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

CARLOS GOMES ARAÚJO

Uma abordagem para análise do impacto da consistência de dados no desempenho e disponibilidade em SGBDs NoSQL

Recife

2022

CARLOS GOMES ARAÚJO

Uma abordagem para análise do impacto da consistência de dados no desempenho e disponibilidade em SGBDs NoSQL

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. Área de concentração: Redes de Computadores e Sistemas Distribuídos.

Orientador: Eduardo Antônio Guimarães Tavares

Coorientador: Meuse Nogueira de Oliveira Junior

Recife

2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A663a Araújo, Carlos Gomes
Uma abordagem para análise do impacto da consistência de dados no desempenho e disponibilidade em SGBDs NoSQL / Carlos Gomes Araújo. – 2022.
138 f.: il., fig., tab.

Orientador: Eduardo Antônio Guimarães Tavares.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e anexos.

1. Redes de computadores. 2. Consistência. 3. Desempenho. I. Tavares, Eduardo Antônio Guimarães (orientador). II. Título.

004.6 CDD (23. ed.) UFPE - CCEN 2023-07

Carlos Gomes Araújo

“Uma abordagem para análise do impacto da consistência de dados no desempenho e disponibilidade em SGBDs NoSQL”

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: 31/08/2022.

_____

Orientador: Prof. Dr. Eduardo Antonio Guimarães Tavares

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática / UFPE

_____

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática / UFPE

_____

Profa. Dra. Aida Araújo Ferreira
Instituto Federal de Pernambuco / Campus Recife

_____

Prof. Dr. Edmundo Roberto Mauro Madeira
Instituto de Computação / UNICAMP

_____

Prof. Dr. Paulo Hugo Espírito Santo Lima
Escola Politécnica de Pernambuco / UPE

Dedico este trabalho a Deus e a meus pais, cujos ensinamentos e exemplos sempre me guiam; a minha família, em especial, minha esposa Jessica, que esteve sempre ao meu lado em todas as dificuldades, e a meus filhos, alicerces que me fazem continuar.

AGRADECIMENTOS

Agradeço a Deus e a minha família, especialmente minha esposa Jessica e a meus filhos, Maria Alice e Heitor; a minha mãe, Maria do Socorro e a meu pai, Arlindo Gomes *im memoriam*, os quais foram os alicerces para a dedicação necessária à obtenção desses resultados.

Agradeço ao meu professor orientador, Eduardo Tavares, pelo comprometimento desde o início do mestrado em 2014, quando ele acreditou em meu potencial, até esse doutorado. Desde então, muitos desafios de ordem pessoal e profissional surgiram e seu comprometimento foi fundamental para a minha evolução acadêmica e profissional.

Da mesma forma, agradeço ao meu coorientador Meuse Nogueira, que vem me acompanhando desde o mestrado e que contribuiu significativamente para os atuais resultados.

Agradeço a todos os professores, a todos os componentes de bancas e revisores de artigos, a todos os que contribuíram para a evolução da pesquisa.

Agradeço, ainda, ao leitor e a todos os que acreditam na pesquisa e na ciência como instrumento de evolução social.

RESUMO

Na última década, o volume de dados gerado e armazenado por computadores aumentou consideravelmente. Os dados são gerados por várias fontes, inclusive por sistemas autônomos. Para atender à demanda do gerenciamento desses dados, as tecnologias de banco de dados convencionais apresentam limitações devido à rigidez de suas arquiteturas. Adjacente a essa necessidade de crescimento dos dados, populariza-se o desenvolvimento dos sistemas gerenciadores de banco de dados (SGBD) NoSQL, (*Not Only SQL*). NoSQL é uma tecnologia de banco de dados concebida principalmente para ultrapassar a escalabilidade horizontal, o desempenho, a disponibilidade e as limitações dos SGBDs relacionais. Para melhorar seu desempenho e disponibilidade, SGBDs NoSQL também oferecem a consistência eventual, a qual permite, temporariamente, uma inconsistência entre os servidores redundantes. Importantes SGBDs NoSQL dispõem da possibilidade de configuração da consistência baseada em quorum, que representa o número de servidores que devem confirmar uma operação para que ela seja concluída. Neste contexto, um grande desafio é quantificar e avaliar o impacto no desempenho, na disponibilidade, na probabilidade de acessar o dado mais recente e no consumo de energia das operações em NoSQL baseado em quorum. Diversos trabalhos avaliam SGBDs NoSQL, mas a estimativa do impacto da escolha da consistência nessas métricas ainda é pouco representada na literatura. As possibilidades da escolha de diversas arquiteturas e configurações em SGBDs NoSQL baseadas em quorum requerem ferramentas e técnicas de avaliação para que um projetista possa selecionar a melhor abordagem na implementação de sistemas. Esta tese de doutorado apresenta e avalia um conjunto de modelos que permitem quantificar o desempenho, a disponibilidade, a probabilidade de acesso ao dado mais recente e o consumo de energia no SGBD baseado em quorum. A proposta utiliza rede de Petri estocástica e diagrama de blocos de confiabilidade para modelar esses sistemas. Um estudo de caso verificou que elevar a consistência do nível 1 para o nível 3 melhorou a probabilidade de acesso ao dado mais recente em 21%, no entanto, aumentou o tempo das operações; a indisponibilidade do sistema foi elevada de 3 segundos para 3 horas.

Palavras-chaves: SPN; RBD; consistência; desempenho; disponibilidade; energia.

ABSTRACT

In the last decade, the volume of data generated and stored by computers has increased considerably. This data is generated by various sources, including autonomous systems. To meet the demand for managing this data, conventional database technologies have limitations due to the inflexibility of their architectures. Adjacent to this need for data growth, the development of database management systems (DBMS) NoSQL (Not Only SQL) is popular. NoSQL is a database technology designed for cloud applications and conceived primarily to overcome the horizontal scalability, performance, availability, and limitations of relational DBMSs. To improve performance and availability, NoSQL DBMS also offers eventual consistency, which temporarily allows for an inconsistency between redundant servers. Additionally, important NoSQL DBMSs have the possibility of configuring levels of consistency based on quorum, which represents the number of servers that must confirm an operation for it to be completed. In this context, a major challenge is to quantify and evaluate the impact on performance, availability, probability of accessing the latest data, and power consumption of quorum-based NoSQL operations. Several works evaluate NoSQL DBMSs, but estimating the impact of choosing consistency on these metrics is still underrepresented in the literature. The possibilities of choosing different architectures and configurations in quorum-based NoSQL DBMSs require evaluation tools and techniques so that a designer can select the best approach to implementing these systems. This doctoral thesis presents and evaluates models that quantify the performance, availability, probability of access to the most recent data, and energy consumption in the quorum-based DBMS. The propose uses stochastic Petri nets and reliability block diagram to model quorum-based NoSQL data storage systems. The results performed demonstrate the impact of consistency on the evaluated metrics. In the case study it was found that raising the consistency from level one to level three improved the probability of accessing the most recent data by 21%. However, it increased operations time. System unavailability was increased from 3 seconds to 3 hours.

Keywords: SPN; RBD; consistency; performance; availability; energy.

LISTA DE FIGURAS

Figura 1 – Exemplo de níveis de consistência (um, dois e três) 30
Figura 2 – Classificação de técnicas de injeção de falhas 39
Figura 3 – Elementos de uma rede de Petri 41
Figura 4 – Estados de um sistema 41
Figura 5 – Exemplo de grafo de alcançabilidade 42
Figura 6 – Arco inibidor 45
Figura 7 – Exemplo de SPN 46
Figura 8 – Diagrama de bloco de confiabilidade – Arranjo em série 48
Figura 9 – Diagrama de bloco de confiabilidade – Arranjo paralelo 48
Figura 10 – Área sob curva normal 51
Figura 11 – Método proposto 63
Figura 12 – Modelo de desempenho e disponibilidade 73
Figura 13 – Modelo de probabilidade de consistência NoSQL 76
Figura 14 – Modelos RBD: (a) *hardware*; e (b) réplicas do SGBD 78
Figura 15 – Modelos RBD 78
Figura 16 – Setup 82
Figura 17 – Gráficos de efeito – latência 90
Figura 18 – Gráficos de efeito – disponibilidade 92
Figura 19 – Gráficos de efeito – probabilidade de acesso ao dado mais recente 96
Figura 20 – Gráficos de efeito – consumo de energia 98
Figura 21 – (Anexo A) Modelo de desempenho e disponibilidade - 01 117
Figura 22 – (Anexo A) Modelo de desempenho e disponibilidade - 02 118
Figura 23 – (Anexo A) Modelo de desempenho e disponibilidade - 03 119
Figura 24 – (Anexo A) Modelo de desempenho e disponibilidade - 04 120
Figura 25 – (Anexo A) Modelo de desempenho e disponibilidade - 05 121
Figura 26 – (Anexo A) Modelo de desempenho e disponibilidade - 06 122
Figura 27 – (Anexo A) Modelo de desempenho e disponibilidade - 07 123
Figura 28 – (Anexo A) Modelo de desempenho e disponibilidade - 08 124
Figura 29 – (Anexo A) Modelo de desempenho e disponibilidade - 09 125
Figura 30 – (Anexo A) Modelo de desempenho e disponibilidade - 10 126

Figura 31 – (Anexo A) Modelo de desempenho e disponibilidade - 11 127
Figura 32 – (Anexo A) Modelo de desempenho e disponibilidade - 12 128
Figura 33 – (Anexo A) Modelo de desempenho e disponibilidade - 13 129
Figura 34 – (Anexo A) Modelo de desempenho e disponibilidade - 14 130
Figura 35 – (Anexo A) Modelo de desempenho e disponibilidade - 15 131
Figura 36 – (Anexo A) Modelo de desempenho e disponibilidade - 16 132
Figura 37 – (Anexo B) Modelo de acesso ao dado - 01 133
Figura 38 – (Anexo B) Modelo de acesso ao dado - 02 134
Figura 39 – (Anexo B) Modelo de acesso ao dado - 03 135
Figura 40 – (Anexo B) Modelo de acesso ao dado - 04 136
Figura 41 – (Anexo B) Modelo de acesso ao dado - 05 137
Figura 42 – (Anexo B) Modelo de acesso ao dado - 06 138

LISTA DE TABELAS

Tabela 1 – Número de noves equivalentes para disponibilidade 32
Tabela 2 – Exemplo fatorial 37
Tabela 3 – Comparação entre esta tese e trabalhos relacionados 60
Tabela 4 – Atributos das transições do modelo de desempenho e disponibilidade . . . 75
Tabela 5 – Atributos das transições do modelo de acesso ao dado 77
Tabela 6 – Configurações das máquinas virtuais 82
Tabela 7 – Parâmetros YCSB 84
Tabela 8 – Tempo das transições para a validação do desempenho com o Sistema Gerenciador de Banco de Dados (SGBD) Cassandra 84
Tabela 9 – Tempo das transições para a validação do desempenho com o SGBD Riak KV 85
Tabela 10 – *Mean Time to Failure* (MTTF) e *Mean Time to Repair* (MTTR) dos componentes do modelo 85
Tabela 11 – Tempos de transição para validação do modelo de probabilidade da consistência 88
Tabela 12 – Classificação dos efeitos principais e de interação (latência) 89
Tabela 13 – Resultados do projeto de experimentos para latência 91
Tabela 14 – MTTF e MTTR dos experimentos 92
Tabela 15 – Classificação dos efeitos principais e de interação (*downtime*) 93
Tabela 16 – Resultados do projeto de experimento para disponibilidade 94
Tabela 17 – Classificação dos efeitos principais e de interação (probabilidade) 95
Tabela 18 – Resultados do projeto de experimentos para probabilidade 96
Tabela 19 – Classificação dos efeitos principais e de interação (consumo de energia) . . 98
Tabela 20 – Resultados do projeto de experimentos para consumo de energia 99
Tabela 21 – Tempos de transição para desempenho, estudo de caso 01 – SGBD (Cassandra) 101
Tabela 22 – Tempos de transição para desempenho, estudo de caso 01 – SGBD (Riak) 101
Tabela 23 – Tempos de transição para consistência, estudo de caso 01 – SGBD (Cassandra) 101
Tabela 24 – Tempos de transição para consistência, estudo de caso 01 – SGBD (Riak) . 101

Tabela 25 – Resultado do estudo de caso 01 102
Tabela 26 – MTTF e MTTR dos componentes do modelo 103
Tabela 27 – Resultado do estudo de caso 02 103

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	*Application Programming Interface*
AWS	*Amazon Web Services*
BASE	*Basically Available, Soft state, Eventually consistent*
CPN	*Colored Petri net*
CS	Consistência Sequencial
CTMC	*Continuous-Time Markov Chains*
DBaaS	*Database as a service*
DoE	*Design of Experiments*
GSPN	*Generalized Stochastic Petri net*
HDFS	*Hadoop Distributed File System*
MTTF	*Mean Time to Failure*
MTTR	*Mean Time to Repair*
OCC	Observable Causal Consistency
PBS	*Probabilistically Bounded Staleness*
PCG	Probabilistic Consistency Guarantee
PN	*Petri net*
QoS	*Quality of Service*
RBD	*Reliability block diagram*
RDBMS	*Relational Database Management System*
SAN	*Storage Area Network*
SD2DS	*Scalable Distributed Two-layered Data Structures*
SGBD	Sistema Gerenciador de Banco de Dados
SLA	Service Level Agreement
SPN	*Stochastic Petri net*

SSD	*Solid State Drive*
TPN	*Timed Petri net*
VM	*Virtual Machine*

LISTA DE SÍMBOLOS

γ	Letra grega Gama
$\in$	Pertence
δ	Delta
θ	Teta
σ	Sigma
μ	Mi

SUMÁRIO

1 INTRODUÇÃO 18
1.1 CONTEXTUALIZAÇÃO 19
1.2 OBJETIVOS 21
1.3 CONTRIBUIÇÕES 23
1.4 ESTRUTURA DO DOCUMENTO 23
2 REFERENCIAL TEÓRICO 25
2.1 SGBD NOSQL 25
2.2 CONSISTÊNCIA EVENTUAL 28
2.3 CONSISTÊNCIA BASEADA EM QUORUM 29
2.4 CONFIABILIDADE E DISPONIBILIDADE 31
2.5 AVALIAÇÃO DE DESEMPENHO 33
2.6 ANÁLISE DE SENSIBILIDADE 35
2.7 INJEÇÃO DE FALHAS 38
2.8 REDE DE PETRI 40
2.8.1 Propriedades das Redes de Petri 42
2.8.2 Redes de Petri Determinísticas e Estocásticas 43
2.9 DIAGRAMA DE BLOCO DE CONFIABILIDADE 46
2.10 FUNÇÕES ESTRUTURAIS 47
2.10.1 Componentes em série 47
2.10.2 Componentes em paralelo 48
2.11 INTERVALO DE CONFIANÇA 49
2.11.1 Intervalo de Confiança para o Desempenho e a Consistência 49
2.11.2 Método de KEESE e o Intervalo de Confiança para a Disponibilidade 52
2.12 CONSIDERAÇÕES FINAIS 53
3 TRABALHOS RELACIONADOS 54
3.1 MODELAGEM, DESEMPENHO E DISPONIBILIDADE 54
3.2 MODELAGEM E AVALIAÇÃO DA CONSISTÊNCIA NOSQL 56
3.3 CONSIDERAÇÕES FINAIS 61
4 MÉTODO PROPOSTO 62
4.1 DEFINIÇÃO DO PROBLEMA 65

4.1.1 Entender o sistema . . . 65
4.1.2 Conceber o ambiente . . . 65
4.1.3 Definir parâmetros e métricas . . . 66
4.2 MODELAGEM . . . 66
4.2.1 Coleta de dados . . . 66
4.2.2 Modelar sistema . . . 67
4.2.3 Validar modelos . . . 68
4.3 ANÁLISE DOS RESULTADOS . . . 68
4.3.1 Realizar experimentos . . . 68
4.3.2 Analisar resultados . . . 69
4.4 CONSIDERAÇÕES FINAIS . . . 69
5 MODELOS . . . 71
5.1 MODELO DE DESEMPENHO E DISPONIBILIDADE . . . 72
5.2 MODELO DE ACESSO AO DADO . . . 75
5.3 MODELO RBD . . . 77
5.4 MODELO DE CONSUMO DE ENERGIA . . . 79
5.5 CONSIDERAÇÕES FINAIS . . . 79
6 RESULTADOS EXPERIMENTAIS . . . 81
6.1 VALIDAÇÃO DOS MODELOS PROPOSTOS . . . 81
6.1.1 Validação do desempenho . . . 83
6.1.2 Validação da disponibilidade . . . 85
6.1.3 Validação do modelo de probabilidade da consistência . . . 87
6.2 EXPERIMENTOS . . . 88
6.2.1 Desempenho . . . 88
6.2.2 Disponibilidade . . . 91
6.2.3 Probabilidade de acesso ao dado mais recente . . . 94
6.2.4 Consumo de energia . . . 97
6.3 ESTUDOS DE CASO . . . 100
6.3.1 Estudo de caso 1 . . . 100
6.3.2 Estudo de caso 2 . . . 102
6.4 CONSIDERAÇÕES FINAIS . . . 104
7 CONCLUSÃO E COMENTÁRIOS FINAIS . . . 105
7.1 CONTRIBUIÇÕES . . . 106

7.2 LIMITAÇÕES 108
7.3 TRABALHOS FUTUROS 108
REFERÊNCIAS 110
ANEXO A – EXEMPLO DE EXECUÇÃO DO MODELO DE DESEMPENHO E DISPONIBILIDADE 117
ANEXO B – EXEMPLO DE EXECUÇÃO DO MODELO DE ACESSO AO DADO 133

1 INTRODUÇÃO

> *"A primeira regra de qualquer tecnologia utilizada nos negócios é que a automação aplicada a uma operação eficiente aumentará a eficiência. A segunda é que a automação aplicada a uma operação ineficiente aumentará a ineficiência."*
>
> — Bill Gates

Nas últimas décadas, houve um crescimento exponencial na geração e armazenamento de dados computacionais nas organizações. A evolução da *Internet*, aplicativos, redes sociais, *e-commerce* e internet das coisas são exemplos que motivaram esse crescimento de dados estruturados e não estruturados em todo o mundo. Além disso, esses dados têm como característica a demanda de alto desempenho nos seus acessos (DAVOUDIAN; CHEN; LIU, 2018).

Nesse sentido, é fundamental que o sistema gerenciador de banco de dados SGBD consiga atender às demandas do negócio de maneira eficaz. O SGBD não relacional (NoSQL), que dispensa maiores controles de consistência, foi introduzido com a finalidade de fornecer uma opção às necessidades que os bancos de dados relacionais não podiam satisfazer devido ao maior controle de transações dos dados desses bancos. NoSQL fornece uma abordagem mais eficiente para lidar com várias categorias de dados e suas possíveis alterações. Por exemplo, o SGBD relacional, no decorrer do tempo, apresenta restrições para alteração do esquema de dados, e suas modificações não podem lidar com diferentes estruturas de dados. O NoSQL não tem essa restrição e fornece, ainda, uma maior flexibilidade, incluindo facilidade para replicar os dados, adicionar e remover novos servidores (SAHATQIJA et al., 2018).

Na manipulação de grande volume de dados (*Big Data*), o NoSQL apresenta melhor desempenho e disponibilidade do que bancos de dados relacionais. No entanto, também há desvantagens, como o não controle de transações. Essas características devem ser analisadas para os requisitos do sistema concebido. Com o objetivo de melhorar o desempenho e a disponibilidade do sistema de banco de dados, o NoSQL fornece a consistência eventual; isso significa que é considerada uma inconsistência em um intervalo de tempo entre as réplicas do banco de dados, mas eventualmente esses servidores serão atualizados com a mesma versão do dado (GOMES et al., 2017). Essas características vêm do conceito definido no acrônimo *Basically Available, Soft state, Eventually consistent* (BASE), caracterizado por disponibilizar o sistema, mesmo que exista falha em algum servidor do *cluster* do SGBD. Essa concepção faz

uma contradição ao acrônimo Atomicidade, Consistência, Isolamento e Durabilidade (ACID), utilizado pelos bancos de dados relacionais (PRITCHETT, 2008; MALHOTRA et al., 2017).

A consistência eventual permite que o sistema forneça o serviço mesmo quando as informações ainda não foram replicadas em todos os servidores. Para possibilitar a escolha de diferentes níveis de consistência, a consistência baseada em quorum permite a escolha de quantas réplicas devem confirmar uma operação. Quorum é o número necessário de servidores que confirmam uma operação para que ela seja concluída (GIFFORD, 1979). Portais de notícias, são exemplos de sistemas que podem utilizar menores níveis de consistência, pois possuem demanda de alto desempenho e podem flexibilizar a consistência do acesso à informação. Por outro lado, *e-commerce* tem a necessidade de apresentar uma correta exibição de preços e estoques, esse tipo de sistema é um exemplo que tende a ser configurado com maiores níveis de consistência.

Este trabalho apresenta uma abordagem baseada em *Stochastic Petri net* (SPN) e *Reliability block diagram* (RBD) para estimar desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia do SGBD NoSQL baseado em quorum. São desenvolvidos modelos que permitem a mensuração de vários cenários, o que contribui no projeto e arquitetura de SGBDs NoSQL baseado em quorum. O modelo de dados do SGBD NoSQL não é considerado neste trabalho.

Os modelos são validados em experimentos que utilizam representativos SGBDs NoSQL (IT, 2022), que possuem boa documentação técnica de suporte e fornecem a consistência eventual em diferentes configurações. Os resultados experimentais dos modelos demonstram que o nível de consistência é um fator proeminente que não deve ser negligenciado no projeto de um sistema, pois influencia além da probabilidade de acesso ao dado mais recente, o desempenho, a disponibilidade e o consumo de energia do NoSQL baseado em quorum.

1.1 CONTEXTUALIZAÇÃO

Para atender aos requisitos de sistemas, diversos SGBDs NoSQL fornecem configurações de níveis de consistência ajustáveis, baseadas em quorum (NOUDOUST; ADABI; REZAEE, 2022; WAHID; KASHYAP, 2019; BARON et al., 2016; KALID et al., 2017). Esses ajustes podem afetar o desempenho, a disponibilidade e a probabilidade de acesso aos dados mais recentes e o consumo de energia desses sistemas. Atualmente quorum é uma abordagem comumente utilizada em sistemas de armazenamento de dados propensos a falhas, adotado por importantes SGBDs

NoSQL, como o Amazon DynamoDB, Cassandra e Riak (NOUDOUST; ADABI; REZAEE, 2022; SOBOL, 2021; GAO et al., 2021).

A opção por uma consistência mais flexível tem um impacto significativo no *design* do sistema. Diversos aplicativos têm pré-requisito de baixa latência, ou seja, um pequeno tempo de espera para operações bem-sucedidas. Por exemplo, a Amazon registrou queda de 1% nas vendas com a adição de $100ms$ na latência do sistema; da mesma forma, $500ms$ de latência adicional na pesquisa do Google resultou em uma diminuição correspondente a 20% no seu tráfego (BAILIS et al., 2014). No mercado financeiro, a redução de alguns milissegundos em atraso comercial pode aumentar os ganhos em uma empresa na bolsa de valores em cerca de 100 milhões por ano, além de oferecer grande vantagem competitiva (TIAN et al., 2015).

Reduzir a latência através de menor nível de consistência em um ambiente replicado pode resultar em maior possibilidade de acesso a informações desatualizadas. A confirmação de operações por menos servidores para cada solicitação geralmente diminui a probabilidade de acesso ao dado mais recente no sistema. No entanto, solicitar que mais servidores confirmem a operação diminui o desempenho, pois há uma tendência de maior espera para a conclusão da operação nesses servidores. Adicionalmente a disponibilidade do sistema também é reduzida, visto que menos servidores podem falhar para atendimento do nível de consistência solicitado.

Como a consistência eventual não garante imediatamente que os dados sejam atualizados em todos os servidores, os desafios que surgem na avaliação da consistência são os impactos reais no serviço prestado, ou seja, quantificar a probabilidade de acesso ao dado mais recente, o desempenho e a disponibilidade do sistema, conforme a sua configuração. Essas métricas permitem que o arquiteto do sistema obtenha subsídios para a tomada de decisão sobre a estrutura desse projeto e apresente um importante indicador das configurações do sistema de banco de dados para a solução proposta. Aspectos probabilísticos de consistência são observados em alguns estudos (YAO; WANG, 2020; SINGLA et al., 2018; BAILIS et al., 2014; BAILIS et al., 2012).

Obter uma eficiência no provisionamento de recursos computacionais é um desafio. Para não degradar o nível do serviço com o usuário, cerca de 30% dos sistemas em nuvem provisionam seus recursos entre 10% e 15% além da sua utilização (MORO et al., 2019; ZAKARYA; GILLAM, 2017). Nesse contexto, os modelos de desempenho e disponibilidade (MACIEL et al., 2011) são muito importantes, pois permitem a avaliação de diferentes arquiteturas antes da implementação real do sistema. O desempenho e a disponibilidade podem ser avaliados simultaneamente, considerando a consistência no sistema NoSQL. Ainda através de modelos,

os componentes físicos e lógicos do sistema podem ser detalhados para avaliação do impacto dessas escolhas nessa disponibilidade. Além disso, um modelo pode fornecer a probabilidade do acesso ao dado mais recente. A maioria dos trabalhos ignora a avaliação conjunta de disponibilidade e desempenho em sistemas NoSQL baseados em quorum. Este trabalho realiza essa análise e fornece uma avaliação do desempenho, disponibilidade e probabilidade de acesso ao dado mais recente desses sistemas de banco de dados NoSQL.

Outra questão importante para o projeto do sistema, que também é considerada nesta tese, é o consumo de energia. Motivado pelo grande volume de dados, o consumo de energia em *data centers* alcançou uma proporção de 3% de toda a energia consumida no mundo (YUAN et al., 2021). Estima-se que esses *data centers* utilizem cinco vezes mais energia em 2030 (quase 974 TWh) do que a energia utilizada em 2016 (200 TWh). Para os sistemas de banco de dados, *analytics* e internet das coisas, estimou-se uma taxa de crescimento anual de 21,4% (KOOT; WIJNHOVEN, 2021; ANDRAE, 2019). Obter economia no consumo de energia com bons níveis de desempenho e disponibilidade dos serviços em *data centers* é um grande desafio.

Esforços em relação à eficiência energética ainda são necessários para limitar o crescimento da demanda de energia e os consequentes impactos socioambientais na próxima década (IEA, 2020). Portanto, mecanismos que forneçam métricas de consumo de energia são fundamentais para a mitigação dos problemas inerentes à alta demanda de energia elétrica, com estratégias de redução desse consumo.

1.2 OBJETIVOS

Esta tese de doutorado propõe o desenvolvimento de um conjunto de modelos para fornecer a relação entre desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia em ambiente NoSQL baseado em quorum. Esses modelos permitem quantificar essas métricas para a escolha da configuração mais adequada ao sistema. Diferentes níveis de consistência e configurações do sistema são considerados. A abordagem adotada utiliza modelos desenvolvidos em SPN e RBD validados em experimentos para fornecer métricas relacionadas.

