Figura 58) e a disponibilidade do sistema como um todo (Ver Figura 59). A curva do gráfico dependerá do número de pontos amostrais previamente estabelecidos e do real impacto do parâmetro de entrada sobre a métrica de interesse. A disponibilidade ou downtime em cada ponto pode ser consultado com o mouse.

Já no experimento do tipo geral o foco é a disponibilidade geral do sistema e como todos os possíveis parâmetros de entrada a afetam. É tarefa do avaliador modificar e estabelecer um intervalo a cada um dos parâmetros, ou, mais uma vez, se utilizar de valores de entrada obtidos através de revisão da literatura.

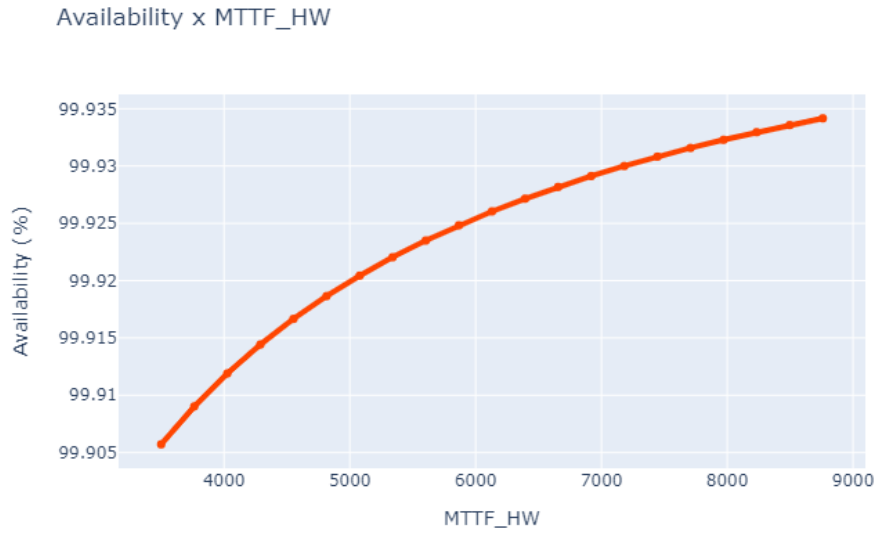
Se definirmos valores fixos a todos os outros parâmetros exceto a um, estaremos tratando de uma análise de sensibilidade, como anteriormente mencionado. Mais uma vez é possível alternar a visão entre um parâmetro e outro através da caixa de seleção ou legenda do gráfico. Deste modo apenas aqueles de maior interesse do avaliador poderão ser consultados, dando indicativo de possíveis melhorias sejam elas de implementação ou

Figura 58 – Downtime x Parâmetro



Fonte: Autoria Própria

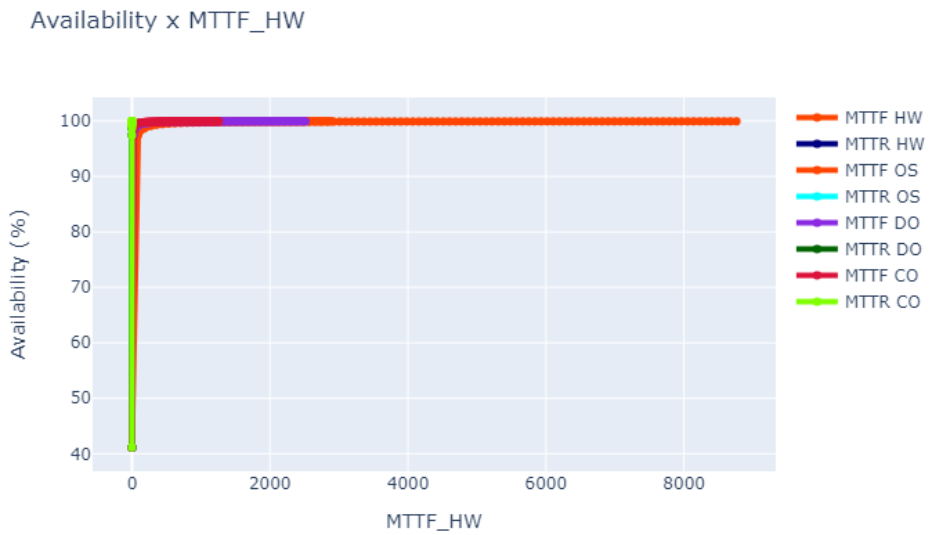
Figura 59 – Disponibilidade x Parâmetro



Fonte: Autoria Própria

de hardware. A Figura 60 mostra um exemplo de experimento geral e todos os parâmetros associados.

Figura 60 – Disponibilidade x Todos os Parâmetros



Fonte: Autoria Própria