O primeiro modelo SPN projetado permite avaliar operações de leitura ou gravação, considerando também falhas dos servidores redundantes do banco de dados na arquitetura física (servidores de suporte) e lógica (máquinas virtuais e SGBD). O segundo modelo SPN, fornece

a probabilidade de acesso ao dado mais recente em função da consistência e do número de servidores. Por meio do modelo RBD, permite-se uma avaliação dos componentes da infraestrutura de provimento do SGBD NoSQL baseado em quorum.

Os resultados experimentais demonstram o impacto da consistência no desempenho e na disponibilidade do sistema. Da mesma forma, a probabilidade de acesso ao dado mais recente é avaliada. Por fim, provemos uma equação para estimar o consumo de energia elétrica conforme o nível de consistência utilizado.

Mais especificamente, os objetivos desta tese são:

- **Modelar o desempenho e disponibilidade do SGBD NoSQL baseado em quorum**. É proposto um modelo formal em SPN para avaliar o desempenho e disponibilidade do NoSQL baseado em quorum, em diferentes níveis de consistência, considerando operações de leitura ou escrita;

- **Modelar a disponibilidade do SGBD NoSQL baseado em quorum, considerando os componentes de infraestrutura do sistema**. É proposto um modelo formal baseado em RBD para avaliar a disponibilidade do NoSQL baseado em quorum, em diferentes níveis de consistência, considerando cada componente necessário a infraestrutura do sistema, a redundância de *hardware* e *software*;

- **Modelar a probabilidade de acesso ao dado mais recente nos SGBDs NoSQL**. É desenvolvido um modelo de desempenho baseado em SPN para mensurar a probabilidade de acesso à última versão do dado no cluster NoSQL baseado em quorum, considerando a frequência de atualizações desse dado;

- **Verificar o impacto dos fatores utilizados nos modelos**. Através de *Design of Experiments* (DoE) (MONTGOMERY, 2017), as métricas de desempenho, disponibilidade e probabilidade de acesso ao dado mais recente e de consumo de energia são comparadas considerando diversas variações nos níveis avaliados.

- **Apresentar resultados experimentais para demonstrar a viabilidade prática do método proposto**. São realizados experimentos em diferentes configurações; a apresentação dos resultados desses experimentos demonstra a efetividade do método proposto.

1.3 CONTRIBUIÇÕES

Como resultado do trabalho desta tese, podem-se destacar as seguintes contribuições:

- Avaliação da qualidade dos serviços oferecidos pelo SGBD NoSQL baseado em quorum para diferentes níveis de consistência considerando desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e o consumo de energia nos SGBDs NoSQL baseados em quorum;
- Proposição de um modelo de desempenho e disponibilidade validado em ambiente real para estimar essas métricas nos sistemas de banco de dados NoSQL baseado em quorum;
- Proposição de um modelo específico de disponibilidade validado em ambiente real que considera a redundância física e lógica do NoSQL baseado em quorum;
- Proposição de um modelo de probabilidade de acesso ao dado mais recente nos SGBDs NoSQL baseados em quorum, considerando diferentes níveis de consistência;
- Avaliação da influência dos fatores considerados no modelo para as métricas de desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia.

A adoção das técnicas de modelagem e suas métricas, propostas neste trabalho, permitem que o projetista do sistema possa avaliar as configurações e requisitos não funcionais do sistema NoSQL baseado em quorum. Dessa forma, é possível fornecer Service Level Agreement (SLA) para o desempenho e a disponibilidade, quantificar a consistência das informações providas pelo SGBD e estimar o consumo de energia do sistema.

Esses benefícios podem reduzir significativamente o custo em recursos desnecessários para o sistema, por exemplo, podem diminuir o número de servidores físicos necessários para atingir a melhor relação entre a disponibilidade e a consistência desejadas.

1.4 ESTRUTURA DO DOCUMENTO

O restante da presente tese de doutorado é organizada da seguinte forma: o Capítulo 2 introduz os conceitos básicos necessários para melhor entendimento do trabalho, por exemplo, SGBD NoSQL, consistência eventual e consistência baseada em quorum. O Capítulo 3 elenca

os trabalhos relacionados à proposição desta tese e suas diferenças. O capítulo 4 demonstra a metodologia de modelagem e avaliação utilizada nesta tese. Os modelos propostos são apresentados no Capítulo 5. Os resultados experimentais analisados através do DoE são apresentados no Capítulo 6. Por fim, apresentamos a conclusão desta tese, com as contribuições, limitações e trabalhos futuros no Capítulo 7.

2 REFERENCIAL TEÓRICO

> *"Só porque alguma coisa não faz o que você planejou que ela fizesse, não quer dizer que ela seja inútil."*
>
> — Thomas Edison

Este capítulo apresenta os conceitos fundamentais para o melhor entendimento desta proposta. Primeiramente, são apresentados os conceitos de SGBD NoSQL, da consistência eventual e da consistência baseada em quorum. Posteriormente, são abordados os conceitos de confiabilidade e disponibilidade. Então, introduz-se a avaliação de desempenho e, em seguida, é apresentada a abordagem de análise de sensibilidade utilizada nesta tese, além dos princípios da injeção de falhas.

A introdução a redes de Petri e redes de Petri estocásticas demonstra as principais características dessas técnicas para os modelos concebidos nesta tese. Em seguida, com o mesmo objetivo, são explicados os princípios do RBD. Finalmente, são descritas as técnicas de obtenção do intervalo de confiança e apresentadas as considerações finais deste capítulo.

2.1 SGBD NOSQL

Nos últimos anos, os avanços em tecnologias de serviços em nuvem, redes sociais, dispositivos móveis e *Internet* resultaram na explosão repentina de dados estruturados e não estruturados, gerados por aplicativos de escopo global. Tais aplicativos têm uma grande variedade de requisitos do sistema de banco de dados que são melhor atendidos por sistemas com capacidade de adicionar mais servidores para responder à demanda dos clientes e para se adaptarem linearmente à grande volume de dados e crescente taxa de processamento de consultas. Assim, priorizam o desempenho, a disponibilidade e a escalabilidade para responder às demandas do sistema (PADHY; KUMARAN, 2019).

Atingir esses requisitos por meio dos sistemas tradicionais de bancos de dados relacionais *Relational Database Management System* (RDBMS) pode não ser a melhor opção. O esquema relacional predefinido eleva consideravelmente a dificuldade de os bancos de dados relacionais realizarem tarefas, como migração de dados (YOO; LEE; JEON, 2018), dificultando, por exemplo, a satisfação de requisitos para abordagens de desenvolvimento de aplicativos que utilizam

métodos ágeis e iterativos exigidos por alguns cenários de processamento, característicos do *Big Data*.

A expansão desses sistemas exige que eles sejam movidos para servidores com *hardware* mais aprimorado, o que encarece o processo e incorre em uma indisponibilidade significativa do sistema durante cada movimento (replicação de dados). A adição de mais servidores para esses sistemas, mesmo em um único ambiente, aumenta a complexidade e a sobrecarga da junção de dados normalizados distribuídos; além disso, esses sistemas encontram problemas de disponibilidade e desempenho devido ao uso de transações distribuídas (DAVOUDIAN; CHEN; LIU, 2018).

Os requisitos que não podem ser totalmente atendidos necessitam de escolhas com base na prioridade da demanda. Isso contribuiu para a nova geração de tecnologias de banco de dados e para uma nova tendência emergente de armazenamentos de dados não relacionais, chamada NoSQL, que é um acrônimo para "*Not Only SQL*". Tais bancos de dados visam satisfazer à alta disponibilidade e aos requisitos de escalabilidade de aplicativos de escopo global (DAVOUDIAN; CHEN; LIU, 2018).

A utilização do NoSQL atende à demanda de três fatores: volume, variedade e velocidade (BOUAZIZ; NABLI; GARGOURI, 2020; KUMAR; Thakur, 2017).

- **Volume** é a quantidade de dados produzidos por qualquer empresa ou organização. Por exemplo, em 2019, o Google processou 100 Petabytes de dados em um dia e o Facebook processou 500 Terabytes de dados por dia (DAS et al., 2019);

- **Variedade** refere-se às diferentes categorias de dados coletados. Podem ser estruturados semiestruturados ou não estruturados. Isso significa que não há limitação quanto à forma do dado a ser armazenado;

- **Velocidade** está relacionada à taxa de geração e transmissão de dados. O NoSQL prioriza o desempenho das operações.

Alguns trabalhos associam mais dois fatores, variedade e valor, que reforçam a importância de informações consistentes (PRAJAPATI; PATEL, 2021; KHAN et al., 2021).

- **Veracidade** envolve as inconsistências e incertezas do conhecimento, se a informação é confiável diversas vezes ou não. Refere-se à anormalidade e ruído nos dados;

- **Valor** é muito importante quando os modelos de negócios são considerados. Refere-se à informação que tendemos a armazenar e processar, e a como usufruir desta enorme quantidade de conhecimento.

Para sistemas distribuídos susceptíveis a falhas, foi desenvolvido o Teorema CAP (BREWER, 2012), segundo o qual não é possível que um sistema distribuído de dados forneça simultaneamente mais de duas das três seguintes garantias: ***C****onsistency* (Consistência), ***A****vailability* (Disponibilidade) e ***P****artition Tolerance* (Partição tolerante a falhas). Portanto, cada sistema pode ser:

- **Consistente e Disponível – CA.** Nesse caso o sistema não tolera falhas, o que é uma garantia difícil de se obter;
- **Consistente e Tolerante a falhas – CP.** Esse sistema não garante a disponibilidade;
- **Disponível e Tolerante a falhas – AP.** A consistência não pode ser garantida.

NoSQL prioriza a disponibilidade e o desempenho, pois a consistência fica em segundo plano (BREWER, 2012). Os SGBDs que aderem às características de atomicidade, consistência, isolamento e durabilidade ACID fazem o oposto: priorizam a consistência, e o desempenho fica em segundo plano.

Algumas características de NoSQL são explícitas, tais como livre de esquema, suporte a fácil replicação, *Application Programming Interface* (API) simples e consistência eventual (não ACID) (ARCHIVE, 2022). A tecnologia NoSQL tem sido notícia nos últimos anos, muitos líderes da *web* 2.0 adotaram NoSQL. Empresas como Facebook[1], Twitter[2], Digg[3], Amazon[4], LinkedIn[5] e Google[6] utilizam esta tecnologia, de diferentes formas (BOBBA et al., 2018).

Diferente do modelo relacional, fundamentado na lógica matemática e na teoria dos conjuntos, e que possui a organização de seus dados em esquemas com estruturas tabulares rígidas (NAVATHE; ELMASRI, 2010), os sistemas NoSQL não seguem essas regras. Eles costumam apresentar flexibilidade em sua estrutura de dados, comumente são sistemas que se apresentam em código aberto, escaláveis e distribuídos (ARCHIVE, 2022). Estas características tornam os

[1] https://www.facebook.com/
[2] https://twitter.com
[3] https://digg.com
[4] https://www.amazon.com
[5] https://br.linkedin.com
[6] https://www.google.com

armazenamentos de dados NoSQL especialmente adequados para utilização como sistemas de gestão de dados de nuvem.

De forma geral, para os SGBDs NoSQL, a ausência de um esquema de dados definido (*schemaless*) permite maior flexibilidade. Outras características apoiam o gerenciamento de grandes dados, tais como: não partilharem a mesma arquitetura, ou seja, em vez de utilizarem um pool de armazenamento comum *Storage Area Network* (SAN), utilizam apenas o armazenamento local, permitindo que a velocidade de acesso seja apenas a de acesso ao disco rígido; sua elasticidade é uma importante característica, isto é, quando um novo servidor é adicionado, um subconjunto de dados deve ser replicado nesses novos servidores. No entanto, os dados podem não ser instantaneamente replicados e também podem ser perdidos em certas janelas (CHANDRA, 2015).

Os SGBDs NoSQL geralmente seguem a propriedade de consistência conhecida como BASE (*basically available, soft state, eventually consistent*). Ao contrário da propriedade ACID, que assegura um controle de integridade, BASE prioriza o desempenho – a consistência fica em segundo plano – e BASE aceita que a consistência esteja em um estado de fluxo, ou seja, há tolerância de inconsistências temporárias para se obter maior disponibilidade (PRITCHETT, 2008).

A HG Insights relacionou 184.000 empresas que utilizam ao menos um SGBD NoSQL. Destas empresas, 26,8% utilizam SGBD NoSQL baseado em quorum, sendo, respectivamente, os exemplos mais utilizados: Cassandra[7], DynamoDB[8] e Riak[9], com atuação nas áreas financeira, entretenimento, comunicação e manufatura. Essas empresas têm receita anual acima de 1 bilhão de dólares americanos (INSIGHTS, 2021).

2.2 CONSISTÊNCIA EVENTUAL

A consistência no SGBD NoSQL pode ser definida como um único estado válido em todas as instâncias do banco de dados, no sentido de que todos os servidores de banco de dados redundantes possuem a mesma versão do dado (MOHAMED; ALTRAFI; ISMAIL, 2014; PAZ, 2018). A consistência é uma característica importante em um sistema de informação. Seu papel é evidenciado em sistemas distribuídos, e sua falta pode levar a problemas graves, por exemplo,

[7] https://discovery.hgdata.com/product/apache-cassandra
[8] https://discovery.hgdata.com/product/amazon-dynamodb
[9] https://discovery.hgdata.com/product/riak

perda de dados. No entanto, preservar a consistência nas operações do sistema pode ser uma tarefa muito difícil (KRECHOWICZ; DENIZIAK; ŁUKAWSKI, 2021).

O desempenho do sistema é afetado durante a busca de maior consistência, pois é necessário verificar se uma operação de gravação foi executada em cada servidor solicitado; além disso, para leituras, um SGBD deve garantir que os dados retornados estejam atualizados. Da mesma forma, a disponibilidade também é influenciada pela política de consistência, porque em um SGBD tolerante a falhas, para elevar a garantia de consistência, é necessária a confirmação de mais servidores redundantes. Com a consistência eventual a disponibilidade é aprimorada, pois, além de haver tolerância a falhas, nem todos os servidores precisam ser atualizados de imediato.

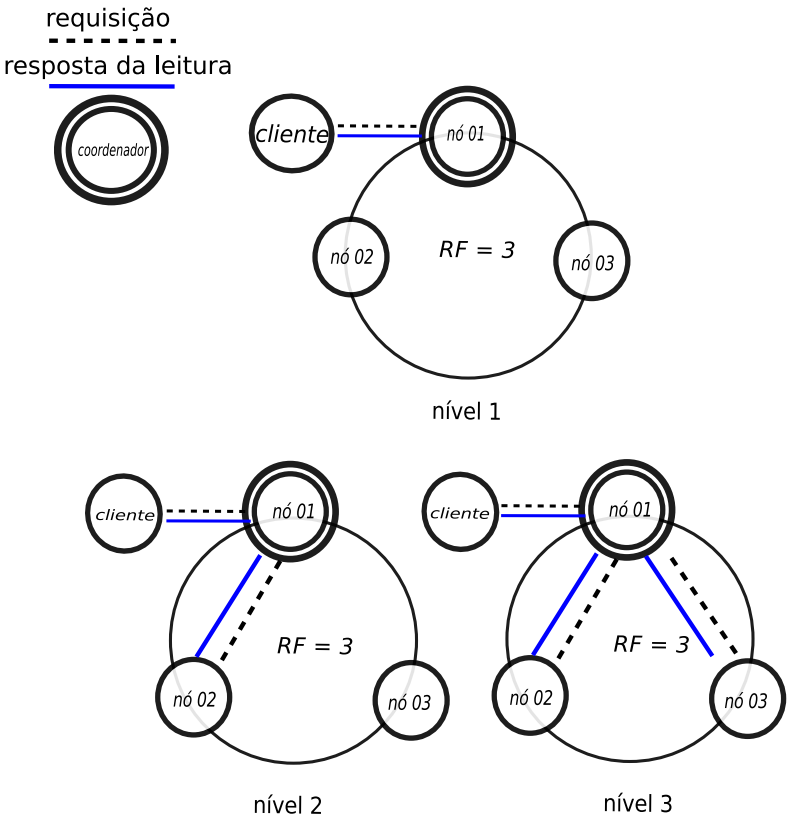
Para maior desempenho e disponibilidade, SGBDs NoSQL adotam consistência eventual, de modo que uma inconsistência temporária seja permitida (ou seja, nem todos os servidores redundantes têm imediatamente os dados mais recentes) e uma réplica pode fazer retornar seus dados acessíveis (que podem não ser os mais recentes). Eventualmente, todos os servidores redundantes serão consistentes (HAHN; CHEREJA; MATEI, 2021).

2.3 CONSISTÊNCIA BASEADA EM QUORUM

Para mitigar essa janela de inconsistência e os diferentes requisitos dos sistemas, representativos SGBDs NoSQL permitem a adoção de diferentes níveis de consistência baseados em quorum, por exemplo, Cassandra, Riak, DynamoDB (GAO et al., 2021; KUMAR et al., 2018). Quorum é o número necessário de servidores que confirmam uma operação para que ela seja concluída. Mais especificamente, o SGBD confirma a execução de uma operação (por exemplo, operação de leitura) se e somente se um número suficiente de servidores (quorum) reconhecer a conclusão bem-sucedida da operação (GIFFORD, 1979; YAO; WANG, 2020). O nível de consistência pode ser especificado explicitamente (exemplo: um, dois, três), é ajustado de acordo com os requisitos do aplicativo e limitado ao número de réplicas do SGBD (YAO; WANG, 2020).

A Figura 1 apresenta um SGBD composto por três servidores redundantes e três níveis diferentes de consistência. Um servidor responsável por receber uma solicitação de operação pode ser também responsável por coordenar a comunicação com outros servidores redundantes, que dependem do nível de consistência. Este servidor é chamado *coordenador* e qualquer um destes servidores pode assumir esta função (que depende do balanceamento de carga do sistema) (HUANG et al., 2017b).

Figura 1 – Exemplo de níveis de consistência (um, dois e três)



Fonte: Elaboração Própria.

Assumindo o nível de consistência um, a confirmação de uma operação (por exemplo, gravação) requer a confirmação de apenas um único servidor. Para o nível $dois$, dois *nós* devem confirmar a operação e, no caso de leitura, os dados são comparados para que o dado mais recente seja considerado a resposta da solicitação. Para esse exemplo, o nível $dois$ com três servidores redundantes coincide com a política $QUORUM$, na qual a consistência dessa escrita é fornecida pelo próximo número inteiro maior que a metade dos *nós* do banco de dados, isto é, mais de 50% das réplicas (KUMAR et al., 2018). Da mesma forma, o nível *três* requer a confirmação de todos os nós; neste exemplo, é o mesmo que a política ALL dado que todos os nós estão sendo considerados na operação. No caso da leitura, é garantido que o dado mais recente armazenado nas réplicas seja sempre acessado (consistência forte) (DIPIETRO, 2020).

Em geral, as variáveis que definem o nível de consistência da operação são: fator de replicação, que contêm dados atualizáveis (RF); número de réplicas, que confirmam a gravação de dados, isto é, quorum de escrita (W); e número de réplicas que comparam a leitura, quorum de leitura (R). Dessa forma, no SGBD NoSQL baseado em quorum, temos a consistência

eventual dada por $(W + R \leq RF)$ e a consistência forte dada por $(W + R > RF)$. Por exemplo, para $W = 1$, $R = 1$ e $RF = 3$, temos um exemplo da consistência eventual, visto que é possível que seja realizada uma leitura do dado em um servidor que ainda não foi atualizado. Para o exemplo $W = 2$, $R = 2$ e $RF = 3$, temos a consistência forte, visto que a leitura sempre acessará ao menos um servidor que recebeu a atualização da escrita (BAILIS et al., 2014; KHELAIFA et al., 2019). A consistência eventual não restringe o comportamento do sistema quando as operações de leitura podem retornar antes da atualização completa dos servidores (GOMES et al., 2017).

A possibilidade de escolher o esforço para obter a consistência forte ou os níveis de consistência eventual é importante, porque permite ao arquiteto do sistema obter o esforço adequado aos requisitos desejados do sistema.

2.4 CONFIABILIDADE E DISPONIBILIDADE

Os principais serviços em nuvem são caracterizados por sua massiva utilização. Garantir os níveis de disponibilidade exigidos por estes sistemas é um grande desafio e comumente são utilizados servidores redundantes para atender a esse requisito (DAVOUDIAN; CHEN; LIU, 2018). Defeitos nesses serviços podem causar degradação no desempenho ou até a sua interrupção, devido à indisponibilidade do recurso solicitado. Essa interrupção pode ser motivada por evento de falha no *hardware*, no *software*, na interconexão de rede ou na infraestrutura.

A repetição de falhas no sistema pode prejudicar a preferência do serviço por parte de seus usuários (SOUSA; LINS, 2018). Ademais, a recuperação do sistema em adventos de falha pode ser muito onerosa, devido à recuperação ou sincronismo desses ambientes.

A seguir, disponibiliza-se a definição formal dos importantes conceitos de confiabilidade e disponibilidade.

- **Confiabilidade** $R(t)$ – é definida como a probabilidade de um sistema realizar sua função predefinida sem falhas para um intervalo de tempo específico (MACIEL; TRIVEDI; KIM, 2010): $R(t) = P\{T \geq t\}$, onde T é a variável aleatória que representa o tempo de falha do sistema (ou de um componente único). Além disso, $R(t) = 1 - F(t)$, de tal forma que $F(t) = P\{T < t\}$ é a função de distribuição cumulativa, a qual representa que uma falha ocorreu antes do tempo T.

- **Disponibilidade** A – é a probabilidade de o sistema estar em uma condição de funciona-

mento. Considera a alternância entre o estado operacional e o de falha (MACIEL; LINS; CUNHA, 1996). A disponibilidade instantânea (A) é comumente adotada, e as equações 2.1 e 2.2 são consideradas, onde A é a disponibilidade, $uptime$ é o tempo em que o sistema esteve disponível e $downtime$ é o tempo em que o sistema esteve indisponível.

$$A = \frac{uptime}{downtime + uptime} \tag{2.1}$$

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

MTTF é o tempo médio para a falha e MTTR é o tempo médio para o reparo, de tal forma que:

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

$$MTTR = \int_0^\infty 1 - M(t)dt \tag{2.4}$$

$M(t)$ é a função de distribuição cumulativa e representa a probabilidade de que um reparo irá ocorrer no tempo t. $R(t)$ é a função de confiabilidade apresentada anteriormente.

É possível classificar a disponibilidade de um sistema por meio do número de nove equivalentes (ÖHMANN; SIMSEK; FETTWEIS, 2014). A Tabela 1 apresenta alguns exemplos da relação entre o *downtime* e o número de noves com suas respectivas categorias (GRAY; SIEWIOREK, 1991).

Tabela 1 – Número de noves equivalentes para disponibilidade

Número de noves	Disponibilidade (%)	Categoria	*downtime*$_{ano}$
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

Para representar os modos de funcionamento e falha dos sistemas, pode-se ainda utilizar funções lógicas por meio do uso de expressões *booleanas* para definir o estado do sistema pelo

estado de seus componentes. Ou seja, conforme o estado individual e arranjo dos componentes do sistema, utilizam-se expressões *booleanas* para inferir o estado do sistema como um todo. Além disso, se um dado sistema consegue prover suas funcionalidades mesmo na presença de falhas, diz-se que esse sistema é tolerante a falhas.

Redundância é uma técnica muito importante para a implementação de sistemas tolerantes a falhas, visando melhorar a disponibilidade e confiabilidade (MACIEL; LINS; CUNHA, 1996). Em geral, essa técnica consiste na adição de componentes extras ao sistema de tal forma que, se um componente falhar, o componente redundante assume o seu lugar. A redundância dinâmica é uma técnica representativa que considera componentes redundantes para substituir os componentes principais sempre que estes estão em estado de falha.

A redundância pode ser fornecida das seguintes formas (ZAKHAROV; KLEPIKOV; PODKHVATILIN, 2020; PATTERSON; GIBSON; KATZ, 1988):

- **sistema** – redundância de componentes e subsistemas individuais;
- **estrutural (arquitetura)** – reconfiguração adaptativa da estrutura do sistema;
- **informação** – relaciona-se à duplicação dos dados ou à adição de informações redundantes, de forma a permitir a verificação da consistência ou correção dos erros detectados;
- **comunicação** – uso de canais de comunicação duplicados e de diferentes tipos;
- ***hardware*** – obtido com a replicação de componentes eletrônicos que assumem as tarefas do sistema, em caso de falha dos hardwares principais.

Para os sistemas avaliados nesta tese, utilizaram-se algumas das opções de redundância acima classificadas, isto é, sistema, informação e hardware.

2.5 AVALIAÇÃO DE DESEMPENHO

O desempenho é um critério importante na concepção, aquisição e uso de sistemas computacionais. Obter o melhor desempenho para determinado custo é um requisito comum em sistemas de informação. Para atingir esse objetivo, os profissionais que projetam os sistemas necessitam, ao menos, conhecer a terminologia de avaliação de desempenho e suas técnicas; e devem ser capazes de indicar as exigências de seus sistemas de desempenho e comparar

diferentes alternativas para encontrar aquela que melhor atenda às suas necessidades (BUKH; JAIN, 1992).

Para a avaliação de desempenho, um conjunto de métricas deve ser escolhido. Uma maneira de preparar esse conjunto é listar os serviços oferecidos pelo sistema. Para cada solicitação de serviço feita ao sistema, há vários resultados possíveis. Por exemplo, um banco de dados oferece o serviço de responder a consultas. Quando executada uma consulta, ele pode responder corretamente, ele pode responder incorretamente ou não elaborar resposta. Se o sistema executa o serviço corretamente, o seu desempenho é medido pelo tempo necessário para realizar o serviço, pela taxa com que ele é executado e pelos recursos consumidos durante a execução desse serviço. A estas três métricas são atribuídos o rótulo de capacidade de resposta (BUKH; JAIN, 1992).

A maioria dos sistemas oferece mais de um serviço, portanto, o número de métricas cresce proporcionalmente. Para muitas métricas, o valor médio é o mais importante. No entanto, o efeito da variabilidade deve ser considerado, pois é um fator que pode degradar significativamente a produtividade.

(LILJA, 2004) apresenta vários objetivos típicos para avaliação de desempenho:

- **comparar alternativas** – na escolha de determinado sistema podem existir várias alternativas. Além disso, em cada sistema podem existir diversas possibilidades de configuração que, de maneira geral, podem afetar tanto o custo quanto o desempenho da solução desejada. Comparar alternativas pode fornecer informações quantitativas a respeito de quais configurações são melhores em condições específicas;

- **determinar o impacto de uma característica** – na concepção ou alteração de sistemas, muitas vezes é necessário conhecer o impacto dessa nova característica adicionada;

- **ajuste do sistema** – neste caso, utiliza-se esse objetivo para encontrar a combinação de parâmetros ideais do sistema a fim de obter seu melhor desempenho na métrica desejada;

- **identificar o desempenho relativo** – o desempenho de um sistema de computador, geralmente, tem significado apenas no contexto da comparação com outro sistema ou configuração de outro sistema. Outro objetivo pode ser quantificar o desempenho em relação a uma expectativa;

- **definir expectativas** – por meio da análise de desempenho, podem ser inferidas as capacidades de uma próxima linha de sistemas de computadores.

O esforço para a execução da análise de desempenho deve considerar o custo da tomada de decisão errada. As técnicas de avaliação de desempenho incluem a medição e a modelagem analítica.

De acordo com (ARAUJO, 2009) e (BUKH; JAIN, 1992), a medição de desempenho consiste essencialmente na monitoração do sistema enquanto ele está sob ação de uma carga de trabalho. Para adquirir resultados representativos, a carga de trabalho utilizada nos estudos de desempenho, que pode ser real ou sintética, deve ser cuidadosamente selecionada.

Embora a carga de trabalho real seja uma boa escolha por representar fielmente o sistema, ocasionalmente, esta opção não é a desejável. Isso acontece quando o tamanho da carga não é considerável, quando esses dados recebem muitas interferências ou, até mesmo, em razão de facilidade de acesso desses dados. Devido a esses motivos, as cargas de trabalho sintéticas têm sido utilizadas com *benchmark* e programas.

A modelagem analítica é outra técnica utilizada para a avaliação de desempenho. Para esses modelos, a representação do sistema é construída por um conjunto de formulações matemáticas que podem ser solucionadas para valores de entrada, do que lhe concerne derivam as métricas de interesse. Modelos analíticos podem representar sistemas complexos com diversas características, incluindo concorrência e paralelismo (RODRIGUES; ENDO; SILVA, 2019) e devem apresentar uma visão em alto nível de abstração do sistema com as características indispensáveis para a sua análise. Os modelos analíticos fornecem resultados exatos, não há necessidade de intervalo de confiança. Essa exatidão se refere às soluções dadas por sistemas de equações. Uma prática comum para a verificação de que esses modelos representam adequadamente os sistemas é a sua validação, comparando-os por inferências estatísticas com medições no sistema em funcionamento.

Esta tese realiza modelagem analítica baseando-se em medições e visa criar modelos com maior grau de confiança. A abordagem por modelos analíticos foi selecionada devido à capacidade da representação das características desejadas na pesquisa e ao fornecimento das métricas de interesse.

2.6 ANÁLISE DE SENSIBILIDADE

O termo análise de sensibilidade pode ser definido como uma investigação sistemática da reação das respostas de simulação para valores extremos da entrada do modelo ou de mudanças drásticas na estrutura do modelo (KLEIJNEN, 1995). (FRANK, 1978) define como um

conjunto de métodos utilizados para identificar componentes de maior influência em determinada métrica.

A análise de sensibilidade ainda não possui uma definição padrão para esse termo. (CRICK; HILL, 1987) realiza uma observação que generaliza a utilização das técnicas utilizadas em parâmetros importantes, aqueles cuja incerteza tem importante contribuição para a indefinição no resultado da avaliação; e parâmetros sensíveis, que são aqueles que têm influência significativa no resultado da avaliação.

Diversas técnicas para a verificação da sensibilidade são descritas na literatura, de forma que resultados ligeiramente diferentes podem ser obtidos. No resultado da análise, o analista elenca uma classificação de sensibilidade com os principais parâmetros de entrada em função da métrica avaliada. O mais importante é a obtenção dos parâmetros que ficam no topo da lista. A divergência da classificação dos parâmetros de menor importância nas diferentes técnicas de análise de sensibilidade não é uma preocupação prática (HAMBY, 1994).

Para especificar claramente o objetivo de uma análise de sensibilidade pode-se, segundo (IOOSS; LEMAÎTRE, 2015):

- identificar e priorizar as entradas mais influentes;
- identificar insumos não influentes para fixá-los em valores nominais;
- mapear o comportamento da métrica em função dos valores de entrada, especificando um domínio de entrada;
- calibrar algumas informações de entrada utilizando informações já disponíveis ou restrições do sistema.

Nesta pesquisa adotamos uma técnica de análise de sensibilidade que é amplamente utilizada e se ajusta à avaliação, alvo deste trabalho. O DoE (do inglês, *Design Of Experiments*) conduz o planejamento de experimentos para prover resultados apropriados para a análise por métodos estatísticos, permitindo conclusões objetivas (MONTGOMERY, 2017).

Os experimentos podem ser definidos como uma ou mais verificações com as quais são aplicadas mudanças nas variáveis de entrada de um sistema, para analisar os resultados das variáveis de saída (MONTGOMERY; RUNGER, 2013). Nessa técnica o termo *fator* define uma variável controlável, que influencia o resultado. Os valores atribuídos aos fatores são conhecidos como *níveis*, e as mudanças nas médias dos resultados são nomeados como *efeitos* (MASON; GUNST; HESS, 2003).

Outro comportamento que deve ser avaliado no DoE é o efeito resultado das interações dos fatores, isto é, que resulta da interação entre os níveis dos fatores. *Design* fatorial é um tipo comum da abordagem DoE, que é eficiente para experimentos que contenham dois ou mais fatores. Quando os fatores são combinados em um *design fatorial*, eles são ditos *cruzados*.

O *efeito* é reconhecido pela mudança na resposta ocasionada pela alteração do nível desse fator. Essa técnica evita conclusão equivocada, quando há interações entre os fatores, permitindo que o efeito do fator seja estimado em vários níveis de outros fatores (MONTGOMERY, 2017).

Quando o efeito se refere aos fatores primários do estudo, é chamado de *efeito principal*. A Tabela 2 apresenta um exemplo de experimento fatorial de dois fatores (A e B) para os níveis (A_{baixo}, A_{alto}, B_{baixo}, B_{alto}). O efeito principal é a diferença entre a média dos níveis mais altos e os níveis mais baixos de cada fator, conforme a Equação 2.5 e a Equação 2.6.

Tabela 2 – Exemplo fatorial

	Fator B	
Fator A	B_{baixo}	B_{alto}
A_{baixo}	x_1	y_1
A_{alto}	x_2	y_2

$$A = \frac{x_1 + y_1}{2} - \frac{x_2 + y_2}{2} \tag{2.5}$$

$$B = \frac{x_1 + x_2}{2} - \frac{y_1 + y_2}{2} \tag{2.6}$$

O fator (A) e o fator (B) podem obter um efeito significativo. No entanto, em algumas avaliações, a diferença entre os níveis de um fator não é a mesma entre os níveis do outro fator. Quando isso ocorre, pode existir uma grande $interação$ entre os fatores. A Equação 2.7 apresenta o efeito da interação.

$$AB = \frac{x_2 + y_1}{2} - \frac{x_1 + y_2}{2} \tag{2.7}$$

O efeito da interação (AB) é a diferença das médias na diagonal. Ou seja, é a metade da diferença entre os efeitos do fator (A) nos dois níveis do fator (B). A observação dos efeitos principais e suas interações podem demonstrar importantes informações de comportamento do sistema.

Nesta tese, para a análise de sensibilidade dos resultados dos experimentos realizados com os modelos, utilizou-se a Classificação dos efeitos. O efeito demonstrou o tamanho e a direção da relação dos fatores com as métricas avaliadas.

2.7 INJEÇÃO DE FALHAS

Um importante problema no desenvolvimento e avaliação de sistemas confiáveis é a validação dessa confiabilidade. Avaliar a ocorrência de falhas em sistemas computacionais pode ser uma tarefa com excessivo dispêndio de tempo. O tempo na espera da ocorrência natural de falhas pode inviabilizar o trabalho a ser realizado. Dessa forma, uma alternativa comum é a utilização de técnicas de injeção de falhas (KOOLI; NATALE, 2014). Essas técnicas ajudam o monitoramento e controle dos experimentos, permitindo a observação do comportamento do sistema avaliado na ocorrência das falhas e o seu reparo (KOOLI; NATALE, 2014; SONG et al., 2011).

(SONG et al., 2011) apresentam uma classificação para técnicas de injeção de falha em três categorias: injeção de falha de *hardware*; injeção de falha de *software*; e injeção de falha baseada em *simulação*. Os pesquisadores afirmam que a injeção de falha de *hardware* e de *software* devem ser injetadas no protótipo ou sistema totalmente operacional, que pode ser coletivamente referido como injeção de falha com base física. Essa abordagem foi realizada neste trabalho.

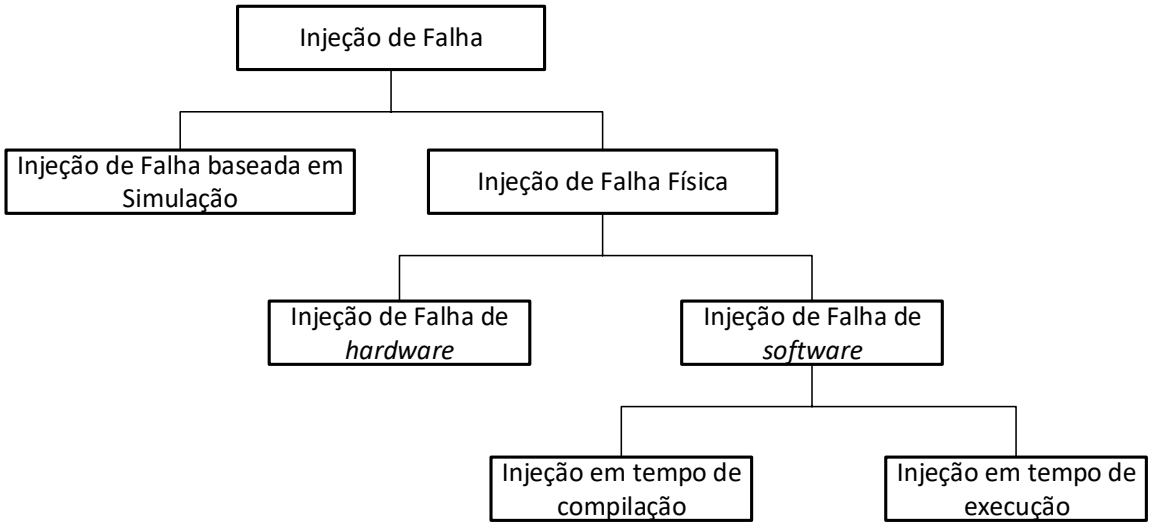
A Figura 2 apresenta a classificação das técnicas de injeção de falhas comumente utilizadas. A seguir, descrevemos esta classificação.

- **Injeção de falha de *hardware*** – sua introdução no sistema de destino é realizada por meio de equipamento adicional. Dependendo das falhas e suas localizações, os métodos de injeção se enquadram em duas categorias:

  injeção de falha de hardware com contato: em que o injetor tem contato físico direto com o sistema alvo, por exemplo, produzindo mudanças de tensão ou corrente para o alvo por meio de um socket;

  injeção de falha de hardware sem contato: em que, sem contato físico direto com o sistema alvo, uma fonte externa produz algum fenômeno, por exemplo, interferência eletromagnética.

Figura 2 – Classificação de técnicas de injeção de falhas



Fonte: (SONG et al., 2011)

- **Injeção de falha de *software*** – altera o conteúdo da memória ou registro para simular a ocorrência de falhas de *hardware* ou de *software*. Podemos categorizar os métodos de injeção de software baseando-nos no momento de injeção das falhas, que pode ocorrer durante o tempo de compilação ou durante o tempo de execução. A injeção de falha em tempo de compilação, injeta erros no código-fonte, ou antes ou durante a compilação do código do programa, para injetar falhas transitórias no hardware ou software. No entanto, a injeção de falha em tempo de execução pode acionar, durante a execução do programa, a injeção de falha. Os mecanismos de disparo comumente usados incluem *time-out*, exceções e inserção de código. Existem diversas ferramentas para injeção de falha de *software*. Ou, ainda, podem ser criados *scripts* para essas injeções de falha.

- **Injeção de falha baseada em simulação** – nesta técnica, um modelo do sistema em teste, que pode ser desenvolvido em diferentes níveis de abstração, é simulado em outro sistema de computador. Durante a simulação, falhas são induzidas pela alteração dos valores lógicos dos elementos do modelo durante a simulação. A abordagem principal dessa técnica consiste em injetar falhas em modelos de alto nível.

Neste trabalho, técnicas de injeção de falhas foram utilizadas na *Virtual Machine* (VM) que contém o serviço avaliado. Uma abordagem similar é encontrada em (LENKA; PADHI; NAYAK, 2018). Ademais, modelos propostos nesse trabalho também consideram falhas.

2.8 REDE DE PETRI

Rede de Petri, *Petri net* (PN) é uma técnica de modelagem formal que pode ser representada graficamente para auxiliar a comunicação visual ao se modelar o comportamento do sistema. A base matemática é representada por equações, equações algébricas e algoritmos no sistema. As redes de Petri são usadas para modelar o fluxo de controle em um sistema e conseguem modelar sistemas não determinísticos, paralelos, concorrentes e assíncronos.

PN é um gráfico bipartido $PN = (P, T, F, W, M_0)$, onde P é um conjunto de *locais*, T é um conjunto de transições e F é uma relação de fluxo e um conjunto de *arcos direcionados*. As transições decodificam ações ou procedimentos do sistema de *software*. Lugares são as condições que definem as dependências causais entre transições; eles podem conter *tokens* que indicam o estado atual do sistema. W é a função de atribuição de peso aos arcos, M_0 é a marcação inicial, ou seja, o estado inicial do modelo.

Os elementos da rede de Petri são detalhados a seguir, com base em (MURATA, 1989):

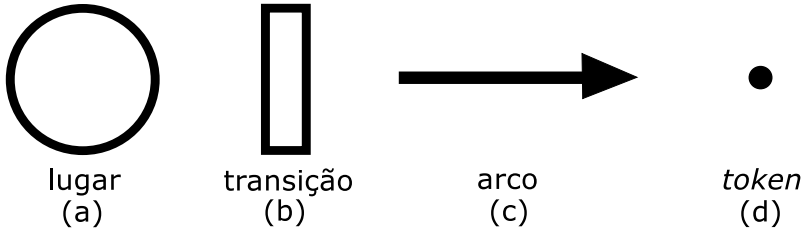
- **lugares** – caracterizam os elementos passivos da rede e têm como principal função o armazenamento de *tokens*; podem conter qualquer número de *tokens* os quais são removidos e adicionados à medida que as transições são disparadas (ver Figura 3 (a));

- **transições** – representam os elementos ativos da rede, ou seja, as ações realizadas pelo sistema. São representadas por traços ou barras (ver Figura 3(b)). A transição é utilizada em um processo chamado disparo. Para que esse processo ocorra, é necessário que cada um dos lugares de entrada contenha, pelo menos, o número de *token* igual ao valor do peso do arco que liga este lugar à transição; então, a transição está habilitada. Quando esse pré-requisito não é atendido, a transição está desabilitada e o disparo de uma transição pode levar a um novo estado de marcação do modelo. Vale salientar que pode haver várias transições habilitadas no modelo, e a seleção de qual transição irá disparar é não determinística;

- **arcos** – representam o fluxo dos *tokens* pela rede. Existe um peso associado a cada arco. Esse peso é um número inteiro positivo. Quando o peso não é especificado nos arcos, assume-se que ele é unitário. As transições e os lugares podem ser conectados por múltiplos arcos. São graficamente representadas por setas (ver Figura 3 (c));

• **tokens** – caracterizam o estado em que o sistema se encontra em determinado momento (ver Figura 3 (d)).

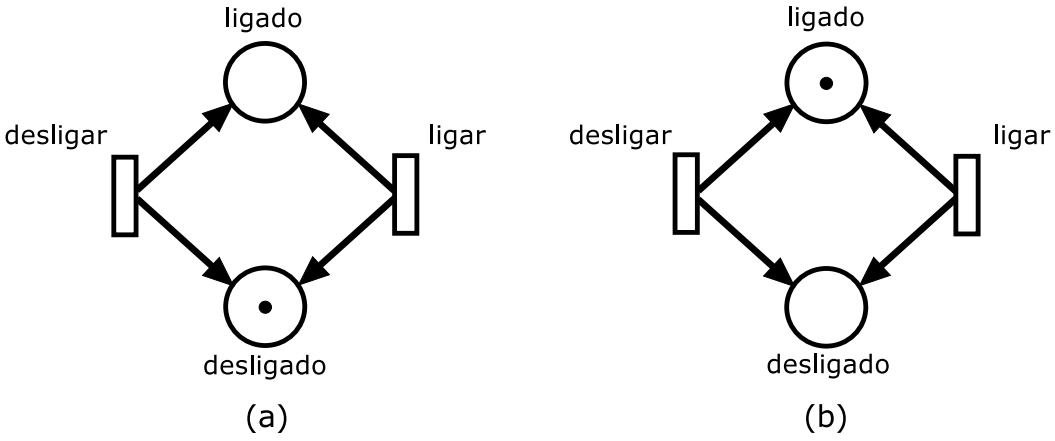
Figura 3 – Elementos de uma rede de Petri



Fonte: Elaboração Própria.

A Figura 4 apresenta o processo de funcionamento de um sistema modelado que utiliza o formalismo das redes de Petri. Os estados possíveis desse sistema são *ligado* e *desligado*. Além disso, observam-se duas ações possíveis neste modelo, quais sejam a de ligar o sistema e desligá-lo. Logo, para modelar esse sistema, são utilizados dois lugares, que representam os possíveis estados, e duas transições, que representam as ações. A marcação inicial deste modelo é com um *token* no lugar desligado (ver Figura 4 (a)). A partir da marcação inicial, a única ação possível é o disparo da transição *ligar*. Uma vez realizado esse disparo, é removido um *token* do lugar desligado e adicionado um *token* no lugar ligado (ver Figura 4 (b)). Quando no estado ligado, o único evento possível é o disparo da transição *desligar*, que leva o modelo novamente ao estado inicial.

Figura 4 – Estados de um sistema



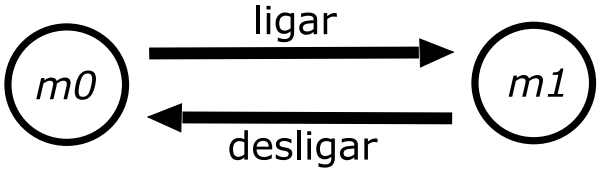
Fonte: Elaboração Própria.

A flexibilidade e alto poder de abstração da rede de Petri permitem a representação de uma infinidade de sistemas. Existe uma série de métodos que possibilitam analisar um grande

número de propriedades de sistemas. As propriedades das redes de Petri podem ser divididas em dois grandes grupos: as comportamentais, que dependem da marcação inicial, e as estruturais, que não dependem. Ademais, ao longo do tempo, diversas extensões de rede de Petri foram desenvolvidas para os mais diversos propósitos (PULIAFITO; TELEK; TRIVEDI, 1997).

O espaço de estados de uma rede de Petri é o seu grafo de alcançabilidade. Trata-se de um grafo rotulado e direcionado utilizado para a representação das marcações que a rede de Petri pode alcançar. Formalmente, um grafo de alcançabilidade pode ser definido como uma tupla (V, E), onde V representa o conjunto dos vértices representados pelas marcações possíveis, e E é o conjunto de arestas rotuladas. Por exemplo, considere o conjunto $M = \{m_0 = \{1, 0\}, m_1 = \{0, 1\}\}$ de marcações alcançáveis da rede de Petri, representada na Figura 4. O respectivo grafo de alcançabilidade da rede de Petri é representado na Figura 5.

Figura 5 – Exemplo de grafo de alcançabilidade



Fonte: Elaboração Própria.

2.8.1 Propriedades das Redes de Petri

Modelos SPN possuem dois tipos de propriedades: comportamentais, que dependem da marcação inicial; e estruturais, independentes dessa marcação (MURATA, 1989). Essas propriedades permitem a verificação de comportamentos do sistema modelado. A seguir, apresentam-se algumas das importantes propriedades comportamentais das redes de Petri (MURATA, 1989).

- **Alcançabilidade (*Reachability*)** – está associada à capacidade de uma rede de Petri atingir uma determinada marcação a partir de uma marcação inicial M_0.
- **Limitação (*Boundedness*)** – a partir da marcação inicial M_0, se qualquer lugar não ultrapassar k *tokens* em qualquer marcação alcançável, a rede é limitada.
- **Vivacidade (*Liveness*)** – é uma propriedade que está ligada à ausência de *deadlock*, ou seja, livre da impossibilidade do disparo de qualquer transição. No entanto, essa ligação

se refere a alguma sequência de transições. Essa propriedade permite a avaliação de marcações que podem nunca ser alcançadas em um modelo.

- **Cobertura (*Coverability*)** – este conceito está diretamente ligado às propriedades de alcançabilidade e vivacidade. A cobertura é uma propriedade que atua em um subconjunto das marcações de uma rede viva e é utilizada para avaliar se uma possível submarcação pode ser coberta.

- **Reversibilidade e *Home state*** – para cada marcação alcançável a partir da marcação inicial M_0, é possível retornar ao estado inicial. Permite-se, ainda, um relaxamento da condição, de modo que, para algumas aplicações, é permitido considerar outra marcação de origem $M^{'}$. Nesse caso, $M^{'}$ é considerado o *home state* se para cada marcação M das marcações alcançáveis por M_0, $M^{'}$ é acessível.

As principais propriedades estruturais para redes de Petri (MURATA, 1989) são:

- **Limitação Estrutural** – se em qualquer lugar o número de *tokens* é limitado para qualquer marcação inicial, a rede é estruturalmente limitada.

- **Conservação** – está relacionada à não destruição de *tokens* na rede de Petri.

- **Repetitividade** – ocorre quando para uma determinada marcação M e uma sequência de transições habilitadas para M, não existe limite de disparo para todas as transições.

- **Consistência** – se a partir da marcação inicial M_0, retorna-se à M_0 com o disparo de todas as transições, a rede é consistente.

Nesta tese buscou-se a avaliação estacionária dos modelos SPN (MARUSSY et al., 2016). Dessa forma, os modelos concebidos atenderam a estes três requisitos fundamentais para a análise estacionária: *Boundedness*, *Home state* e *Liveness*.

2.8.2 Redes de Petri Determinísticas e Estocásticas

Em 1982, Molloy apresentou a rede de Petri estocástica - SPN como uma técnica capaz de especificar sistemas e apresentar uma análise probabilística deles. Surgiu como uma extensão do formalismo de *Timed Petri net* (TPN), que tem como sua principal característica a associação de um atraso fixo para cada transição do modelo. SPN define que todas as suas transições

são temporizadas ($timed$) e que possuem um retardo exponencialmente distribuído (MOLLOY, 1982). As redes de Petri Determinísticas e Estocásticas (DSPNs) são uma extensão das redes de Petri temporizadas, são compostas por transições exponenciais e determinísticas (GERMAN, 2000). Formalmente, as DSPNs podem ser definidas, conforme descreve a Definição 1.

Definição 1. ***(Redes de Petri Determinísticas e Estocásticas)*** *Uma DSPN é uma tupla,* $DSPN = (P, T, I, O, H, \prod, G, M_0, ATTs)$*, onde:*

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

$T = (t_1, t_2, ..., t_m)$ é o conjunto finito de transições imediatas, determinísticas e exponenciais,

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

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

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

$\prod \in \mathbb{N}^m$ é um vetor que associa o nível de prioridade para cada transição,

$G \in (\mathbb{N}^n \to \{true, false\})^m$ é o vetor que associa uma função de guarda para cada transição que pode depender da marcação atual,

$M_0 \in \mathbb{N}^n$ é o vetor que define a marcação inicial para cada lugar (estado inicial),

$ATTs = (Dist, W, Markdep, Policy, Concurrency)^m$ compreende o conjunto de atributos para as transições, onde:

– $Dist \in \mathbb{N}^m \to \mathcal{F}$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, sendo que o domínio de $\mathcal{F}$ é $[0, \infty)$ (a distribuição pode ser dependente da marcação)

– $W \in \mathbb{N}^m \to \mathbb{R}^+$ é a função peso, que associa um peso (w_t) às transições imediatas e uma taxa λ_t às transições temporizadas, onde:

$$W(t) = \begin{cases} w_t \geq 0, & \text{se } t \text{ é uma transição imediata;} \\ \lambda_t > 0, & \text{caso contrário.} \end{cases}$$

– $Markdep \in \{constant, enabdep\}$ define se a distribuição de probabilidade associada ao tempo de uma transição é constante ($constant$) ou depende de uma marcação ($enabdep$),

– $Policy \in \{prd, prs\}$ é a política de preempção adotada pela transição (prd - $preemptive\ repeat\ different$ corresponde à $resampling$ e prs - $preemptive\ resume$ possui significado idêntico à $age\ memory\ policy$),

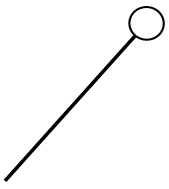
– $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica do servidor único e is significa a semântica do servidor infinito (mesmo sentido dos modelos de fila).

Apesar da Definição 1 mais completa, todas as transições adotadas neste trabalho são exponenciais.

As redes de Petri estocásticas possuem transições que podem ser imediatas ou temporizadas. As transições imediatas possuem tempo de retardo associado igual a zero. As transições temporizadas possuem tempo de atraso exponencialmente distribuído. A propriedade de ausência de memória da distribuição exponencial no atraso dos disparos implica o fato de as SPNs serem isomórficas às cadeias de Markov de tempo contínuo *Continuous-Time Markov Chains* (CTMC), o que significa que o espaço de estados da SPN pode ser convertido para uma CTMC, por meio do grafo de alcançabilidade da SPN. Adicionalmente, as SPNs permitem a adoção de técnicas de simulação como uma alternativa à geração de cadeias de Markov (MACIEL; LINS; CUNHA, 1996).

A seguir, a Figura 6 ilustra os arcos inibidores que são utilizados para impedir o disparo das transições da rede. De maneira semelhante aos arcos tradicionais, também podem ter um peso associado.

Figura 6 – Arco inibidor

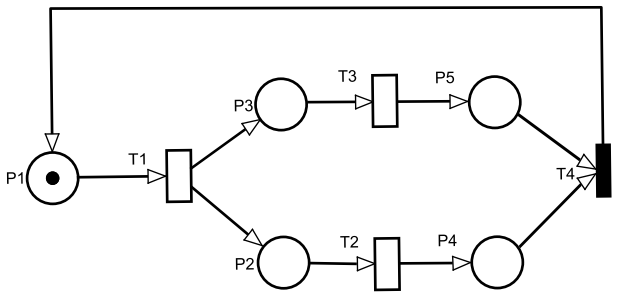


Fonte: Elaboração Própria.

SPN permite associar distribuição exponencial a transições temporizadas (representadas por retângulos brancos) ou atrasos zero a transições imediatas (representadas como retângulos

pretos finos). A figura 7 mostra um exemplo. As transições temporizadas podem ter diferentes graus de simultaneidade: semântica de servidor único (*single server* - ss) ou semântica de servidor infinito (*infinity server*- is). Em geral, is é adotado para representar disparos de transição paralelos, considerando que a taxa de disparo é linearmente aumentada de acordo com o grau de habilitação da transição. Em ss, a taxa de disparo é mantida constante (MARSAN et al., 1998).

Figura 7 – Exemplo de SPN



Fonte: Elaboração Própria.

2.9 DIAGRAMA DE BLOCO DE CONFIABILIDADE

Diagrama de bloco de confiabilidade, do inglês *Reliability Block Diagram* – RBD, é um formalismo para a construção de modelos combinatórios, nos quais a relação estrutural entre os componentes do sistema é utilizada para avaliar a disponibilidade ou confiabilidade do sistema, sendo comumente utilizado em diversos sistemas com módulos independentes, onde cada módulo pode ser representado por um bloco RBD.

O RBD é um modelo sem espaços de estado, o que possibilita uma solução geralmente rápida, pois não requer a geração de espaço de estados durante a sua solução (SATHAYE; RAMANI; TRIVEDI, 2000). Utilizou-se essa abordagem para simplificar a análise da disponibilidade nos SGBDs baseados em quorum.

No formalismo RBD, o sistema se encontra em estado operacional se houver ao menos um caminho ligando os blocos em estado operacional do início ao fim do modelo. Por tratar-se de um modelo combinatorial, é possível a existência de restrições para a representação de sistemas redundantes dinâmicos ou, ainda, de técnicas de manutenção preventiva.

2.10 FUNÇÕES ESTRUTURAIS

Funções estruturais do RBD associam o estado de funcionamento de cada componente de um determinado sistema ao estado de funcionamento do sistema como um todo. Seja x_i variável que representa o estado do componente i para $1 \leq i \leq n$ onde n é o número de componentes do sistema. Então, o estado x_i do componente $_i$ é dado por:

Estado do componente i definido por x_i:

$$x_i = \begin{cases} 1, & \text{componente } i \text{ funcionando} \\ 0, & \text{componente } i \text{ falhou} \end{cases}$$

Para $i = 1, 2, ..., n$. onde n representa o número de componentes.

O vetor $x = (x_1, x_2, ..., x_n)$ representa o estado de todos os componentes. O estado do sistema é determinado pela combinação dos estados dos componentes. A função da estrutura ϕ representa o estado do sistema a partir dos estados dos componentes do vetor x.

Função ϕ Estado do sistema:

$$\phi(x) = \begin{cases} 1, & \text{sistema funcionando} \\ 0, & \text{sistema falhou} \end{cases}$$

O estado do sistema é uma função determinística do estado de todos os componentes, conforme demonstrado na Equação 2.8.

$$\phi = \phi(x) = \phi(x_1, x_2, ..., x_n) \tag{2.8}$$

O esquema visual de um RBD representa os componentes do sistema usando blocos e suas relações lógicas com ligações; esses blocos são conectados em série ou paralelo. Cada bloco contém a taxa de falhas e reparos do componente representado (MACIEL; TRIVEDI; KIM, 2010; VERMA et al., 2010).

2.10.1 Componentes em série

Para as relações em série, quando qualquer componente dessa relação falha, o sistema em sua totalidade não realizará sua tarefa, ou seja, estará indisponível. Dessa forma, o sistema

está em estado operacional se todos os componentes conectados em série estão no estado operacional. A Figura 8 demonstra graficamente esse arranjo.

Figura 8 – Diagrama de bloco de confiabilidade – Arranjo em série

Fonte: Elaboração Própria.

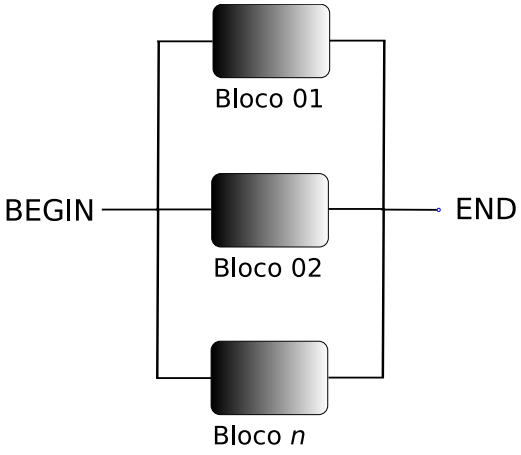
Em um modelo RBD com n componentes em série, a disponibilidade do sistema (A_S) pode ser estimada da seguinte forma: (ver Equação 2.9).

$$A_S = \prod_{i=1}^{n} Ai \tag{2.9}$$

2.10.2 Componentes em paralelo

As relações com arranjo em paralelo entre blocos indicam a redundância entre os componentes ou subsistemas. Para que o sistema esteja em estado operacional, é necessário que apenas um destes componentes esteja operacional (ver Figura 9)

Figura 9 – Diagrama de bloco de confiabilidade – Arranjo paralelo



Fonte: Elaboração Própria.

Em um sistema com n componentes em paralelo, a disponibilidade do sistema (A_S) pode se estimada por meio da Equação 2.10.

$$A_S = 1 - \prod_{i=1}^{n} (1 - Ai) \tag{2.10}$$

Existe ainda outra estrutura nos modelos RBD representada por arranjos de blocos K-out-of-N (K de N). Significa que este arranjo estará operacional enquanto K de N componentes estiverem operacionais. Por exemplo, um sistema possui a necessidade de que $(K = 2)$ elementos estejam funcionando e o arranjo possui $(N = 5)$ elementos totais.

Esse arranjo continuará em funcionamento até que 4 elementos falhem; nessa situação, apenas um componente estará funcionando, o que não é suficiente para que este arranjo esteja disponível.

Os modelos RBD têm restrições para representar relações complexas entre os componentes do sistema. No entanto, se a falha ou recuperação de um componente do sistema não é influenciada por nenhum outro componente do sistema, um modelo RBD pode ser adotado.

2.11 INTERVALO DE CONFIANÇA

O intervalo de confiança é uma estimativa intervalar que contém um parâmetro populacional; é associado à probabilidade de que a estimativa contenha esse parâmetro populacional. O parâmetro é encontrado através da amostra de uma população.

Modelos, quando são implementados de maneira correta, apresentam resultados compatíveis com os sistemas modelados. Para assegurar que os modelos desenvolvidos representem os sistemas aos quais fazem referência, é possível aplicar técnicas de validação a esses modelos.

Podemos dizer que o resultado do modelo está adequado ao sistema modelado quando este resultado está dentro de um intervalo de confiança. Esse intervalo pode ser por inferência estatística. Deste modo, a seguir, apresentam-se os métodos utilizados para a obtenção dos intervalos de confiança utilizados nesse estudo.

2.11.1 Intervalo de Confiança para o Desempenho e a Consistência

As métricas de desempenho e consistência dos modelos propostos foram submetidas a procedimentos de validação (ver. Capítulo 6). Essas validações utilizaram o intervalo de confiança para a média com o desvio padrão (σ) não conhecido. Para verificar se os resultados do modelo representam o comportamento do sistema, verifica-se, inicialmente: se σ não é conhecido, se a amostra é aleatória, e (ou) se a população é normalmente distribuída ou $n \geq 30$.

A quantidade de dados amostrais pode ser selecionada em função do Teorema do Limite Central (MONTGOMERY; RUNGER, 2013) que estabelece que se amostras de tamanho n, em

que $n \geq 30$, são retiradas ao acaso de uma população qualquer com uma média m e um desvio padrão σ, então, a distribuição amostral das médias se aproxima de uma distribuição normal (AHSANULLAH; KIBRIA; SHAKIL, 2014).

A distribuição normal possui três características importantes para essa inferência baseada em intervalo de confiança, que são:

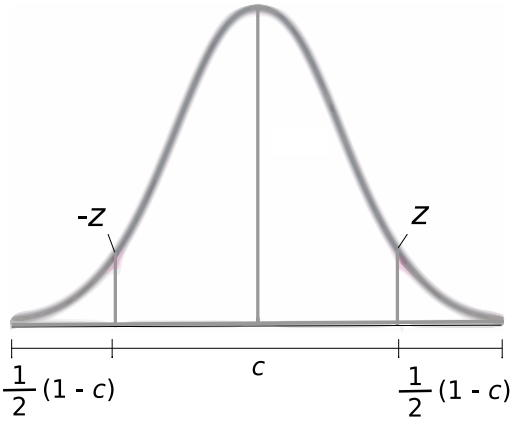
- possui um formato de sino e simétrica em torno da média;
- possui média no topo da curva;
- possui sob a curva a área total igual a 1.

Após a obtenção dos dados, verifica-se se eles obedecem a uma distribuição normal de probabilidade (D'AGOSTINO, 2017). Neste estudo, utilizamos a ferramenta Minitab para verificar a hipótese de que os dados obedecem a essa distribuição (SOFTWARE, 2021).

Caso os dados se aproximem da curva normal, estabelecemos o nível de confiança (c), que é a probabilidade de que a estimativa contenha o parâmetro populacional, ainda sob a suposição de que o processo de estimação é repetido em significativo número de vezes. O nível de confiança separa os resultados prováveis, ou seja, mais próximos à média (entre os valores críticos $-z$ e z) dos mais incomuns. Os resultados menos prováveis são conhecidos como cauda.

A Figura 10 demonstra graficamente um exemplo no qual temos que c é a área sob a curva normal entre -z_c e z_c. A área restante é $1 - c$; então, a área em cada cauda é $(1 - c)/2$. Os $escore\ z$ são constantes em função do nível de confiança. Para as validações desse estudo, consideramos os limites superiores e inferiores do intervalo de confiança, ou seja, realizamos testes bicaudais.

Figura 10 – Área sob curva normal



Fonte: Adaptado de (MONTGOMERY; RUNGER, 2013).

Para uma variável aleatória normalmente distribuída, a variável média amostral tem o comportamento da *distribuição t*. Quando o $sigma$ não é conhecido, a margem de erro E é obtida utilizando-se o desvio padrão amostral e o valor crítico de t (MONTGOMERY; RUNGER, 2013).

Dessa forma, para construir o intervalo de confiança para a média com o σ desconhecido, verificam-se as estatísticas amostrais n, $\bar{x}$ e σ, demonstradas na equação 2.11 e equação 2.12:

$$\bar{x} = \frac{\sum x}{n} \tag{2.11}$$

$$\sigma = \sqrt{\sum \frac{(x_i - \bar{x})^2}{n-1}} \tag{2.12}$$

Em seguida, identificam-se os graus de liberdade $(n-1)$, o nível de confiança c e o valor crítico de t. Calcula-se a margem de erro E, que é utilizada para a obtenção dos limites inferior e superior do intervalo de segurança, representada pela equação 2.13.

$$E = t_c \frac{\sigma}{\sqrt{n}} \tag{2.13}$$

Agora é possível calcular os limites superior (L_U) e inferior (L_L):

$$L_U = \bar{x} + E \tag{2.14}$$

$$L_L = \bar{x} - E \tag{2.15}$$

Dessa forma, temos o intervalo de confiança como: $\bar{x} - E < \mu < \bar{x} + E$

2.11.2 Método de KEESE e o Intervalo de Confiança para a Disponibilidade

A validação do modelo de disponibilidade foi realizada por meio da utilização de um *testbed* com injeção de falhas e reparos controlados. Para determinar o intervalo de confiança do processo de validação, utilizou-se o método de (KEESEE, 1965).

Após a obtenção dos dados, o procedimento foi iniciado com a adaptação da equação de disponibilidade 2.16.

$$A = \frac{\mu}{\lambda + \mu} = \frac{1}{1 + \frac{\lambda\lambda}{\mu}} = \frac{1}{1 + \rho} \tag{2.16}$$

Nesta equação ρ é a relação entre $\frac{\lambda}{\mu}$.

Na Equação 2.17 e na Equação 2.18, estimam-se, respectivamente, λ e μ onde temos n representando os eventos totais de falha e reparo. De forma semelhante, o tempo total de falha sendo S_n e o tempo total de reparo é Y_n são considerados:

$$\hat{\Lambda} = \frac{n}{S_n} \tag{2.17}$$

$$\hat{M} = \frac{n}{Y_n} \tag{2.18}$$

Um intervalo de confiança $100\times(1-\alpha)$ para λ é dado por uma distribuição qui-quadrada com os parâmetros definidos na Equação 2.19 e Equação 2.20.

$$\left(\frac{x^2_{2n;1-\frac{\alpha}{2}}}{2S_n}, \frac{x^2_{2n;\frac{\alpha}{2}}}{2S_n} \right) \tag{2.19}$$

$$\left(\frac{x^2_{2n;1-\frac{\alpha}{2}}}{2Y_n}, \frac{x^2_{2n;\frac{\alpha}{2}}}{2Y_n} \right) \tag{2.20}$$

$\hat{\rho}$ é o estimador de probabilidade máxima para a relação $\frac{\lambda}{mu}$, conforme Equação 2.21.

$$\hat{\rho} = \frac{\hat{\Lambda}}{\hat{M}} = \frac{\frac{n}{S_n}}{\frac{n}{Y_n}} = \frac{Y_n}{S_n} \tag{2.21}$$

O intervalo de confiança é dado pelo limite superior ρU e o limite inferior ρL, demonstrado pela função de probabilidade da Distribuição F, e apresentado na Equação 2.22 e Equação 2.23 respectivamente.

$$\rho U = \frac{\hat{\rho}}{F_{2n;2n;1-\frac{\alpha}{2}}} \tag{2.22}$$

$$\rho L = \frac{\hat{\rho}}{F_{2n;2n;\frac{\alpha}{2}}} \tag{2.23}$$

Finalmente podemos obter o estimador de probabilidade máximo para a disponibilidade $\hat{A} = \frac{1}{1+\hat{\rho}}$. Considerando que a disponibilidade A é uma função monotonicamente decrescente de ρ, o intervalo de confiança 100*(1-α) para A, ou seja, A_U e A_L, é dado na Fórmula 2.24.

$$\left(\frac{1}{1+\rho_L}, \frac{1}{1+\rho_U}\right) \tag{2.24}$$

2.12 CONSIDERAÇÕES FINAIS

Este capítulo introduziu os principais conceitos necessários para melhor entendimento deste trabalho. Inicialmente, foram introduzidos ao leitor os conceitos de SGBD NoSQL, consistência eventual e consistência baseada em quorum.

Discorremos sobre as definições de confiabilidade e disponibilidade e, posteriormente, abordamos os conceitos da avaliação de desempenho e injeção de falhas, assim como foram demonstrados Rede de Petri, rede de Petri Determinística e Estocástica e o diagrama de blocos de confiabilidade. Por fim, foram explicadas as técnicas para obtenção dos intervalos de confiança utilizados nessa pesquisa.

3 TRABALHOS RELACIONADOS

"A tecnologia só é tecnologia para quem nasceu antes dela ter sido inventada."

— Alan Kay

Este Capítulo descreve os principais trabalhos relacionados ao tema desta tese de doutorado. Inicialmente, são apresentados trabalhos representativos na área de modelagem, avaliação de desempenho e disponibilidade de sistemas. Posteriormente, abordam-se trabalhos com foco na consistência NoSQL baseada em quorum, bem como técnicas de modelagem e estimativa de consistência; o consumo de energia também é abordado.

3.1 MODELAGEM, DESEMPENHO E DISPONIBILIDADE

Modelagem de desempenho e disponibilidade são tarefas importantes, e diversos estudos têm sido propostos com o objetivo de avaliar esses atributos em sistemas computacionais.

(RODRIGUES et al., 2019) avaliam métricas de qualidade de serviço *Quality of Service* (QoS) em SGBD NoSQL, considerando cargas de trabalho distintas, mecanismos de redundância e *Pool Size*, que são *threads* que executam requisições de processos simultaneamente. Os autores apresentam uma abordagem baseada em *Generalized Stochastic Petri net* (GSPN) para avaliação de desempenho e disponibilidade do sistema. O SGBD NoSQL MongoDB foi utilizado para validação do modelo. Os resultados demonstraram a importância da avaliação do desempenho, e foram influenciados principalmente pelo *Pool Size*, que obteve variação de mais de 300% nos resultados. Para a disponibilidade, a variação obtida por influência do número de réplicas do SGBD variou entre 45 horas e 229 horas por ano. Apesar dos importantes resultados, este trabalho não avaliou a consistência dos dados.

(ARAUJO et al., 2021) comparam o desempenho dos SGBD NoSQL Cassandra e MongoDB em dois experimentos executados na *Amazon Web Services* (AWS) EC2 cloud platform. Foram considerados 3 servidores em cada *cluster* de SGBD; para minimizar a diferença na arquitetura destes SGBDs, o Cassandra foi configurado sem replicação de dados. O primeiro experimento utilizou a ferramenta YCSB para geração de cargas de trabalho distintas. Nesse experimento, o Cassandra obteve o melhor desempenho em todas as cargas avaliadas. Na segunda categoria, foi simulado um banco de dados de investidores do Tesouro Nacional, com operações específicas. O MongoDB obteve melhor resultado nesse segundo experimento, não obtendo melhor

desempenho que o Cassandra apenas nas operações de *update*. Os aspectos de disponibilidade e consistência não foram avaliados.

(CHATTARAJ; SARMA; SAMANTA, 2019) utilizam um modelo baseado em rede de Petri estocástica para representar um sistema de arquivos distribuídos Hadoop *Hadoop Distributed File System* (HDFS); os atributos analisados são confiabilidade e disponibilidade do serviço. Os resultados apontam que o número de blocos nos quais a informação é dividida possui grande influência na disponibilidade do serviço assim como o número de nós do HDFS; e que a adição de replicações em bloco torna o HDFS mais tolerante a falhas. Conclui-se ainda que para a política de replicação do sistema de arquivos distribuído Hadoop, o fator de replicação padrão igual a 3 é a melhor escolha para maior disponibilidade do sistema. O desempenho não foi analisado pelos autores.

(GORBENKO; ROMANOVSKY; TARASYUK, 2020) analisam a interação do desempenho do SGBD Cassandra NoSQL e suas configurações de consistência baseadas em quórum. Utilizou-se o YCSB para a geração da carga de trabalho nos experimentos realizados no ambiente Amazon EC2, com três servidores replicados. Os procedimentos utilizaram leitura e escrita em diferentes proporções. Os resultados relatados mostram que as configurações de consistência usadas podem afetar significativamente o tempo de resposta do Cassandra. Foi verificado que a consistência forte obteve um desempenho até 26% menor, em relação a outros níveis de consistência, e a melhor configuração para consistência forte depende da razão das operações de leitura e escrita. O trabalho destaca, que as configurações de consistência ideais que maximizam o desempenho do Cassandra, dependem significativamente da carga de trabalho atual e da proporção de leitura e escrita em execução.

(NOUDOUST; ADABI; REZAEE, 2022) realizam experimentos com diferentes níveis de consistência de dados baseados em quorum, para verificar o desempenho e a consistência de SGBDs NoSQL. São realizadas operações de leitura e gravação em dois conjuntos de dados de diferentes tamanhos. Verificou-se nesses experimentos que o nível de consistência 1 é o mais sensível para o número de leituras desatualizadas, obtendo 5 vezes mais acessos desatualizados que o nível de consistência 2. Na avaliação do desempenho, os menores níveis de consistência obtiveram o maior número de operações por segundo em todos os experimentos, e os maiores níveis de consistência indicam menor disponibilidade. Este trabalho realiza a análise experimental do desempenho e da consistência, sem, contudo, executar nenhum método para a avaliação de disponibilidade nem propor modelos que avaliem as métricas em conjunto.

Os trabalhos apresentados demonstram os benefícios da modelagem e avaliação princi-

palmente de atributos de desempenho e disponibilidade em diferentes sistemas. Nesta tese, os modelos possibilitam a estimativa do desempenho em função do ambiente, assim como a disponibilidade do sistema, considerando taxas de falha e recuperação de um servidor NoSQL baseado em quorum. Adicionalmente, esta tese fornece ferramentas para os projetistas avaliarem os componentes do sistema antes de sua implementação.

3.2 MODELAGEM E AVALIAÇÃO DA CONSISTÊNCIA NOSQL

Nos últimos anos, diversas técnicas têm sido propostas para lidar com a consistência em SGBD NoSQL. É importante mencionar que o problema de avaliação da consistência baseada em quorum verifica a capacidade do sistema retornar à versão mais recente do dado. Os principais desafios são entender o impacto da escolha da consistência no desempenho e na disponibilidade desses sistemas, e quantificar essa consistência. Contudo, até onde se pôde observar, os trabalhos não abordam essas questões em conjunto.

O protocolo de consistência chamado LibRe é apresentado por (KUMAR et al., 2018). Esse protocolo melhora a possibilidade de leitura do dado mais recente em SGBD NoSQL baseado em quorum. O protocolo utiliza um registro na memória que armazena o conjunto de servidores de réplica e contém a versão mais recente dos itens de dados até que todas as réplicas estejam atualizadas. Esse registro é acessado durante a leitura para encaminhar as solicitações para os servidores que contêm a versão mais recente do dado. O Cassandra NoSQL foi utilizado para os experimentos. Os resultados apresentados demonstram que o LibRE obteve para a consistência eventual resultados semelhantes à consistência forte oferecida pelo Cassandra, sem perda significativa de latência. Para aumentar a quantidade de dados não atualizados nas réplicas da instância de dados, o Cassandra foi modificado para que, ao invés de propagar a atualização para todos os três servidores de réplicas, a atualização seja propagada para apenas dois desses servidores. Esse trabalho não deixa claro o ganho do protocolo correspondente ao desempenho e à consistência dos dados acessados para a replicação padrão do sistema, que corresponde ao fator de replicação.

Em (BAILIS et al., 2014) os autores apresentam um modelo para estimar a relação entre a latência das operações e a consistência dos dados no SGBD NoSQL, introduzindo o conceito *Probabilistically Bounded Staleness* (PBS). Este modelo utiliza o método Monte Carlo para fornecer uma observação quantitativa da garantia de consistência (isto é, previsão da consistência). A quantidade de réplicas do SGBD N, o número de servidores que confirmam a

leitura R e o número de servidores que confirmam a escrita W também são considerados. Seus experimentos demonstram que a redução dos valores de R e W reduzem significativamente a latência das operações. Maiores garantias de consistência foram encontradas nos experimentos realizados utilizando discos *Solid State Drive* (SSD). No entanto, os autores não consideram a disponibilidade do sistema.

(YAO; WANG, 2020) analisam a Probabilistic Consistency Guarantee (PCG) em SGBD NoSQL baseado em quorum. O modelo PCG estima o quorum necessário para atingir a consistência após a última gravação com a menor latência da operação. A análise apresentada determina o tamanho do quorum de leitura, e os resultados são atualizados pela última gravação na probabilidade especificada pelo usuário. Quando algumas réplicas estão indisponíveis, o modelo redimensiona a consistência das operações para reduzir as leituras desatualizadas. Os resultados experimentais avaliados no Cassandra demonstraramm predições até 77,7% mais precisas que o modelo PBS de (BAILIS et al., 2014).

No trabalho de (KRECHOWICZ; DENIZIAK; ŁUKAWSKI, 2021), é avaliada a consistência da arquitetura *Scalable Distributed Two-layered Data Structures* (SD2DS), desenvolvida para otimizar o gerenciamento de dados grandes em ambiente NoSQL distribuídos, com características de alta escalabilidade, disponibilidade e taxa de transferência. A arquitetura considera a execução simultânea de algumas tarefas e a organização em filas dessas tarefas. Os protótipos desenvolvidos consideraram variações na política de escalonamento do SD2DS. Os resultados obtidos demonstram uma redução significativa na inconsistência dos dados e melhora no desempenho das operações para as soluções propostas.

(KHELAIFA; BENHARZALLAH; KAHLOUL, 2022) propõem um sistema de consistência baseado na consistência causal e na ordem das operações, o MinidoteACE, para a computação de borda, que utiliza aplicativos com requisitos de consistência diferentes. Utilizando o SGBD AntidoteDB, com operações assíncronas semelhante ao NoSQL, além de ser tolerante a falhas e possuir alta disponibilidade. Os pesquisadores avaliam o MinidoteACE utilizando um gerador de carga de trabalho para aferir o desempenho das operações na consistência causal e consistência forte em operações de leitura e atualização. Os resultados experimentais realizados na arquitetura avaliada demonstraram que a consistência forte obteve um rendimento somente de até 10% menor comparado a consistência causal, nos experimentos predominância de leitura. No entanto, quando houve a proporção a partir de 25% de atualizações, a latência das operações foi até três vezes maior na consistência forte.

(HUANG et al., 2017a) utilizam modelagem em *Colored Petri net* (CPN) para desenvolver

um simulador que analisa e possibilita o ajuste de sistema NoSQL baseado em quorum, o SGBD Cassandra NoSQL foi utilizado para os experimentos e validação do modelo. Vários cenários foram simulados utilizando instâncias EC2 da AWS. As estimativas de interesse são o desempenho e o tempo de inconsistência entre réplicas e os resultados do trabalho enfatizaram a adequação do modelo ao sistema. Apesar da similaridade na avaliação do desempenho, este trabalho não considera a disponibilidade do sistema. Ademais, a avaliação de consistência não considera a probabilidade de acesso à informação mais recente.

O trabalho de (SINGLA et al., 2018) propõe e avalia um modelo teórico para estimar o número de violações da Consistência Sequencial (CS) em um sistema de rede social implementado com o Cassandra NoSQL. Uma violação da CS consiste em um acesso não sequencial à informação. A estimativa otimiza o relacionamento entre desempenho, utilização de recursos e violações da CS. Os autores sugerem uma nova forma de consistência que elevou a taxa de transferência avaliada em 2,4 vezes. Nos experimentos realizados, o novo modelo proporcionou uma melhoria na latência de até 37% em relação ao aplicativo original.

A influência do número de réplicas e do número de servidores que confirmam operações de gravação e leitura no ambiente de replicação NoSQL é considerada por (BURDAKOV et al., 2016). Um modelo de consistência eventual é descrito para estimar a probabilidade de que, durante o processo de replicação, exista pelo menos uma solicitação de leitura não atualizada. Os autores também propõem um modelo de estimativa de consistência forte, baseado na espera da atualização do registro e concluem que modelos matemáticos são adequados para a avaliação de características de consistência em bancos de dados replicados. No entanto, a disponibilidade do ambiente não foi verificada.

(ATTIYA; ELLEN; MORRISON, 2016) apresentam modelos matemáticos para realizarem a comparação da consistência eventual com o modelo de consistência Observable Causal Consistency (OCC) em bancos de dados NoSQL. Eles demonstram que a consistência do armazenamento eventualmente consistente não pode ser mais forte que um modelo de OCC, os experimentos demonstraram que a consistência eventual possui uma garantia menor que a OCC. Esse modelo não considera o desempenho ou a disponibilidade do sistema.

(GAO et al., 2021) implementam alterações na replicação do SGBD Cassandra NoSQL adicionando abordagens de consistência causal, consistência forte e consistência quorum baseada no algoritmo bizantino de tolerância a falhas. Essas alterações são avaliadas observando o impacto na latência das operações. O YCSB foi utilizado para gerar a carga de trabalho no ambiente construído na Google Cloud Plataform. A menor latência foi obtida no nível pa-

drão de consistência eventual do Cassandra. A consistência causal obteve o segundo melhor resultado, seguido pelo nível quórum do Cassandra. As demais customizações de consistência obtiveram desempenho inferior à consistência eventual do Cassandra NoSQL. A disponibilidade do SGBD não foi avaliada.

(SHAH; KOTHARI; PATEL, 2022) analisam trabalhos sobre o consumo de energia em SGBD NoSQL. São comparados diferentes bancos de dados, operações, métricas, ferramentas de medição do consumo e *benchmarks*. Os pesquisadores elencam entre as principais contribuições a análise do consumo de energia para os níveis de consistência e a latência das operações.

Diferentemente dos trabalhos citados, esta proposta de tese propõe uma abordagem unificada para estimativa de desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e de consumo de energia no ambiente NoSQL baseado em quorum. Para o desempenho, o modelo considera o tempo de leitura ou escrita da operação no servidor NoSQL, número de servidores e nível de consistência. Adicionalmente, para a disponibilidade, são considerados atributos como o MTTF e o MTTR em servidores NoSQL. Consideramos ainda o arranjo lógico e físico do sistema.

A probabilidade de acesso ao dado mais recente considera a expectativa de o dado estar atualizado em todos os servidores redundantes. E, por fim, estimamos o consumo de energia em função das configurações de consistência. Os modelos concebidos utilizam técnicas baseadas em SPN e RBD e possibilitam que, sem a necessidade da utilização do sistema de armazenamento NoSQL baseado em quorum, sejam estimados o desempenho, a disponibilidade do sistema, a probabilidade de acesso ao dado atualizado e o consumo de energia. Isso fornece ao projetista da solução elementos para a escolha de uma configuração que atenda aos requisitos do sistema.

A Tabela 3 resume a comparação dos trabalhos. Embora tais abordagens não levem em conta todas as questões, apenas lidem com um subconjunto dos problemas, a técnica proposta nesta tese difere em algumas suposições. Por exemplo, nosso trabalho avalia a probabilidade de os dados estarem consistentes em todas as réplicas de acordo com o nível de consistência, enquanto os autores (BAILIS et al., 2014) avaliam a probabilidade de ler os dados mais recentes após uma confirmação da escrita. Em (BURDAKOV et al., 2016), a abordagem avalia a probabilidade de consistência no sistema, mas não considera a natureza estocástica dos componentes do sistema. Além disso, alguns trabalhos, por exemplo, (SINGLA et al., 2018) não avaliam a consistência do quorum. Ademais, os modelos propostos nessa tese são generalizados para SGBDs NoSQL baseado em quorum.

Tabela 3 – Comparação entre esta tese e trabalhos relacionados

Autor	NoSQL	Modelo	Desempenho	Disponibilidade	Consistência
(RODRIGUES et al., 2019)	✓	✓	✓	✓	-
(ARAUJO et al., 2021)	✓	-	✓	-	-
(CHATTARAJ; SARMA; SAMANTA, 2019)	-	✓	-	✓	-
(GORBENKO; ROMANOVSKY; TARASYUK, 2020)	✓	-	✓	-	✓
(NOUDOUST; ADABI; REZAEE, 2022)	✓	-	✓	-	✓
(YAO; WANG, 2020)	✓	✓	✓	-	✓
(KUMAR et al., 2018)	✓	-	✓	-	✓
(BAILIS et al., 2014)	✓	✓	✓	-	✓
(KRECHOWICZ; DENIZIAK; ŁUKAWSKI, 2021)	✓	-	✓	-	✓
(KHELAIFA; BENHARZALLAH; KAHLOUL, 2022)	-	-	✓	-	✓
(HUANG et al., 2017a)	✓	✓	✓	-	✓
(SINGLA et al., 2018)	✓	✓	✓	-	✓
(BURDAKOV et al., 2016)	✓	✓	-	-	✓
(ATTIYA; ELLEN; MORRISON, 2016)	✓	✓	-	-	✓
(GAO et al., 2021)	✓	✓	-	-	✓
(SHAH; KOTHARI; PATEL, 2022)	✓	-	✓	-	✓
Este trabalho	✓	✓	✓	✓	✓

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais trabalhos relacionados à área de pesquisa desta tese. Inicialmente, foram apresentados trabalhos representativos na área de modelagem, desempenho e disponibilidade de sistemas. Posteriormente, foram discutidos trabalhos na área de modelagem, consistência NoSQL baseado em quorum e consumo de energia com ênfase na comparação de desempenho em função da consistência e sua quantificação.

4 MÉTODO PROPOSTO

> *"A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê."*
>
> — Arthur Schopenhauer

Esta tese tem como importante objetivo desenvolver e analisar modelos que estimem o desempenho, a disponibilidade, a probabilidade de acesso aos dados mais recentes e o consumo de energia em SGBD NoSQL baseado em quorum. Por meio desses modelos são fornecidas métricas que possibilitam a obtenção da melhor configuração do ambiente para a prioridade desejada do sistema.

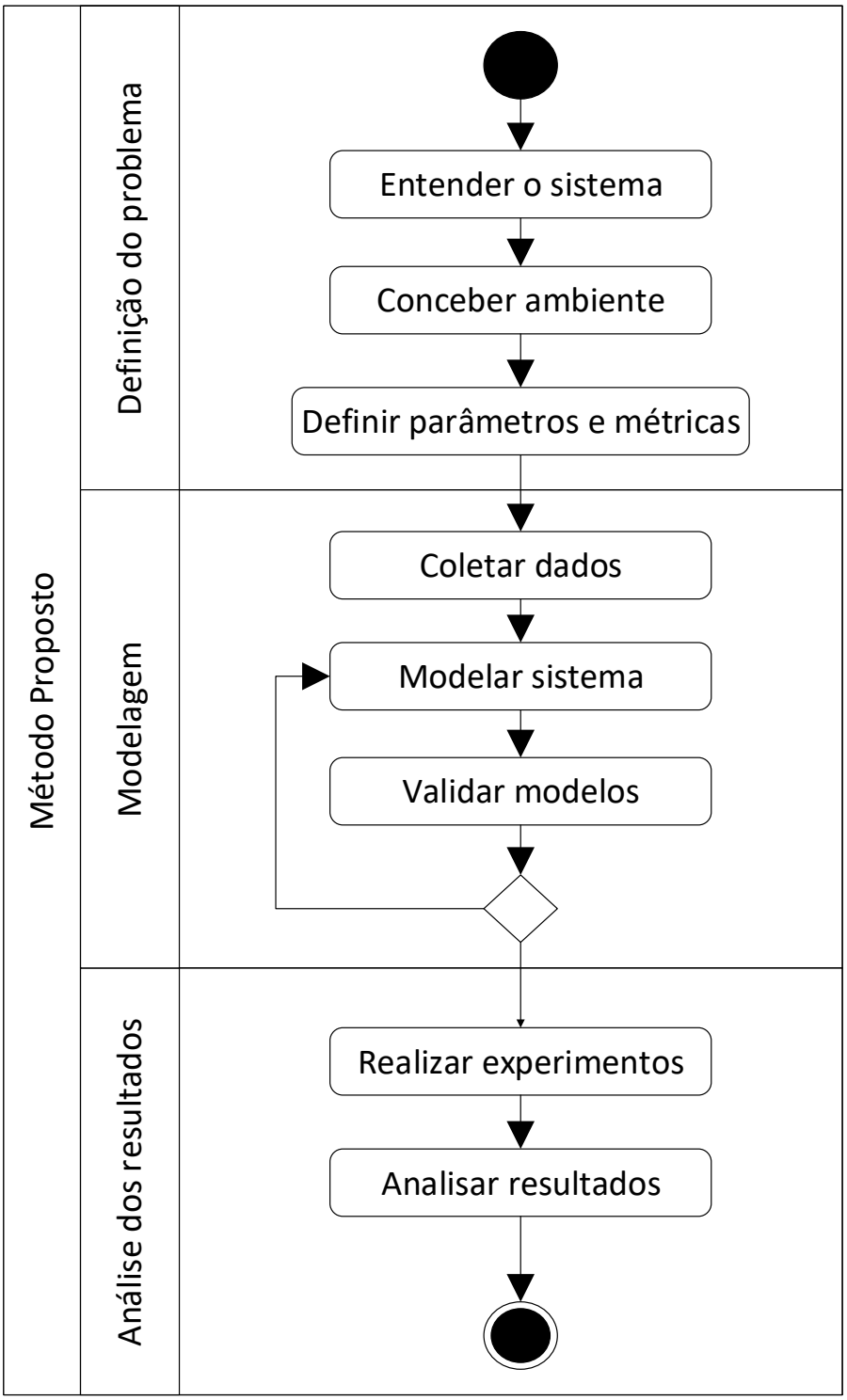
A seguir, apresentamos a metodologia utilizada para atingir os objetivos dessa pesquisa. Entre os quais consta fornecer suporte ao arquiteto do sistema para a escolha da configuração adequada ao nível de serviço desejável ao sistema.

A metodologia proposta pode ser aplicada por usuários com conhecimento básico em modelagem de sistemas, formalismos SPN e RBD para projeto de sistemas de banco de dados NoSQL baseado em quorum. O método permite ainda que as métricas desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia possam ser avaliadas, de maneira isolada ou em conjunto.

Essa avaliação possibilita ao projetista do sistema o conhecimento da relação entre esses fatores, de modo a fomentar a seleção da estrutura mais adequada aos requisitos do sistema avaliado.

Foram consideradas as seguintes métricas: latência, ou seja, o tempo de uma operação, que pode ser leitura ou escrita; disponibilidade, que representa a probabilidade de o sistema estar em funcionamento, verificada em diferentes configurações; probabilidade de acesso ao dado mais recente, que está definida neste trabalho como a probabilidade de acesso ao dado mais recente dentre as réplicas do SGBD; e consumo de energia, que considera a leitura e a escrita realizada no SGBD NoSQL baseado em quorum. Em acréscimo, são considerados diferentes números de servidores redundantes para consistência eventual.

Figura 11 – Método proposto



Fonte: Elaboração Própria.

O fluxograma apresentado na Figura 11 demonstra graficamente o método proposto para as fases de definição do problema, modelagem e análise dos resultados. Os retângulos representam

cada etapa da metodologia seguida, e obedecem à ordem de execução direcionada pelas setas, o conjunto de etapas pertence a uma fase. É possível que o fluxo retorne à fase anterior quando identificada a necessidade de que algum requisito seja revisto. O losango representa uma etapa que pode seguir por dois caminhos diferentes, dependendo do resultado obtido. Dessa forma, uma etapa pode prosseguir apenas se todas as condições forem satisfeitas com sucesso, caso contrário, segue-se na mesma etapa ou retorna-se à fase anterior para ajustes de execução. O método proposto apresenta como resultado os modelos desenvolvidos e as avaliações de ambientes selecionados.

A metodologia utilizada consiste nas fases ou atividades:

- **definição do problema** – consiste no estudo para entendimento dos sistemas NoSQL e que compreende a identificação de suas características, considerando relevância e aspectos de impacto na qualidade do serviço desses sistemas. Ainda nessa fase, o ambiente é concebido para que o projetista possa verificar o sistema em execução. Os parâmetros e as métricas decorrem desse estudo e representam características importantes para os objetivos do sistema avaliado;

- **modelagem** – nesta fase observam-se os principais aspectos voltados para o funcionamento do sistema, dessa forma, esses aspectos são representados em modelos formais. Além disso, dados oriundos de outros estudos e avaliação de sistemas reais são coletados para orientação dos modelos. Quando a viabilidade da inclusão de métricas distintas em um mesmo modelo não é verificada em razão, por exemplo, do aumento da complexidade ou do custo computacional do modelo, outro modelo é proposto de forma a atender essa outra métrica. Ainda nesta fase, os modelos são validados por método estatístico, observando a aproximação entre o sistema em funcionamento e o comportamento do modelo. Após a validação do modelo, técnicas de projeto de experimentos são utilizadas para selecionar ambientes de estudo e aplicá-las nos modelos desenvolvidos;

- **análise de resultados** – esta análise é realizada em função da comparação estatística dos resultados e fatores avaliados nos modelos, nos quais são verificados diversos ambientes para desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia. Discussões dos resultados são realizadas nessa análise.

4.1 DEFINIÇÃO DO PROBLEMA

Esta seção detalha o estudo realizado na definição do problema para compreender o escopo do sistema, conceber o ambiente e elicitar os parâmetros e métricas a serem avaliados na pesquisa. Esta fase utiliza os passos:

4.1.1 Entender o sistema

Para avaliar a consistência em SGBDs NoSQL baseados em quorum, é importante que o projetista do sistema entenda como esses sistemas funcionam, começando por identificar as restrições e os componentes a serem avaliados, visando ao entendimento do impacto na qualidade esperada do serviço. Para tanto, é necessário o conhecimento das principais soluções utilizadas com suas respectivas vantagens e desvantagens.

Atribuímos muita importância a esta fase do projeto, já que é nessa etapa que se promove a compreensão geral das características em comum dos sistemas gerenciadores de banco de dados NoSQL baseados em quorum. O entendimento desses sistemas demanda considerável atenção devido às suas características mais flexíveis e ao seu menor controle. Essa atividade permite a prospecção das características e métricas a serem avaliadas pelo projetista do sistema.

4.1.2 Conceber o ambiente

Após o entendimento do sistema, aplicamos a etapa de concepção do ambiente, que consiste na implementação de uma plataforma de experimentação (*testbed*) para verificar se a compreensão do sistema foi realizada corretamente. O *testbed* compreende as seguintes atividades:

1. **escolha das tecnologias** – mais especificamente, ambiente de virtualização, SGBD, ferramenta de *benckmark* e sistema operacional;

2. **instalação e configuração do ambiente** – para a obtenção de resultados com menor interferência de fatores externos à pesquisa, utilizamos apenas as aplicações essenciais para a execução e monitoramento do sistema. Ademais, os SGBDs utilizaram configuração padrão;

3. **gerenciamento do ambiente** – o *testbed* deve permitir que a execução do sistema seja monitorado no nível necessário para a avaliação de seu comportamento.

Nessa etapa, buscamos a seleção de um ambiente que permitisse a verificação da técnica de quorum, considerando soluções *free* e *open-source*, de maneira a facilitar a replicação da montagem desse ambiente. Ademais, adicionamos um segundo SGBD NoSQL para verificar se a técnica desenvolvida pode ser generalizada para outros SGBDs NoSQL baseados em quorum.

4.1.3 Definir parâmetros e métricas

Na etapa de definição de parâmetros e métricas, o projetista do sistema deve elencar inicialmente as métricas de interesse. Após essa definição, adicionalmente ao conhecimento do sistema e concepção do ambiente, é possível a seleção dos parâmetros. Para tanto, devem ser identificados aqueles que possuem maior influência nas métricas selecionadas.

Nesta tese, obtivemos como parâmetros essenciais para a abordagem dos SGBDs NoSQL baseados em quorum, em todas as métricas, o nível de consistência da operação e o fator de replicação dos dados. Para a avaliação do desempenho e da probabilidade de acesso ao dado mais recente, consideramos a demanda de operações (escrita ou leitura) e a capacidade dos servidores executarem essas operações. Em relação à disponibilidade, utilizou-se o tempo médio para a falha e o tempo médio para reparo dos componentes do sistema.

Dessa forma, métricas de desempenho, disponibilidade, consistência e consumo de energia foram elicitadas nesta tese.

4.2 MODELAGEM

Esta seção apresenta a sequência de ações para a construção dos modelos, que correspondem a representação da abstração do comportamento desejado do sistema. Para a execução da modelagem, as etapas anteriores são pré-requisitos fundamentais.

4.2.1 Coleta de dados

A obtenção dos valores de entrada do modelo é realizada na etapa de coleta de dados. Para a correta representação do sistema, a coleta de dados pode ser inicialmente executada por

medições, que são executadas em experimentos no ambiente concebido na fase anterior. Para a execução desses experimentos podem ser demandadas ferramentas adicionais, que podem ser *benchmarks* conhecidos ou até desenvolvidos para o propósito desses experimentos. Podem-se, ainda, utilizar dados verificados na observação de pesquisas com parâmetros em comum. Outra alternativa aos experimentos pode ser a obtenção dos valores nos *logs* de sistemas em funcionamento.

As métricas foram estimadas em modelos distintos e os parâmetros foram coletados com base em diferentes técnicas de geração de carga de trabalho, para a avaliação do desempenho, foi utilizado um *benchmark* para a geração de carga de trabalho. Na avaliação da dispobililidade, observou-se os MTTF e MTTR disponíveis na literatura. Para a probabilidade de acesso ao dado mais recente, utilizou-se um *script* desenvolvido para a geração de carga de trabalho adequada para a avaliação do modelo. Os dados para o modelo de consumo de energia foram coletados do trabalho realizado em (GOMES; TAVARES; JUNIOR, 2016).

4.2.2 Modelar sistema

O comportamento do sistema pode ser representado em um único modelo. No entanto, para diminuir a complexidade desse modelo e consequentemente a demanda computacional dos resultados, pode ser melhor desenvolver modelos distintos para verificação desses comportamentos. Para o desenvolvimento dos modelos, é salutar a utilização de uma ferramenta de modelagem visual, que permita o desenvolvimento dos modelos e a aplicação das métricas elicitadas. Uma importante ferramenta de modelagem é o Mercury (BASHIR; LUŠTREK, 2021).

Nesta tese foram desenvolvidos modelos distintos em SPN e RBD para avaliar as métricas de interesse. A modelagem SPN foi escolhida por conciliar características importantes para análise desse trabalho, a exemplo de concorrência, sincronismo e processos de decisão.

RBD foi escolhida devido à interdependência de seus componentes, para que o sistema continue funcionando; além do fato de não existir prioridade entre os componentes, essas características auxiliaram na avaliação da disponibilidade desses sistemas.

Para a construção desses modelos foi utilizada uma abordagem iterativa, onde várias versões de cada modelo foram desenvolvidas até a adequada representação do comportamento do sistema. Dessa forma, foram desenvolvidos modelos de alto nível que conseguem estimar as métricas de desempenho, disponibilidade e probabilidade de acesso ao dado mais recente. Foi proposto, ainda, um modelo matemático para estimar o consumo de energia com as ca-

racterísticas encontradas nos modelos anteriormente desenvolvidos nessa pesquisa.

4.2.3 Validar modelos

Esta etapa de validação verifica se cada modelo representa o nível de abstração utilizado para representar o comportamento do sistema. Nesta etapa o modelo é alimentado com os valores coletados do sistema em funcionamento e os resultados são comparados por método estatístico. Essa comparação estatística indica se o modelo representa o sistema. Caso o modelo não represente o sistema, retorna-se à etapa de modelagem do sistema para nova análise e verificação dos componentes avaliados, seguindo novamente o fluxo.

Para a realização desta etapa, utilizou-se o ambiente concebido na fase de definição do problema, o YCSB, um *benchmark* reconhecido em pesquisas NoSQL (LIU; ZHOU, 2021), outras ferramentas de geração de carga e *scripts* de injeção de falhas desonvolvidos para essa pesquisa.

4.3 ANÁLISE DOS RESULTADOS

A fase de análise dos resultados possui como requisitos o entendimento do sistema, a definição dos parâmetros e métricas e os modelos desenvolvidos. Compreende as etapas de realização dos experimentos e análise dos resultados.

4.3.1 Realizar experimentos

Nesta etapa realizam-se os experimentos com os modelos concebidos. Estes modelos permitem a verificação do comportamento dos sistemas sem a necessidade da implementação de todos os cenários que serão avaliados.

O método aplicado utiliza nesta etapa a técnica baseada no DoE por meio da qual foram adotados experimentos l^k fatoriais, onde l são os níveis e ^{k} representa os fatores do experimento, para avaliar desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia em ambiente NoSQL baseado em quorum. Nessas técnicas são considerados diversos tratamentos para a avaliação do sistema.

Para cada modelo, são propostos os fatores e níveis para avaliação dos ambientes e percepção das relações entre as métricas de interesse. O objetivo é prover resultados para verificação

dos fatores de maior impacto sobre a métrica.

4.3.2 Analisar resultados

Os resultados são avaliados com a análise da *classificação dos efeitos dos fatores avaliados nos experimentos* (ver Seção 2.6), e a comparação dos resultados dos modelos distintos para os parâmetros em comum dos modelos, nível de consistência da operação e fator de replicação. Entre as questões que essas análises podem ajudar a responder, constam:

- como deve ser configurado o sistema para melhor desempenho;
- de que modo deve ser configurado o sistema para melhor disponibilidade;
- qual a forma de prover melhor acesso ao dado atualizado;
- qual o impacto da consistência para a obtenção de melhor desempenho e disponibilidade;
- como obter uma configuração com menor consumo de energia conforme o perfil de utilização;
- como obter a melhor relação entre estes requisitos de desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia que melhor atenda à necessidade da solução.

A análise dos resultados verifica os valores obtidos do modelo com base em cenários variados, com a expectativa de que o mesmo comportamento possa ser confirmado por um sistema em funcionamento. Dessa forma, é possível perceber a influência de determinado parâmetro e atuar de maneira mais eficaz na métrica ou conjunto desejado. A escolha da correta combinação de parâmetros fornece o melhor cenário baseado nas características utilizadas, por exemplo, com menor número de servidores pode-se, em caso específico, obter-se melhor desempenho e consistência com uma disponibilidade que atende ao requisito do sistema e utilizando menos energia.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo inicialmente apresentou a introdução do método proposto. Em seguida, apresenta o método organizado em fases, em que cada fase possui suas respectivas etapas e atividades

des. O método proposto avalia o desempenho, a disponibilidade, a probabilidade de acesso ao dado mais recente e o consumo de energia em SGBDs NoSQL baseado em quorum. Considera-se a consistência como o foco principal na avaliação. Este capítulo proporciona detalhes do planejamento e avaliação que permite a replicação por outros pesquisadores.

5 MODELOS

"Aqueles que se sentem satisfeitos sentam-se e nada fazem. Os insatisfeitos são os únicos benfeitores do mundo."

— Walter S. Landor

A modelagem de sistemas permite o prognóstico dos parâmetros para o provimento do serviço, com vistas ao direcionamento dos recursos e configurações necessárias ao nível de serviço desejado, assim como com vistas à antecipação da probabilidade de falha desses sistemas, por intermédio das configurações do seu ambiente. O uso de modelos que permitam a avaliação do desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia em SGBDs NoSQL baseados em quorum pode ser de grande relevância para os provedores de serviços que utilizam dessa infraestrutura. Esses aspectos podem impactar positivamente ou inviabilizar o provimento do serviço.

Este capítulo apresenta os modelos SPN adotados para estimar o desempenho, a disponibilidade e o acesso ao dado atualizado no ambiente NoSQL baseado em quorum. Em seguida, apresenta o modelo RBD, desenvolvido para a avaliação da disponibilidade considerando os componentes da infraestrutura desses sistemas. Por fim, propõe-se um modelo matemático para estimar o consumo de energia. Modelos com técnicas distintas foram desenvolvidos para facilitar a análise das métricas propostas.

Redes de Petri Estocásticas foram adotadas por conciliar características importantes de redes de Petri, redes de filas e cadeia de Markov. Uma vez que não foi viável desenvolver todas as métricas em um mesmo modelo, foram concebidos os modelos de desempenho e disponibilidade, e o modelo de probabilidade da consistência. Ambos representam a consistência eventual característica desses sistemas.

Por permitir a avaliação da disponibilidade considerando isoladamente os componentes do sistema, foi utilizada a modelagem RBD. Isso permitiu um maior nível de detalhe nessa avaliação. O consumo de energia foi estimado considerando as características do funcionamento do sistema. Dessa forma, as métricas desempenho, disponibilidade, probabilidade de consistência e consumo de energia podem ser acessadas para diferentes configurações do sistema, permitindo ao projetista o melhor ajuste necessário aos seus requisitos.

São adotados os seguintes operadores para os modelos SPN: $\#p$ representa o número de *tokens* em p; $P\{exp\}$ representa a probabilidade da expressão exp ser verdadeira; e $E\{\#p\}$

representa o número médio de *tokens* no lugar p. Como modelos SPN são avaliadas usando-se a geração da CTMC equivalente, neste trabalho, $P\{exp\}$ é estimado pela soma das probabilidades de estado de cada CTMC que satisfaz a expressão exp.

As propriedades de uma Rede de Petri podem restringir o método de análise do modelo que pode ser estacionário ou transiente. A análise estacionária pode ser utilizada quando não há restrição da exploração das possibilidades do modelo para a obtenção do resultado. Quando não é possível a avaliação de todas as possibilidades do modelo, pode-se utilizar a análise transiente, na qual a avaliação do modelo é dependente do tempo.

Neste trabalho buscou-se a avaliação estacionária (MARUSSY et al., 2016) dos modelos SPN. Dessa forma, os modelos SPN concebidos atenderam a estes três requisitos fundamentais para a análise estacionária: *Boundedness*, *Home state* e *Liveness* (ver Seção2.8.1)

5.1 MODELO DE DESEMPENHO E DISPONIBILIDADE

Essa abordagem utilizada para a modelagem considera um sistema de armazenamento que utiliza SGBD NoSQL com a consistência baseada em quorum. Dessa forma, as operações são realizadas com sucesso quando é possível atender o nível de consistência requisitado (ver Seção 2.3).

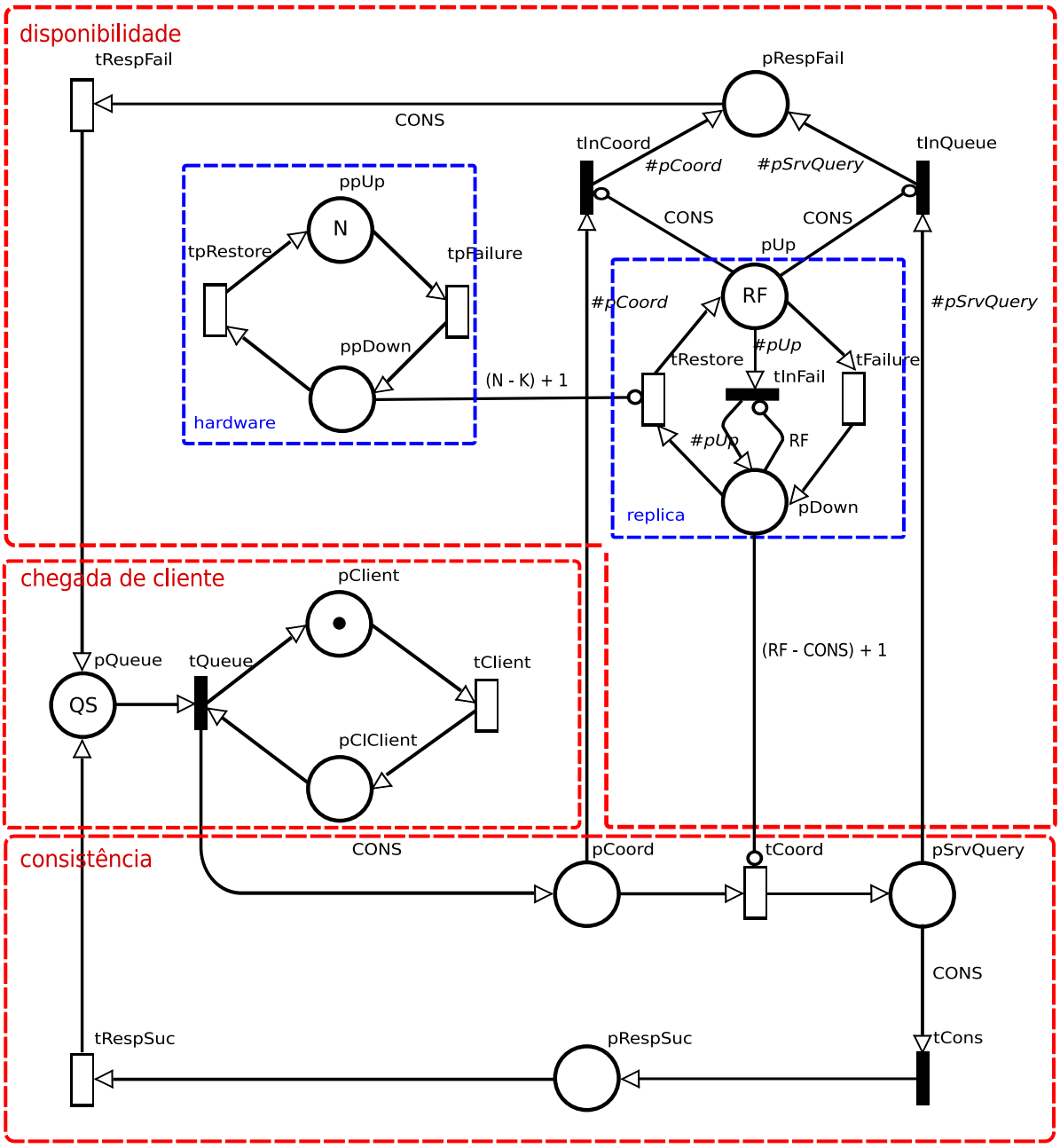
O sistema é composto por N servidores físicos para hospedar máquinas virtuais (VMs), e cada VM possui um SGBD em uma relação *um-para-um*. RF denota o número de réplicas do SGBD NoSQL e $CONS$ é o nível de consistência. Um servidor físico pode executar qualquer VM e, em caso de falha do servidor, as VMs migram para um servidor em funcionamento. Considera-se que a migração não tem impacto na disponibilidade. O sistema está operacional se K servidores e $CONS$ réplicas estiverem operacionais por meio das restrições ($K \leq N$) e ($CONS \leq RF$).

A Figura 12 apresenta o modelo concebido para estimar o desempenho e a disponibilidade do SGBD NoSQL baseado em quorum. Esse modelo é composto por três blocos: disponibilidade, chegada de cliente e consistência. A seguir, são apresentados os blocos dos modelos.

O bloco de disponibilidade contempla os servidores físicos ($hardware$) e réplicas do SGBD ($replica$). Os lugares $ppUP$ e pUP denotam o número de servidores físicos e réplicas que estão operacionais, respectivamente. Inicialmente, todos os servidores físicos (N) e réplicas (RF) estão funcionando.

As transições $tFailure$ e $tpFailure$ denotam as falhas, e seus parâmetros de tempo são

Figura 12 – Modelo de desempenho e disponibilidade



Fonte: Elaboração Própria.

fornecidos pelos respectivos MTTF, que é o valor médio que contempla todos os componentes do servidor que tendem a falhar. Um valor MTTF pode ser obtido utilizando, por exemplo, RBD (MACIEL et al., 2011). O lugar $ppDown$ representa a falha de um servidor e, da mesma forma, o lugar $pDown$ indica a falha de uma VM/réplica.

Para réplicas do SGBD, a transição imediata $tInFail$ tem a expressão de guarda ($\#ppUp < K$), que verifica se não existem K servidores físicos disponíveis. Se verdadeiro, todos os SGBDs falham, uma vez que a quantidade mínima de servidores físicos não está operacional. Além

disso, o arco inibidor do local $ppUp$ para $tRestore$ não permite a recuperação de réplicas de SGBD, a menos que (N - K) + 1 servidores físicos estejam disponíveis. O tempo de migração dos SGBDs nos servidores físicos disponíveis K não foi considerado.

$tRestore$ e $tpRestore$ representam a manutenção do sistema. A transição $tRestore$ representa a recuperação de uma réplica. Quando essa transição dispara, são adicionados *tokens* no lugar $pSrvDown$. Da mesma forma, quanto a transição $tpRestore$ é acionada, *tokens* são transferidos para o lugar $ppUp$.

Os parâmetros de tempo para as transições $tRestore$ e $tpRestore$ são indicados pelo MTTR, informado pelo projetista do sistema que pode ser influenciado, por exemplo, pela experiência da equipe de manutenção. As transições $tFailure$, $tpFailure$, $tRestore$ e $tpRestore$ adotam a semântica *infinity server*, ou seja, os servidores podem falhar e se recuperar de forma simultânea. O modelo também adota a variável $CONS$, representando um número inteiro que indica o nível de consistência requerido pela operação.

As transições imediatas $tInCoord$ e $tInQueue$ ativam o fluxo de uma operação que não é possível ser executada com sucesso, devido à falta de servidores para garantir o nível de consistência. Os arcos inibidores impedem que essas transições disparem, caso não seja possível atender ao nível de consistência da operação, direcionando os *tokens* para uma resposta de negação de acesso no lugar $pRespFail$.

O bloco de chegada do cliente representa as solicitações periódicas do usuário, indicadas pelo disparo da transição temporizada $tClient$. O valor de $tClient$ pode ser obtido, por exemplo, no *log* do sistema. Se houver espaço na fila ($QS > 0$ no lugar $pQueue$), um cliente será aceito pelo sistema, enviando *token* para o lugar $pCoord$.

É importante mencionar que o arco que liga $tQueue$ à $pCoord$ possui peso $CONS$; dessa forma, o lugar recebe $tokens$ igual ao nível de consistência, o que possibilita a representação do comportamento do servidor coordenador. O coordenador é o servidor do SGBD que recebe a requisição do cliente e realiza a operação conforme o nível de consistência requerido pelo cliente.

No bloco consistência, a transição $tCoord$ representa a comunicação do coordenador com os servidores que contêm réplicas do dado (que pode incluir o próprio coordenador) para executar uma operação. Conforme explicado no parágrafo anterior, essa transição efetuará disparos, conforme a quantidade de $tokens$ em $pCoord$, atribuído com peso igual a $CONS$. Um arco inibidor com peso (RF - $CONS$) + 1 garante que a transição $tCoord$ será habilitada apenas se houver servidores operacionais suficientes para atingir o nível de consistência. O valor

de $tCoord$ pode ser obtido no ambiente do SGBD.

Sempre que um servidor falha e o nível de consistência não pode ser garantido, uma resposta da falha é retornada (transição $tRespFail$). Se os servidores do banco de dados exigidos pela consistência adotada executarem a operação, uma resposta bem-sucedida será retornada (transição $tRespSuc$).

A transição imediata $tCons$ possui um arco de entrada com peso $CONS$; dessa forma, representa a confirmação das operações no nível de consistência determinado, retornando apenas uma resposta da operação. As transições $tRespFail$ e $tRespSuc$ têm como entrada o valor de *resp*; os valores de *resp* são fornecidos no sistema.

As transições do modelo são apresentadas na Tabela 4.

Tabela 4 – Atributos das transições do modelo de desempenho e disponibilidade

Transição	***Tipo***	***Atraso***	***Marcação***	***Concorrência***
$tFailure$	exponencial	MTTF	constante/dependente	$infinity\ server$
$tpFailure$	exponencial	MTTF	constante/dependente	$infinity\ server$
$tRestore$	exponencial	MTTR	constante/dependente	$infinity\ server$
$tpRestore$	exponencial	MTTR	constante/dependente	$infinity\ server$
$tInCoord$	imediata	-	dependente	-
$tInQueue$	imediata	-	dependente	-
$tClient$	exponencial	CL	constante/dependente	$single\ server$
$tQueue$	imediata	-	dependente	-
$tCoord$	exponencial	CC	constante/dependente	$infinity\ server$
$tCons$	imediata	-	dependente	-
$tRespFail$	exponencial	resp	constante/dependente	$infinity\ server$
$tRespSuc$	exponencial	resp	constante/dependente	$infinity\ server$

O modelo de desempenho e disponibilidade é adotado para estimar a probabilidade de o sistema estar disponível para atender as requisições para determinado nível de consistência e a latência entre operações bem-sucedidas. Usando o modelo proposto, a disponibilidade é fornecida por $A = P\{\#pUp \geq CONS\}$. A vazão é estimada como $throughput = P\{\#pRespSuc > 0\} * 1/resp$, em que $resp$ é o tempo associado à transição $tRespSuc$ (resposta bem-sucedida). Portanto, latência é $throughput^{-1}$.

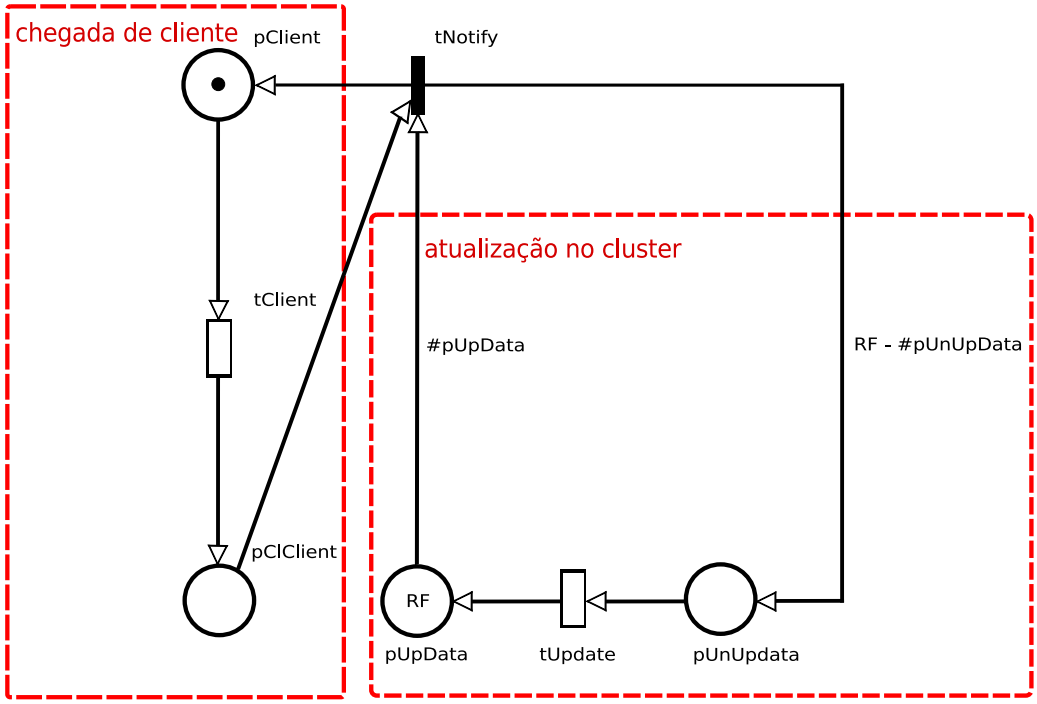
5.2 MODELO DE ACESSO AO DADO

O dado está consistente no SGBD NoSQL quando está atualizado em todos os servidores redundantes. Uma única operação de gravação torna todos os servidores temporariamente inconsistentes. Dessa forma, o modelo concebido considera a probabilidade de os servidores

possuírem a versão mais recente do dado, considerando a chegada de novas atualizações (ver Figura 13).

A falha de um servidor não é considerada neste modelo, que possui dois blocos, chegada de cliente e atualização do *cluster*. O modelo adota duas constantes: (i) RF que denota o número de réplicas do banco de dados; e (ii) $CONS$ que representa o nível de consistência (quorum) da operação de leitura. Além disso, dois blocos de construção são considerados: a chegada do cliente e a atualização do cluster. As transições do modelo são apresentadas na Tabela 5 e os blocos são representados a seguir.

Figura 13 – Modelo de probabilidade de consistência NoSQL



Fonte: Elaboração Própria.

O bloco de chegada do cliente representa as solicitações de gravação periódicas para o *cluster* NoSQL, e o disparo da transição $tClient$ indica a chegada de uma solicitação. O bloco de atualização do cluster descreve a execução de operações de gravação em cada servidor. O lugar $pUpData$ representa os nós com os dados mais recentes e sua marcação inicial é igual a RF.

Sempre que uma solicitação está pendente ($\#pClClient > 0$) e há servidores que não

estão executando uma operação de gravação ($\#pUpData > 0$), a transição $tNotify$ é acionada. A marcação do local $pUnUpData$ reflete os servidores sem os dados mais recentes. Além disso, a execução real das operações de gravação é representada pela transição temporizada $tUpdate$, que adota a semântica de servidor infinito.

Tabela 5 – Atributos das transições do modelo de acesso ao dado

Transição	***Tipo***	***Atraso***	***Marcação***	***Concorrência***
$tClient$	exponencial	UPDATE	constante/dependente	$single\ server$
$tNotify$	imediata	-	dependente	$single\ server$
$tUpdate$	exponencial	WRITE	constante/dependente	$infinity\ server$

A abstração do modelo considera sempre quando o sistema possui servidores que não realizaram solicitações de gravação anteriores ($\#pUnUpdata > 0$) e uma nova solicitação de gravação chega ($\#pClClient > 0$); os servidores com solicitações pendentes também irão contemplar a nova solicitação durante uma atualização (transição $tUpdate$). Quando uma nova atualização de dados é realizada ($\#pClient > 0$), qualquer atualização em andamento (ainda sendo replicada) torna-se obsoleta. Essa abstração garante que o estado de consistência seja sempre obtido com os dados mais recentes em todas as réplicas.

$$P_s = E\{\#pUpData\}/RF \tag{5.1}$$

$$P_{nd} = 1 - (1 - P_s)^{CONS} \tag{5.2}$$

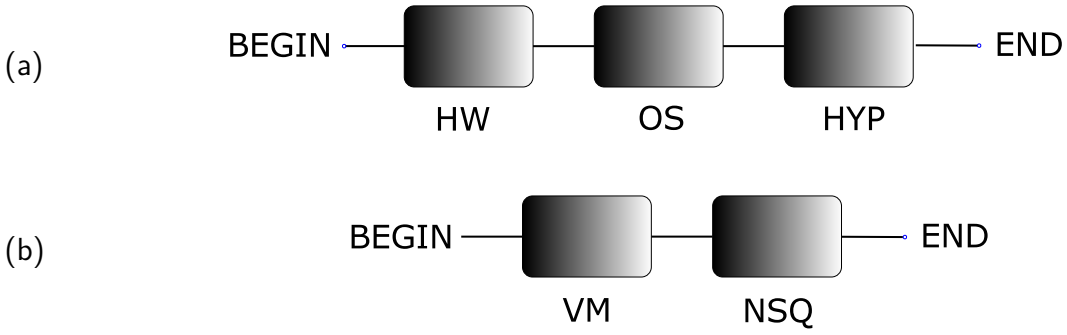
A Equação 5.1 apresenta a probabilidade de um único servidor conter os dados mais recentes, isto é, a probabilidade de um nível de consistência igual a 1 ($CONS = 1$). Para $1 < CONS \leq RF$, a probabilidade de acessar os dados mais novos (P_{nd}) é a probabilidade de acessar pelo menos um servidor atualizado no quorum (Equação 5.2).

5.3 MODELO RBD

A modelagem através de RBD foi realizada por considerar de forma independente os componentes do sistema, para o seu funcionamento. Dessa forma, permitiu-se a análise dos componentes da infraestrutura do sistema na avaliação da disponibilidade dos SGBDs NoSQL baseados em quorum.

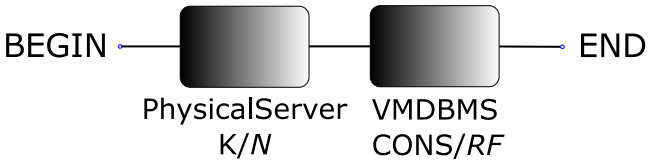
As Figuras 14 (a) e (b) representam, respectivamente, os modelos RBD para o *hardware* e a réplica dos SGBDs. Ambos os modelos adotam arranjo em série, em que o sistema estará

Figura 14 – Modelos RBD: (a) *hardware*; e (b) réplicas do SGBD



Fonte: Elaboração Própria.

Figura 15 – Modelos RBD



Fonte: Elaboração Própria.

disponível se todos os componentes estiverem operacionais. Assumindo N componentes, a disponibilidade/confiabilidade do sistema (P_s) é estimada na Equação 5.3, em que p_i denota a disponibilidade/confiabilidade do componente i.

$$P_s = \prod_{i=1}^{N} p_i \tag{5.3}$$

O servidor físico (Figura 14 (a)) é composto de *hardware* (HW), sistema operacional (SO) e *hipervisor* (HYP). O modelo de réplica do SGBD (Figura 14 (b)) contempla a máquina virtual (VM) e o SGBD NoSQL (NSQ). Cada componente é um bloco e, a partir dos respectivos MTTF e MTTR, obtém-se a disponibilidade do componente.

Utilizando as equações descritas na Seção 2.4, o modelo também pode ser reduzido a um único bloco com um MTTF e MTTR aproximados (BUZACOTT, 1967). Para a Figura 14 (a), o bloco reduzido é denominado $PhysicalServer$ e sua disponibilidade é $A_{PhysicalServer}$. Da mesma forma, para a Figura 14 (b), o bloco único é denominado $VMDBMS$ e sua disponibilidade é A_{VMDBMS}.

Para obter o modelo final do sistema (Figura 15), é necessário um arranjo $K\text{-}out\text{-}of\text{-}N$. Para este caso, a disponibilidade/confiabilidade é estimada na Equação 5.4. p denota a

disponibilidade/confiabilidade de um componente.

$$P_{kon}(k,n,p) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{5.4}$$

Uma vez que K de N servidores físicos e $CONS$ de RF réplicas de banco de dados precisam estar operacionais, a disponibilidade do sistema (A_{sys}) é obtida na Equação 5.5:

$$A_{sys} = P_{kon}(K,N,A_{PhysicalServer}) \times P_{kon}(CONS,RF,A_{VMDBMS}) \tag{5.5}$$

5.4 MODELO DE CONSUMO DE ENERGIA

O modelo de consumo de energia é embasado na consistência do SGBD NoSQL baseado em quorum (Seção 2.3), que contempla o consumo de energia (E) relacionado à soma das operações de leitura e escrita em um dado intervalo de tempo T:

$$E = [(P_{write} \times C_{write} \times RF) + (P_{read} \times C_{read} \times CONS)] \times T \tag{5.6}$$

A Equação 5.6 leva em consideração a proporção ($C_{write} + C_{read} = 1$) das operações de escrita (C_{write}) e leitura (C_{read}), em um período de tempo (T). P_{write} denota o consumo de energia para operação de escrita em um único servidor; portanto, este valor é multiplicado pelo fator de replicação (RF) e C_{write}, que é realizado em cada réplica. Da mesma forma, P_{read} é o consumo de energia para operação de leitura em um único servidor; o valor é multiplicado pelo nível de consistência ($CONS$) e C_{read}. Para leitura, adota-se $CONS$, pois nem todos os servidores participam de uma operação de leitura.

P_{read} e P_{write} podem ser obtidos por medição ou com base na experiência do arquiteto do sistema. Neste trabalho, utilizamos os valores de (GOMES; TAVARES; JUNIOR, 2016). A proporção pode ser verificada ou estimada usando-se contadores para leitura e gravação. Esses contadores são comumente fornecidos pelo SGBD, a exemplo do Cassandra NoSQL (DATASTAX, 2022).

5.5 CONSIDERAÇÕES FINAIS

Este capítulo inicialmente apresentou os modelos SPN propostos para avaliação do desempenho, da disponibilidade e do acesso ao dado atualizado em ambiente NoSQL. Foi apresen-

tado, primeiramente, o modelo de desempenho e disponibilidade e, logo após, o modelo de probabilidade da consistência. Esses modelos proporcionam ao projetista da solução observar o custo-benefício entre a prioridade das métricas de desempenho, disponibilidade e probabilidade da consistência.

Em seguida, é proposta a modelagem que utiliza RBD o que permite o aperfeiçoamento da análise de disponibilidade, considerando componentes isolados da infraestrutura do serviço avaliado. Por fim, apresenta-se um modelo para estimativa do consumo de energia. Todos os modelos consideram as características principais do SGBD NoSQL baseado em quorum, o que permite a avaliação em conjunto de todas as métricas apresentadas neste capítulo.

6 RESULTADOS EXPERIMENTAIS

"A coisa mais bela que o homem pode experimentar é o mistério. É essa emoção fundamental que está na raiz de toda ciência e toda arte."

— Walter S. Landor

Este capítulo apresenta os experimentos computacionais conduzidos para demonstrar a viabilidade prática das técnicas de modelagem propostas. Inicialmente, são apresentados os experimentos realizados para validar nossos modelos. Em seguida, adotamos uma abordagem de DoE (MONTGOMERY, 2017) para demonstrar a influência de cada fator (por exemplo, nível de consistência e número de servidores redundantes) nos modelos. Por fim, os modelos são avaliados para mostrar como nossa proposta pode ajudar o arquiteto de uma solução NoSQL baseado em quorum.

6.1 VALIDAÇÃO DOS MODELOS PROPOSTOS

Esta seção apresenta os experimentos conduzidos para validar os modelos propostos de desempenho, disponibilidade e consistência NoSQL baseado em quorum. Adicionalmente os experimentos são utilizados para estimar os valores iniciais do DoE. No primeiro modelo, a métrica de desempenho validada corresponde ao tempo de uma operação.

A métrica de disponibilidade representa a probabilidade do sistema encontrar-se em estado operacional, ou seja, em funcionamento. No segundo modelo, valida-se a proporção dos dados acessados que estão consistentes, ou seja, possuem uma mesma versão nos servidores redundantes. Em seguida, aplica-se outra técnica de validação no modelo RBD, o qual é concebido para análise da disponibilidade nos sistemas NoSQL baseados em quorum.

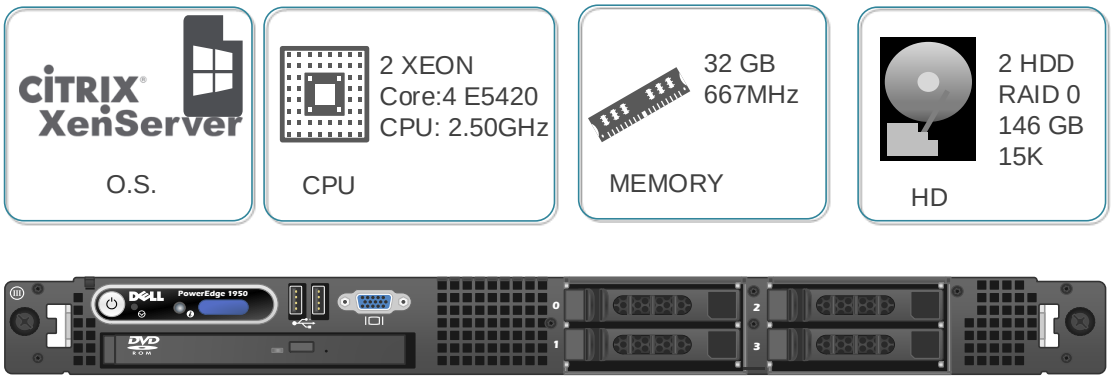
Nossos modelos foram validados considerando a plataforma XenServer e máquinas virtuais VM, para o cliente e SGBDs NoSQL. Cada VM utiliza o sistema operacional Debian 9.0 e CPU *dual core*. A VM do cliente possui 4 GB de RAM e cada SGBD possui 8 GB. Nas validações, foram utilizados o SGBD Cassandra NoSQL (HAUGHIAN; OSMAN; KNOTTENBELT, 2016) e o Riak KV (BARON et al., 2016). A Tabela 6 apresenta a configuração de cada VM utilizada.

A Figura 16 apresenta a configuração (*setup*) do *hardware* utilizado na avaliação dos sistemas. O ambiente foi virtualizado em um servidor dedicado DELL 1950 que contém 2 processadores E5420GHZ, 32 GB de memória RAM, dois discos rígidos de 146 GB e 15.000

Tabela 6 – Configurações das máquinas virtuais

Plataforma	SO	Máquina	RAM	CPU	SGBD	*benchmark*
XenServer	Debian	cliente	4GB	*dualcore*	-	YCSB, *script*
XenServer	Debian	SGBD 01	8GB	*dualcore*	Cassandra, Riak KV	
XenServer	Debian	SGBD 02	8GB	*dualcore*	Cassandra, Riak KV	

Figura 16 – Setup



Fonte: Elaboração Própria.

rotações configurados em RAID 0 para melhor desempenho. O RAID 0 foi utilizado para que o ambiente de experimentação obtivesse maior *throughput*. Essa configuração não é indicada para utilização em sistemas corporativos, pois não tolera falhas no disco. O sistema operacional do *hardware* e ferramenta de virtualização utilizada foi o Xen Server.

Para a métrica de desempenho no modelo de desempenho e disponibilidade, considerou-se, na validação, o nível de consistência 1 para o SGBD Cassandra e nível de consistência 2 para Riak KV. Esses níveis representam a configuração padrão desses SGBDs. Os ambientes foram validados com duas réplicas do SGBD. Isto foi necessário porque o sistema se comporta de maneira diferente quando não é replicado, não sendo, pois, necessário gerenciar a replicação. Além disso, na validação dessa métrica não se contemplaram falhas nos servidores referentes as operações de leitura consideradas. A escolha da operação de leitura foi realizada em razão de essa ser uma operação mais sensível a variações de resultados que a escrita, então validou-se

em um cenário mais difícil. Para a métrica de disponibilidade, utilizou-se apenas um *nó* do SGBD com toda a sua infraestrutura de suporte. Essa escolha foi realizada para reduzirmos o tempo da avaliação.

Para o modelo de probabilidade, é utilizada a consistência nível 1 com 2 réplicas do SGBD, uma vez que nessa configuração podem ser acessadas informações consistentes ou não consistentes nas réplicas do SGBD. Na validação do modelo que considera os componentes da infraestrutura do sistema para a disponibilidade, utiliza-se o mesmo experimento e técnica de validação da métrica disponibilidade do primeiro modelo. Esse aproveitamento foi possível porque o modelo a ser validado tem as mesmas características principais dos modelos e métricas avaliadas.

Para cada validação realizada, a hipótese nula H_0 afirma que o resultado do modelo M_r é igual ao sistema S_r em um intervalo de confiança de 95%, ou seja, para H_0 no nível de significância $\alpha = 0,05$ dar-se (M_r - $S_r = 0$).

6.1.1 Validação do desempenho

A validação do modelo de desempenho e disponibilidade utilizou o Yahoo! Cloud Serving Benchmark (YCSB) (COOPER, 2019); a Tabela 7 apresenta os seguintes valores utilizados na ferramenta, onde:

- (i) *data size* é o tamanho de cada registro utilizado no experimento. Utilizou-se o valor padrão da ferramenta por ser uma abordagem comum em pesquisas correlatas;
- (ii) *request distribution* é a distribuição de probabilidade utilizada para a seleção da leitura dos dados na ferramenta. Utilizamos a distribuição $uniform$ devido à aleatoriedade de acesso aos dados no SGBD;
- (iii) *record count* é o número de registros já contidos no banco de dados. Foi selecionado um valor no qual a leitura de dados contidos no disco do servidor fosse mais provável, com possibilidade de eventual acesso a dados na memória RAM;
- (iv) *operation count* é o número de operações que o YCSB realiza em cada execução. Foi selecionado um número de execuções suficiente para a avaliação do estado estacionário do sistema;

- (v) *read all fields* é uma variável *booleana* que permite a leitura de todos os campos do registro. Esta opção foi mantida por razões de acompanhamento às pesquisas correlatas.

Ao final de cada operação do YCSB, o ambiente foi reiniciado para que resíduos da execução anterior (ex. *cache*) não interfirissem na execução subsequente.

Tabela 7 – Parâmetros YCSB

Parâmetro	Valor
$data\ size$	1KB $(default)$
$request\ distribution$	$uniform$
$record\ count$	200000
$operation\ count$	30000
$read\ all\ fields$	$true$ $(default)$

O tempo de chegadas de clientes foi de $12,4ms$ nos experimentos realizados com o Cassandra e $11,8ms$ nos experimentos realizados com o Riak. 30 amostras foram obtidas em cada servidor de banco de dados. Tal tamanho da amostra está relacionado ao teorema do limite central (MONTGOMERY; RUNGER, 2013), considerando que estamos preocupados com valores médios. O tempo para cada transição temporizada no modelo de desempenho foi obtido com base nas medições na plataforma considerando essas amostras. A Tabela 8 e a Tabela 9 apresentam os tempos obtidos. Finalmente, o modelo foi avaliado usando-se análise estacionária (MARUSSY et al., 2016) devido às propriedades *Boundedness*, *Home state* e *Liveness* dessa rede de Petri. (ver Seção 2.8).

A latência obtida entre as respostas bem-sucedidas nos experimentos realizados com o SGBD Cassandra foi de $18,454ms$. Esse valor está contido no intervalo de confiança [$18,391$, $19,315$] com nível de significância $\alpha = 0,05$, ou seja, intervalo de confiança de 95%, obtido com as medidas na plataforma real. Para o Riak KV, a latência foi estimada em $13,928ms$, com valores contidos dentro do intervalo de confiança de 95% [$13,912$, $14,232$]. Portanto, não há evidência para rejeitarmos H_0, ou seja, não há evidência que sugira que o modelo apresenta resultado diferente do sistema no nível de confiança especificado.

Tabela 8 – Tempo das transições para a validação do desempenho com o SGBD Cassandra

Transição	Tempo (ms)
$tClient$	12,4
$tCoord$	10,24
$tRespSuc$	2,6

Tabela 9 – Tempo das transições para a validação do desempenho com o SGBD Riak KV

Transição	Tempo (ms)
$tClient$	11,8
$tCoord$	3,49
$tRespSuc$	1,8

6.1.2 Validação da disponibilidade

A validação da disponibilidade foi realizada para o modelo SPN (ver Figura 12) e para o modelo RBD; para este último, consideramos o modelo base (ver Figura 14(a)). Dessa forma, foi possível comparar o modelo com um sistema real. Foram utilizadas técnicas de injeção de falhas em um sistema em funcionamento para observação de sua disponibilidade. Considerou-se uma infraestrutura sem redundância em cada um de seus seus componentes, e foi utilizada uma instância do SGBD Cassandra. Esse sistema equivale a um arranjo em série no formalismo RBD. No modelo SPN, corresponde às variáveis [k, n, rf e $cons$] = 1.

A Tabela 10 apresenta os valores de MTTF e MTTR utilizados nessa validação. As colunas $MTTF_{reduzido}$ e $MTTR_{reduzido}$ representam os valores simplificados com os respectivos $MTTF$ e $MTTR$ para a avaliação da disponibilidade no modelo SPN, que não permite a avaliação com a mesma profundidade do modelo RBD concebido. Para simulação da avaliação no tempo de um ano, os valores foram acelerados em 100 vezes. Ademais, para este propósito de validação, foi assumida a distribuição exponencial nos valores.

Tabela 10 – MTTF e MTTR dos componentes do modelo

Componente	$MTTF$ (h)	$MTTF_{reduzido}$	$MTTR$ (h)	$MTTR_{reduzido}$
HW	8760		1,67	
OS	1440	481,46	1	1,03
HYP	788,4		1	
.....				
VM	2880	1439,99	0,17	0,17
NSQ	2880		0,17	

Para inserir falhas correspondentes ao não provisionamento dos serviços modelados nessa pesquisa, desenvolvemos uma ferramenta de injeção de falhas por meio de um *script*; outro *script* foi desenvolvido para monitorar o provisionamento do serviço avaliado. Essas ferramentas foram utilizadas na validação do modelo de disponibilidade desenvolvido e os *scripts* foram

desenvolvidos na linguagem *Python* a qual possui todas as bibliotecas necessárias para os cálculos, conexões e instruções necessários ao sistema alvo.

```
   Data: block_mttf[], block_mttr[], ttf, ttr
   Result: system restore or repair.
 1 while True do
 2 |   generate time to failure (ttf);
 3 |   sleep(ttf);
 4 |   interrupt system;
 5 |   generate time to repair (ttr);
 6 |   sleep(ttr);
 7 |   restore system;
 8 end
```

Algoritmo 1: Injeção de Falhas

```
   Data: status
   Result: log register status.
 1 while True do
 2 |   if dbms connect then
 3 |   |   if status != up then
 4 |   |   |   status = up;
 5 |   |   |   register repair in log;
 6 |   |   end
 7 |   end
 8 |   else if dbms does not connect then
 9 |   |   if status == up then
10 |   |   |   status = down;
11 |   |   |   register failure in log;
12 |   |   end
13 |   end
14 end
```

Algoritmo 2: Monitoramento do sistema

O Algoritmo 1 demonstra os passos realizados pela ferramenta para inserir uma falha e recuperar o sistema. As ações do algoritmo são baseadas em duas variáveis: *ttf*, gerado considerando uma distribuição exponencial e o valor base do MTTF utilizado no modelo a ser verificado; e o *ttr* que considera o MTTR do modelo avaliado para gerar o próximo intervalo de reparo do sistema, por meio da mesma distribuição.

Para minimizar o atraso entre as ações de falha e reparo no sistema, essa ferramenta atua diretamente na *interface* de rede do sistema nos respectivos eventos de falha e reparo. Dessa forma, sua execução é realizada a partir do *host* do serviço avaliado. O algoritmo foi desenvolvido para inserção de falhas em cada componente de forma independente. Dessa forma, cada componente do sistema possui uma *thread* específica para as falhas e reparos.

O Algoritmo 2 apresenta as etapas do monitoramento do sistema avaliado. Por meio da variável $status$ a ferramenta registra em arquivo a mudança de estado do sistema que pode ser disponível ou indisponível. Essa verificação da disponibilidade do sistema é realizada por conexão ao banco de dados. Para que se obtenha o mínimo atraso na percepção de mudança do estado, a ferramenta não utiliza *delay* adicional entre as iterações, ex. ($sleep()$). Considerou-se um fator de aceleração de 100 vezes e os experimentos no sistema real foram executados em um período de 97,3 horas.

Para o cálculo do intervalo de confiança de 95% da disponibilidade do sistema utilizamos uma técnica descrita por Keese (ver Seção 2.11.2), que forneceu o intervalo: [$0,997319$, $0,998306$]. Utilizando a ferramenta Mercury, o modelo RBD estimou a disponibilidade em $0,997732$. Este valor está contido no intervalo de confiança 95%. O modelo SPN forneceu o valor de $0,997395$. Como ambos os valores fornecidos pelos modelos RBD e SPN, respectivamente, estão contidos no intervalo de confiança, não há evidência para rejeitarmos H_0, ou seja, não há evidência estatística para sugerir que esses modelos apresentam resultados diferentes do sistema no nível de confiança especificado.

6.1.3 Validação do modelo de probabilidade da consistência

Para este modelo, o YCSB não é adequado, uma vez que se trata de um *framework* desenvolvido para avaliar o desempenho de SGBD NoSQL. O YCSB não avalia a consistência dos dados acessados (TAPEKHIN; BOGOMOLOV; VELIKANOV, 2019). Dessa forma, o modelo de probabilidade da consistência foi validado em experimentos que utilizaram um *script* baseado em duas *threads*: (i) a primeira executou escritas com intervalo definido para as réplicas do SGBD; e (ii) a segunda realizou leituras simultâneas em cada réplica para avaliar a probabilidade de acessar os dados mais recentes.

A chegada periódica de atualizações foi definida como $100ms$, valor que facilitou o controle dessas atualizações no *script* desenvolvido com essa finalidade. Semelhante à validação do modelo anterior, 30 amostras foram coletadas, número suficiente para validar o modelo no intervalo de confiança de 95%. Os valores médios para cada operação representada pelas transições temporizadas no modelo SPN foram obtidos (ver a Tabela 11) para representar a chegada de atualizações $tClient$ e o tempo de escrita $tUpdate$ do servidor NoSQL baseado em quorum.

Uma análise estacionária foi realizada no modelo para obtenção do valor da métrica que

considera a probabilidade de o dado estar atualizado em todos os servidores redundantes e, como resultado, obtive-se o valor de $0,974$ para essa métrica de probabilidade. As medidas na plataforma do sistema realizadas por experimentos forneceram como resultado o intervalo de confiança 95% $[0,974,\ 0,976]$ para o nível de significância $\alpha = 0,05$ da mesma métrica. Portanto, não há evidência estatística para rejeitarmos H_0, isso significa que não há evidência para sugerir que o modelo apresenta resultado diferente do sistema no nível de confiança especificado.

Tabela 11 – Tempos de transição para validação do modelo de probabilidade da consistência

Transição	Tempo (ms)
$tClient$	100
$tUpdate$	2,639

6.2 EXPERIMENTOS

Esta seção demonstra a aplicabilidade dos modelos propostos os quais são analisados usando-se abordagem de projetos de experimentos DoE (MONTGOMERY, 2017), por meio de um projeto fatorial l^k. Especificamente, os resultados apresentam para cada modelo a classificação dos efeitos diante das interações com os fatores adotados para cada métrica e com os resultados dos tratamentos obtidos no projeto de experimento.

As métricas de interesse são a latência da operação, a disponibilidade do sistema, a probabilidade da consistência e o consumo de energia. Tais métricas são importantes, uma vez que podem determinar a viabilidade da configuração de sistemas NoSQL baseados em quorum. Ademais, demonstra-se que essas métricas podem ser conflitantes, implicando minuciosa análise para verificar o melhor equilíbrio em relação ao objetivo do sistema.

6.2.1 Desempenho

Este projeto de experimento visa avaliar o tempo de uma operação em diferentes tratamentos. O experimento de desempenho considera: 4 fatores ($k = 4$) com 2 níveis ($l = 2$); e dados como: (i) nível de consistência ($cons$) - 1, 3; (ii) tempo resposta bem-sucedida ($resp$) - $0,9$, $1,8ms$; (iii) comunicação do coordenador (cc) - $1,7$, $3,4ms$; e (iv) fator de replicação dos dados (rf) - $3,5$.

A influência da disponibilidade não foi considerada nessas experiências, dessa forma, para o servidor físico, os valores de $MTTF$ e $MTTR$ são $1115h$ e $0,47h$, respectivamente. Para a réplica do SGBD, os valores são $1440h$ para $MTTF$ e $0,5h$ para $MTTR$. Também foi constante o tempo entre as solicitações dos clientes com valor de $11,8ms$, uma vez que a avaliação dos componentes do SGBD foram priorizados na análise. Para a escolha dos níveis, utilizou-se a validação do Riak com redução de 50% dos valores no nível seguinte. Essa redução representa uma melhor capacidade de *hardware*. O fator ($cons$) nível de consistência representa a quantidade de servidores que confirmam o sucesso de uma operação antes da resposta ao cliente.

Os níveis selecionados para ($cons$) permitem a avaliação da consistência eventual e forte. Para (rf), 3 réplicas atendem ao nível mais alto para ($cons$); e 5 réplicas permitem uma melhor avaliação dos efeitos em relação a mais recursos disponíveis. Os fatores ($resp$) resposta bem-sucedida e (cc) comunicação do coordenador são obtidos, respectivamente, do servidor secundário e do coordenador da requisição (servidor que recebe a requisição do cliente). Por fim, o fator de replicação (rf) é o número de servidores que contém uma réplica dos dados.

Após a execução do projeto de experimentos de desempenho, obteve-se a classificação dos efeitos principal e interações, por exemplo, ($cc \times cons$) dos fatores adotados na latência bem-sucedida (Tabela 12) e (Figura 17). Efeito é a mudança na resposta devido a uma alteração no nível do fator. A classificação é ordenada de forma decrescente, considerando os valores absolutos de todos os efeitos. Além disso, a Tabela 12 considera os efeitos principais e as interações de segunda ordem, já que as interações de ordem superior geralmente são desprezíveis (MONTGOMERY, 2017).

Tabela 12 – Classificação dos efeitos principais e de interação (latência)

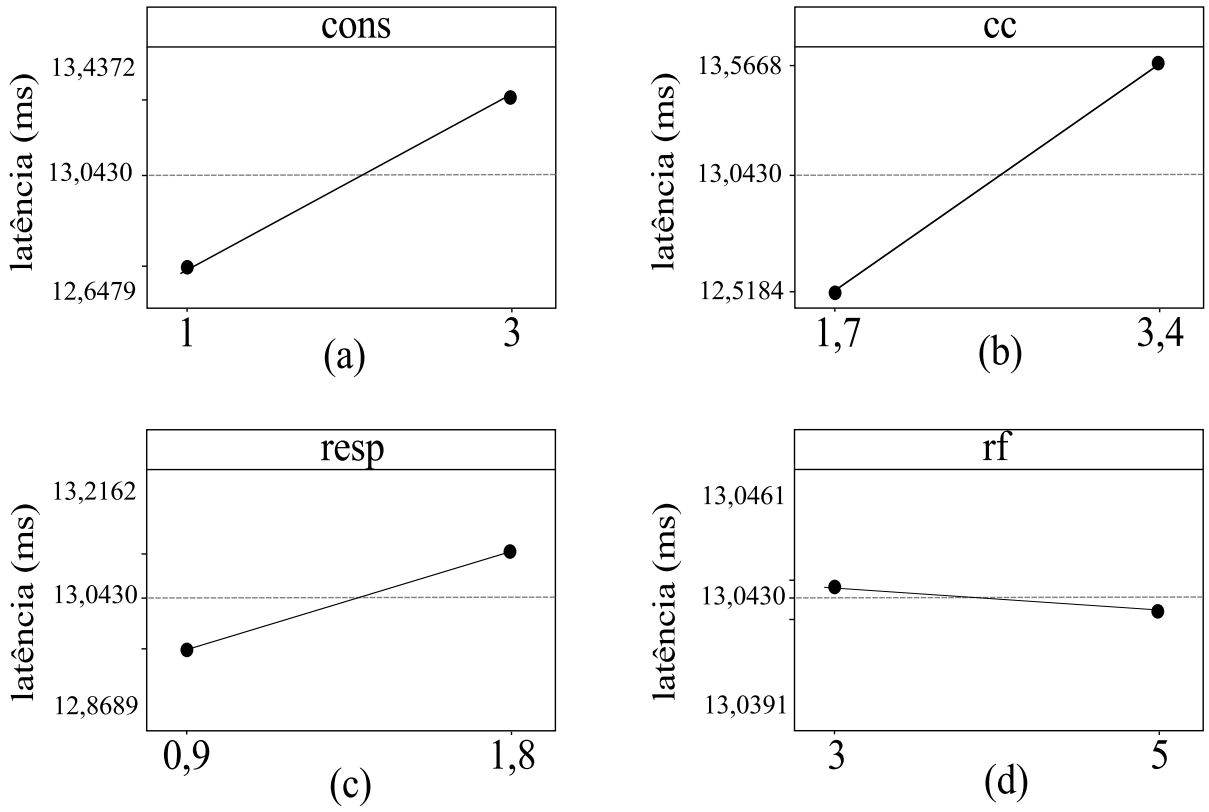
Fator/Interações	|Efeito| (ms)
cc	1,048
$cons$	0,789
$cc \times cons$	0,385
$resp$	0,347
$cc \times resp$	0,047
$cons \times resp$	0,046

A comunicação do coordenador (cc) tem o maior efeito. O nível de consistência ($cons$) aparece com o segundo maior efeito e também tem uma contribuição importante para o desempenho. Isso é explicado devido à quantidade necessária de confirmações dos servidores

para uma operação que tem uma influência proeminente no desempenho do sistema. Devido à importância de (cc) e $(cons)$, sua interação $(cc \times cons)$ tem o maior efeito entre todas as interações.

O fator de replicação (rf) e suas interações não desempenharam um papel significativo na determinação do desempenho do sistema. Isso é explicado porque, apesar de (rf) indicar o número de réplicas dos dados, o número de servidores redundantes para os parâmetros de sistema utilizados não apresentaram significativa influência em relação aos outros fatores.

Figura 17 – Gráficos de efeito – latência



Fonte: Elaboração Própria.

A Tabela 13 apresenta a latência bem-sucedida de cada tratamento do projeto de experimento, considerando uma ordem crescente. Os valores mais baixos estão associados à consistência 1, pois são necessárias menos confirmações das operações entre os servidores para sua execução. Para melhor visualização, a Figura 17 mostra gráficos de efeitos que indicam a mudança de desempenho para cada fator e seus níveis.

Tabela 13 – Resultados do projeto de experimentos para latência

$cons$	cc (ms)	$resp$ (ms)	rf	latência (ms)
1	1,7	0,9	3	12,183
1	1,7	0,9	5	12,183
1	1,7	1,8	3	12,448
1	1,7	1,8	5	12,448
3	1,7	0,9	5	12,546
3	1,7	0,9	3	12,559
1	3,4	0,9	3	12,811
1	3,4	0,9	5	12,811
3	1,7	1,8	5	12,881
3	1,7	1,8	3	12,895
1	3,4	1,8	3	13,148
1	3,4	1,8	5	13,148
3	3,4	0,9	5	13,920
3	3,4	0,9	3	13,935
3	3,4	1,8	3	14,386
3	3,4	1,8	5	14,371

6.2.2 Disponibilidade

Este projeto de experimento visa avaliar a disponibilidade do sistema em diferentes tratamentos. Essa métrica pode ser alcançada por nosso modelo SPN de desempenho e disponibilidade ou por meio do modelo RBD concebido. Para demonstrar a possibilidade de uma avaliação da disponibilidade considerando os componentes da infraestrutura do NoSQL baseado em quorum considerou-se o modelo RBD.

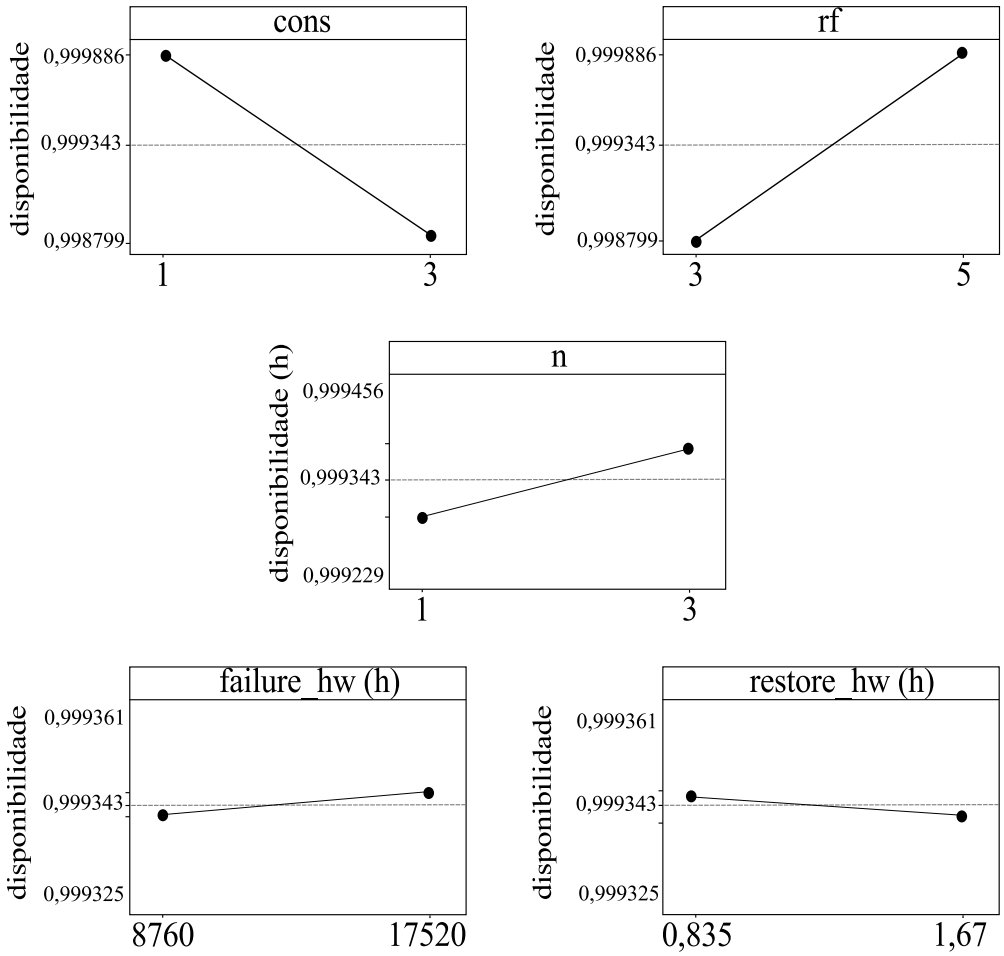
O experimento de disponibilidade considera os seguintes fatores ($k = 5$) e níveis ($l = 2$): (i) nível de consistência ($cons$) - 1, 3; (ii) fator de replicação (rf) - 3, 5; (iii) servidor físico (n) - 1, 3; (iv) *hardware* MTTF ($failure_{hw}$) - 8760h, 17520h; e (v) *hardware* MTTR ($restore_{hw}$) - 0.835h, 1.67h. $failure_{hw}$ e $restore_{hw}$ representam o *hardware* do servidor físico, e esses valores podem variar devido a melhores componentes ou habilidades de manutenção da equipe. ($cons$) e (rf) são iguais aos utilizados no experimento de desempenho. A Tabela 14 apresenta os valores fixos que foram adotados nos demais componentes do sistema.

Os resultados dos experimentos de disponibilidade são apresentados com base na análise do projeto de experimentos para disponibilidade (A) e tempo de inatividade *downtime* - (DT). Esse último é utilizado para apresentar a classificação dos efeitos, pois o tempo de inatividade

Tabela 14 – MTTF e MTTR dos experimentos

Componente	$MTTF$	$MTTR$
OS	2893	0,25
HYP	5000	0,17
VM	2880	0,17
NSQ	3000	2,0

Figura 18 – Gráficos de efeito – disponibilidade



Fonte: Elaboração Própria.

fornece melhor visualização para pequenos valores relacionados à variação da disponibilidade. Assumindo-se um ano como um período de tempo, o tempo de inatividade (em horas) é dado por $DT = (1 - A) \times 8760$.

A Tabela 15 apresenta a classificação dos efeitos principais e suas interações. Os fatores $(cons)$, (rf) e sua interação são os responsáveis pela maior variação de disponibilidade, valores superiores a 9 horas por ano, considerando os níveis adotados. Embora 9 horas pareçam

pequenas em comparação com 8.760 horas de um ano, elas podem gerar multas por violações de acordos de qualidade de serviço ou perda de usuários.

O número de servidores físicos, apesar do alto impacto financeiro na implementação de componentes redundantes (aquisição, manutenção, depreciação) tem um impacto de quase 2 horas, o que é um efeito significativo na disponibilidade. No entanto, é somente (25,7%) do impacto, por exemplo, do nível de consistência ($cons$). Para os níveis adotados, *hardware* $MTTF$ ($failure_{hw}$) e $MTTR$ ($restore_{hw}$) têm uma contribuição menor, cerca de 30 minutos por ano. É importante enfatizar que esses últimos fatores são apenas um dos componentes do bloco físico do sistema.

Tabela 15 – Classificação dos efeitos principais e de interação (*downtime*)

Fator/interação	$\|Efeito\ (h)$
$cons$	9,52
rf	-9.52
$cons \times rf$	9,52
n	1,98
$restore_{hw}$	-0,31
$failure_{hw}$	0,31
$n \times failure_{hw}$	-0,31
$n \times restore_{hw}$	0,31
$failure_{hw} \times restore_{hw}$	0,10

A Figura 18 apresenta gráficos de efeito que indicam a mudança de disponibilidade para os fatores e seus níveis. Para ($cons$) o tempo de inatividade pode variar de cerca de uma hora por ano, respectivamente, no (nível 1) a $10,52h$ (nível 3)(ver Figura 18(a)). (rf) apresenta o mesmo comportamento para seus níveis 3 e 5 (ver Figura 18(b)). $failure_{hw}$ e $failure_{hw}$ apresentam os mesmos valores, no entanto, em diferentes direções (ver Figura 18(d) e (e)).

A Tabela 16 apresenta os resultados de cada tratamento. Os valores de tempo de inatividade inferiores a um minuto são representados como $< 0,01$. A disponibilidade é consideravelmente reduzida quando a consistência forte ($cons = r$) é considerada, e nenhuma máquina redundante é acessível para substituição de um nó com falha. Por outro lado, máquinas físicas adicionais melhoram a disponibilidade. Os únicos tratamentos com menor disponibilidade são aqueles que adotaram consistência forte.

Tabela 16 – Resultados do projeto de experimento para disponibilidade

$cons$	rf	n	$failure_{hw}$ (h)	$restore_{hw}$ (h)	disponibilidade	*downtime* (h)
1	5	3	17520	0,835	0,999999999	$<$ 0,01
1	5	3	17520	1,67	0,999999999	$<$ 0,01
1	5	3	8760	0,835	0,999999999	$<$ 0,01
1	5	3	8760	1,67	0,999999999	$<$ 0,01
1	3	3	17520	0,835	0,999999999	$<$ 0,01
1	3	3	17520	1,67	0,999999999	$<$ 0,01
1	3	3	8760	0,835	0,999999999	$<$ 0,01
1	3	3	8760	1,67	0,999999999	$<$ 0,01
3	5	3	17520	0,835	0,999999996	$<$ 0,01
3	5	3	17520	1,67	0,999999996	$<$ 0,01
3	5	3	8760	0,835	0,999999996	$<$ 0,01
3	5	3	8760	1,67	0,999999996	$<$ 0,01
1	5	1	17520	0,835	0,999831944	1,4721
1	3	1	17520	0,835	0,999831943	1,4721
3	5	1	17520	0,835	0,999831940	1,4722
1	5	1	17520	1,67	0,999784297	1,8895
1	5	1	8760	0,835	0,999784297	1,8895
1	3	1	17520	1,67	0,999784296	1,8895
1	3	1	8760	0,835	0,999784296	1,8895
3	5	1	17520	1,67	0,999784293	1,8895
3	5	1	8760	0,835	0,999784293	1,8895
1	5	1	8760	1,67	0,999689016	2,72421
1	3	1	8760	1,67	0,999689015	2,72422
3	5	1	8760	1,67	0,999689012	2,72425
3	3	3	17520	0,835	0,997825954	19,0446
3	3	3	17520	1,67	0,997825954	19,0446
3	3	3	8760	0,835	0,997825954	19,0446
3	3	3	8760	1,67	0,997825954	19,0446
3	3	1	17520	0,835	0,997658264	20,5136
3	3	1	17520	1,67	0,997610720	20,9300
3	3	1	8760	0,835	0,997610720	20,9300
3	3	1	8760	1,67	0,997515647	21,7629

6.2.3 Probabilidade de acesso ao dado mais recente

A análise do projeto de experimentos da probabilidade (P) de acessar dados mais recentes nos SGBDs NoSQL baseado em quorum é apresentada a seguir. Essa probabilidade é a expec-

tativa de que um dado acessado esteja em sua última versão dentre os servidores redundantes.

Este projeto de experimento adota quatro fatores ($k = 4$) com 2 níveis ($l = 2$): (i) nível de consistência ($cons$) - 1, 3; (ii) fator de replicação (rf) - 3, 5; (iii) tempo entre solicitações do cliente ($arrival$) - $100ms$, $200ms$; e (iv) atraso de gravação ($update$) - $2,5ms$, $5ms$. Os valores para ($arrival$) representam uma demanda distinta do cliente, e ($update$) pode variar devido a uma melhor capacidade de *hardware* ou tecnologia. Ambos têm um incremento de 100% para o nível mais alto.

A Tabela 17 descreve a classificação dos principais efeitos de fatores e suas interações. ($cons$) tem a maior influência na probabilidade de acesso ao dado mais recentes, no sentido de que um nível mais alto eleva a possibilidade de obter os dados mais recente dentre as réplicas. Para uma melhor visualização, a Figura 19 apresenta gráficos de efeito que indicam o impacto na probabilidade devido a fatores e níveis adotados no DoE, por exemplo, o efeito da consistência se destaca entre os demais fatores (ver Figura 19(a)).

($arrival$) e ($update$) têm o mesmo impacto no acesso aos dados mais recentes. No entanto, possuem direções opostas; o tempo entre chegadas mais alto aumenta a probabilidade, mas uma operação de gravação mais lenta diminui a probabilidade de acessar os dados mais recentes. Considerando que ($cons$) é um fator de grande influência, ($cons \times arrival$) e ($cons \times update$) são as interações mais importantes. (rf) não está presente nas fileiras, pois a respectiva influência é desprezível. No entanto, (rf) está implicitamente associado ao nível de consistência ($cons$).

Tabela 17 – Classificação dos efeitos principais e de interação (probabilidade)

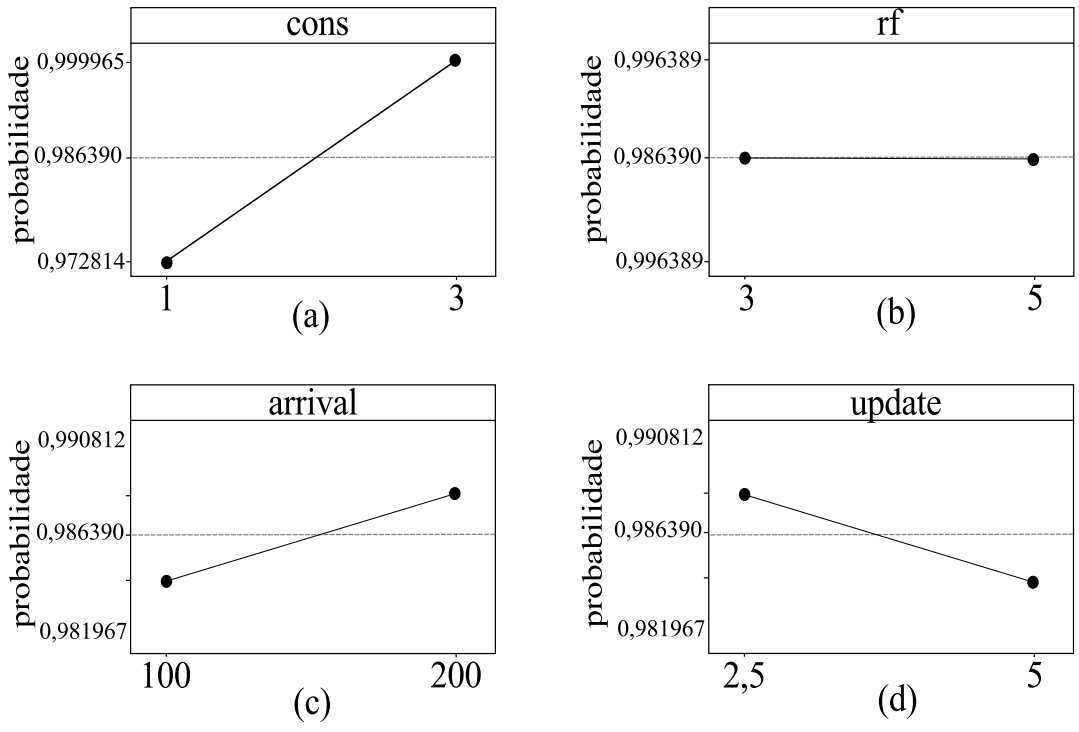
Fator/interação	|Efeito
$cons$	0,02715
$update$	-0,00884
$arrival$	0,00884
$cons \times update$	-0,00879
$cons \times arrival$	0,00879
$arrival \times update$	-0,00281

O resultado de cada tratamento do DoE é apresentado na Tabela 18. Os valores mais altos estão associados ao nível de consistência 3, pois são adotados mais servidores para operação de leitura.

Tabela 18 – Resultados do projeto de experimentos para probabilidade

$cons$	rf	$arrival$ (ms)	$update$ (ms)	probabilidade P
1	3	100	5	0,952380
1	5	100	5	0,952380
1	3	100	2,5	0,975609
1	5	100	2,5	0,975609
1	3	200	5	0,975609
1	5	200	5	0,975609
1	3	200	2,5	0,987654
1	5	200	2,5	0,987654
3	3	100	5	0,999892
3	5	100	5	0,999892
3	3	100	2,5	0,999985
3	5	100	2,5	0,999985
3	3	200	5	0,999985
3	5	200	5	0,999985
3	3	200	2,5	0,999998
3	5	200	2,5	0,999998

Figura 19 – Gráficos de efeito – probabilidade de acesso ao dado mais recente



Fonte: Elaboração Própria.

6.2.4 Consumo de energia

Para estes experimentos foram adotados cinco fatores ($k = 5$) com 2 níveis ($l = 2$): (i) nível de consistência ($cons$) - 1, 3; (ii) fator de replicação (rf) - 3, 5; (iii) proporção das operações de leitura e escrita (crw) - $0,8\ read/0,2\ write$, $0,6\ read/0,4\ write$; (iv) consumo de energia para leitura ($pread$) - 32, 40; e consumo de energia para gravação ($pwrite$) - 40, 48. O tempo T foi mantido em 1 hora. Para permitir a avaliação do nível de consistência ($cons = 3$), o nível mais baixo para (rf) tem o mesmo valor. (crw) representa perfis de uso distintos e seus valores indicam sistemas que executam cargas de trabalho diferentes; os valores para ($pread$) e ($pwrite$) denotam a variação de consumo de energia entre operações. ($pread$) e ($pwrite$) foram estimados em um sistema real usando a ferramenta E-meter (GOMES; TAVARES; JUNIOR, 2016). A métrica de consumo de energia é dada em $Joule$ (J).

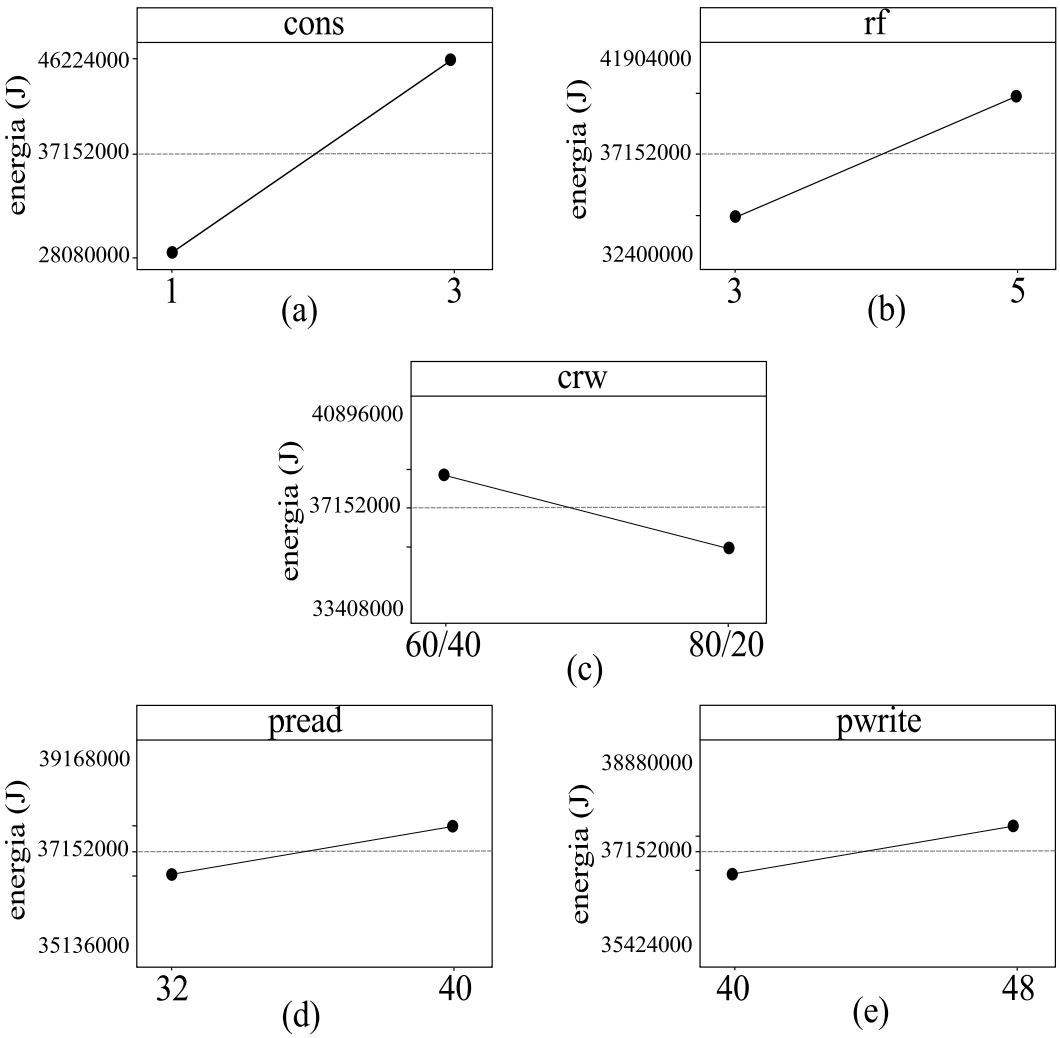
A Tabela 19 descreve a classificação dos efeitos e suas interações. ($cons$) tem a maior influência no consumo de energia, pois diferentes números de servidores precisam ser levados em consideração para confirmar a operação de leitura. (rf) é o segundo fator mais influente, uma vez que também considera o número de servidores que executa a operação de escrita. Considerando a proporção de operações de leitura e escrita (crw), o efeito é na direção oposta, pois o segundo nível tem uma proporção de leitura menor que o primeiro.

É importante destacar que, diferentemente dos modelos anteriores, para a escrita, a avaliação de consumo de energia não é confirmada com base no nível de consistência, pois rf escritas são realizadas, mesmo depois de confirmada a operação para a requisição do cliente.

A Figura 20 demonstra gráficos de efeito que indicam o impacto do consumo de energia devido aos níveis adotados para os diversos fatores. O consumo de energia é quase o dobro para o nível de consistência 3 em relação ao nível 1 (ver Figura 20(a)). Outra característica interessante apresentada é o fato de que quando a proporção da leitura aumenta, o consumo de energia diminui(ver Figura 20(c)).

As interações dos fatores têm pouca influência no consumo de energia. Além disso, a Tabela 20 indica o consumo de energia para cada tratamento. Conforme apresentado anteriormente, valores maiores estão associados ao nível de consistência 3 devido à adoção de servidores adicionais.

Figura 20 – Gráficos de efeito – consumo de energia



Fonte: Elaboração Própria.

Tabela 19 – Classificação dos efeitos principais e de interação (consumo de energia)

Fator/interação	|Efeito|(J)
$cons$	181440
rf	95040
crw	-74880
$pread$	40320
$pwrite$	34560
$rf \times crw$	-31680
$cons \times crw$	25920
$cons \times pread$	20160
$crw \times pwrite$	-11520
$rf \times pwrite$	8640
$crw \times pread$	5760

Tabela 20 – Resultados do projeto de experimentos para consumo de energia

$cons$	rf	crw	$pread$	$pwrite$	consumo de energia (J)
1	3	(80/20)	32	40	178560
1	3	(80/20)	32	48	195840
1	3	(80/20)	40	40	201600
1	3	(80/20)	40	48	218880
1	5	(80/20)	32	40	236160
1	3	(60/40)	32	40	241920
1	5	(80/20)	40	40	259200
1	3	(60/40)	40	40	259200
1	5	(80/20)	32	48	264960
1	3	(60/40)	32	48	276480
1	5	(80/20)	40	48	288000
1	3	(60/40)	40	48	293760
1	5	(60/40)	32	40	357120
3	3	(80/20)	32	40	362880
1	5	(60/40)	40	40	374400
3	3	(60/40)	32	40	380160
3	3	(80/20)	32	48	380160
1	5	(60/40)	32	48	414720
3	3	(60/40)	32	48	414720
3	5	(80/20)	32	40	420480
1	5	(60/40)	40	48	432000
3	3	(80/20)	40	40	432000
3	3	(60/40)	40	40	432000
3	5	(80/20)	32	48	449280
3	3	(80/20)	40	48	449280
3	3	(60/40)	40	48	466560
3	5	(80/20)	40	40	489600
3	5	(60/40)	32	40	495360
3	5	(80/20)	40	48	518400
3	5	(60/40)	40	40	547200
3	5	(60/40)	32	48	552960
3	5	(60/40)	40	48	604800

6.3 ESTUDOS DE CASO

Esta seção apresenta dois estudos de casos que demonstram a utilização prática dos modelos propostos nesta tese. Os estudos de caso são baseados no provimento de uma solução de *Database as a service* (DBaaS). Essas soluções têm se popularizado por fornecerem serviços de banco de dados sem incluir preocupações com a infraestrutura e a configuração da tecnologia para os clientes (AL-REFAI; HAYA et al., 2021). Os provedores desses serviços necessitam fornecer níveis de SLA claros para esses clientes, assim como obter bons níveis de eficiência em seus sistemas para a sustentação do negócio.

O primeiro estudo de caso visa obter uma referência para o SLA dos serviços, considerando o desempenho, a disponibilidade e a probabilidade de acesso ao dado mais recente em duas opções de SGBD NoSQL baseado em quórum para diferentes níveis de consistência.

No segundo estudo de caso, agora considerando opções específicas na infraestrutura do serviço, de forma a possibilitar a escolha da abordagem mais eficiente para o provimento do serviço, estendemos o primeiro estudo na avaliação da disponibilidade. Ademais, introduziremos uma breve avaliação do consumo de energia desses ambientes.

6.3.1 Estudo de caso 1

SGBDs NoSQL distintos apresentam desempenho diferentes, mesmo que utilizem a mesma carga de trabalho e compartilhem o mesmo *hardware* (KLEIN et al., 2015; MARTINS; ABBASI; SÁ, 2019). Dessa forma, este primeiro estudo de caso visa avaliar o desempenho, a probabilidade de acesso ao dado mais recente e a disponibilidade em dois SGBDs NoSQL baseados em quorum para um mesmo ambiente DBaaS.

Inicialmente, para fins dessa comparação, o experimento utiliza os valores baseados na validação do modelo de desempenho proposto com os SGBDs Cassandra e Riak para o procedimento de leitura. Os valores da operação de escrita consideram como base a variação do desempenho distinta entre operações do SGBD. Por exemplo, alguns SGBDs têm melhor desempenho de leitura em relação a sua escrita. Outros SGBDs invertem essa relação (GOMES; TAVARES; JUNIOR, 2016; MARTINS; ABBASI; SÁ, 2019).

Avaliam-se os níveis de consistência 1 e 3. Para simplificar o experimento, foi fixado o valor do fator de replicação $RF = 3$. Os valores para falha e reparo do sistema foram: $MTTF =$ 481,46h e $MTTR =$ 1,03h para o bloco físico, que inclui toda a infraestrutura do sistema; e

$MTTF = 1439{,}99$h e $MTTR = 0{,}17$h para o bloco da máquina virtual que integra o NoSQL e a VM. Adicionalmente, inferimos os valores, servidores físicos, $N = 3$, e número de servidores físicos necessários, $K = 1$. A Tabela 21 e a Tabela 22 apresentam os demais valores utilizados no experimento de desempenho. Já as Tabelas 23 e 24 apresentam os demais valores utilizados no experimento da probabilidade de acesso ao dado mais recente.

Tabela 21 – Tempos de transição para desempenho, estudo de caso 01 – SGBD (Cassandra)

Transição	Tempo
tClient ($chegada_w$)	18,6ms
tCoord (cc_w)	5,2ms
tClient ($chegada_r$)	12,4ms
tCoord (cc_r)	10,24ms
tRespSuc ($resp$)	2,6ms

Tabela 22 – Tempos de transição para desempenho, estudo de caso 01 – SGBD (Riak)

Transição	Tempo
tClient ($chegada_w$)	18,6ms
tCoord (cc_w)	4ms
tClient ($chegada_r$)	12,4ms
tCoord (cc_r)	3,49ms
tRespSuc ($resp$)	1,8ms

Tabela 23 – Tempos de transição para consistência, estudo de caso 01 – SGBD (Cassandra)

Transição	Tempo
tClient ($chegada_w$)	18,6ms
tUpdate ($escrita$)	5,2ms

Tabela 24 – Tempos de transição para consistência, estudo de caso 01 – SGBD (Riak)

Transição	Tempo
tClient ($chegada_w$)	18,6ms
tUpdate ($escrita$)	4ms

A Tabela 25 apresenta os resultados da comparação dos SGBDs NoSQL baseados em quorum avaliados. As colunas $escrita$ e $leitura$ são as latências dessas operações, $disponibilidade$ é a probabilidade do sistema atender às requisições, e $probabilidade$ é a probabilidade de acesso ao dado mais recente. Os resultados reforçam as conclusões verificadas nos experimentos anteriores para o nível de consistência das operações. No entanto, outra informação

Tabela 25 – Resultado do estudo de caso 01

SGBD NoSQL	$cons$	escrita (ms)	leitura (ms)	disponibilidade	probabilidade
Cassandra	1	20,55	18,45	0,9999999	0,781512
	3	22,37	24,49	0,9996459	0,989570
Riak	1	19,75	13,73	0,9999999	0,823000
	3	20,90	14,97	0,9996459	0,994454

importante é observada nessa comparação. O SGBD com melhor desempenho, fornece uma maior probabilidade de acesso ao dado mais recente.

Isso ocorre porque um melhor desempenho na escrita dos dados, dentro do *cluster* do SGBD baseado em quorum, diminui a janela de inconsistência entre as réplicas. É importante poder mensurar a qualidade do serviço no impacto da escolha do SGBD. Por exemplo, os maiores níveis de consistência apresentam menor desempenho das operações, mas o SGBD Riak apresentou maior consistência de acesso no nível de consistência 3 em relação ao nível de consistência 1 do SGBD Cassandra.

Outra observação interessante é que, para o SGBD Riak, a influência do maior nível de consistência no desempenho das operações de escrita e leitura são de (5,4%) e (8,2%), respectivamente, enquanto para a probabilidade de acesso ao dado mais recente é de (17,2%). Isso pode indicar que nessas configurações é mais viável manter o maior nível de consistência das operações.

A escolha do nível de consistência afetou o *downtime* anual do sistema com variação de 3 segundos no nível de consistência 1 para 3 horas no nível de consistência 3. Ademais, uma maior disponibilidade implicou menor probabilidade de acesso aos dados mais recentes.

6.3.2 Estudo de caso 2

Para este estudo de caso, avalia-se o impacto do nível de consistência e da arquitetura escolhida na disponibilidade e consumo de energia das operações no NoSQL baseado em quorum. A avaliação verifica o resultado da escolha de uma tecnologia específica na infraestrutura do sistema. Essa avaliação é possível por meio do modelo RBD proposto nessa tese para estimar a disponibilidade.

A escolha da arquitetura avalia duas tecnologias de virtualização: a primeira baseada em um *hypervisor* (MELO et al., 2018) e a segunda baseada em *container Docker* (ARAUJO et al., 2019). Apesar de não influenciarem significativamente na infraestrutura de *hardware*, elas

possuem tempos de falha ligeiramente diferentes. Outro ponto aferido é a redundância dos equipamentos físicos desse sistema $N = 1, 2$. Consideramos que apenas uma infraestrutura é suficiente para que o sistema funcione $K = 1$. Semelhante ao estudo de caso anterior, avaliam-se os níveis de consistência 1 e 3 das operações, e foi fixado o valor do fator de replicação $rf = 3$.

Para a avaliação da disponibilidade, utilizamos os valores descritos em (MELO et al., 2018; ARAUJO et al., 2019). Os valores para falha e reparo do *hypervisor* foram: $MTTF = 2.900$h e $MTTR = 1$h. Para o *Docker*, foram $MTTF = 2.990$h e $MTTR = 1$h. Os demais valores utilizados são apresentados na Tabela 26.

Para o consumo de energia, baseamos a proporção das operações de leitura e escrita $read/0,95\ write, 0,05$ no *benchmark* YCSB, utilizado em diversos estudos (LIU; ZHOU, 2021). O consumo de energia para leitura ($pread$) - $32W$; e o consumo de energia para gravação ($pwrite$) - $40W$. O tempo T foi verificado em 1 ano. A métrica de consumo de energia é dada em $Joule$ (J).

Tabela 26 – MTTF e MTTR dos componentes do modelo

Component	MTTF (h)	MTTF (h)
HW	8760	1,67
OS	2893	0,25
VM	2880	0,0958
NSQ	1440	0,33

Tabela 27 – Resultado do estudo de caso 02

Virtualização	$cons$	N	disponibilidade	$downtime$ (h/ano)	energia (J/ano)
Docker	1	2	0,9999996	0,0035	1147910400
Hypervisor	1	2	0,9999996	0,0035	1147910400
Docker	1	1	0,9993887	5,35	1147910400
Hypervisor	1	1	0,9993783	5,44	1147910400
Docker	3	2	0,9992127	6,89	3065299200
Hypervisor	3	2	0.9992127	6,89	3065299200
Docker	3	1	0,9986023	12,24	3065299200
Hypervisor	3	1	0.9985919	12,33	3065299200

A Tabela 27 apresenta em ordem decrescente os resultados da avaliação da disponibilidade ($disponibilidade$) e do consumo de energia ($energia$) das operações de leitura e escrita de

um determinado SGBD NoSQL baseado em quorum. Adicionalmente a coluna $downtime$ demonstra o tempo em que o sistema está indisponível para o período de 1 ano (8760 horas).

Os resultados ratificam a importância da consistência para a disponibilidade do sistema. A indisponibilidade $downtime$ do sistema, dentre as possibilidades fornecidas nesse estudo de caso, variou entre 12,33 horas no pior caso e 12,6 segundos no melhor caso. Para o nível de consistência 3, mesmo com redundância de infraestrutura, o $downtime$ anual impossibilita que o sistema seja classificado na categoria de alta disponibilidade. Essa classificação apenas é alcançada quando o sistema utiliza o nível de consistência 1 e 2 infraestruturas redundantes (ÖHMANN; SIMSEK; FETTWEIS, 2014). Para a tecnologia de virtualização, não há impacto relevante nessa escolha na disponibilidade dos sistemas avaliados.

A influência do consumo de energia das operações demonstra que essas configurações não são triviais no impacto do custo do sistema. Justificada pela relação de 95% de leitura das operações em detrimento de 5% de escrita, a energia utilizada no nível de consistência 3 foi 2,67 vezes maior que no nível 1. Esse consumo de energia não considera a redundância física da infraestrutura do sistema.

6.4 CONSIDERAÇÕES FINAIS

A consistência eventual é um recurso importante do SGBD NoSQL baseado em quorum com significativo impacto na disponibilidade, desempenho, probabilidade de acesso aos dados mais recentes e consumo de energia. Um nível de consistência adequado é uma decisão de projeto que deve considerar os objetivos do sistema e seu custo de implementação. Os modelos propostos podem auxiliar os projetistas a avaliarem os níveis de consistência em configurações distintas do sistema.

Esses resultados indicam que os níveis de consistência afetam várias métricas, como desempenho, disponibilidade, probabilidade de acessar informações mais recentes e o consumo de energia. Provedores de serviços de banco de dados NoSQL baseado em quorum, e outras organizações que busquem a melhor relação de custo-benefício dos sistemas podem se beneficiar por meio do uso dos modelos apresentados, podendo, então, definir o nível de consistência e as configurações mais adequadas para considerar o melhor conjunto de requisitos para obtenção dessas métricas.

7 CONCLUSÃO E COMENTÁRIOS FINAIS

"Eu não tenho ídolos. Tenho admiração por trabalho, dedicação e competência."

— Ayrton Senna

O SGBD não relacional (NoSQL) foi introduzido para fornecer uma opção para atendimento a necessidades com as quais os bancos de dados relacionais têm dificuldade de lidar, como, maior flexibilidade, disponibilidade e escalabilidade do sistema. Nesse sentido, o referido sistema oferece uma abordagem mais eficiente para lidar com várias categorias de dados e suas possíveis alterações. Sistemas Gerenciadores de Banco de Dados NoSQL baseado em quorum, atualmente, são uma importante categoria de SGBD com relevante utilização em organizações em todo o mundo.

Apesar dessa relevância, a caracterização dos aspectos de qualidade que dependem de sua configuração, em especial, os aspectos relacionados ao nível de consistência das operações, ainda são pouco explorados em relação a outros aspectos. A consistência do ponto de vista abordado tem sua importância relacionada à manipulação do dado mais recente o que pode implicar aspectos como vantangem comercial ou maior confiança no sistema. Nas organizações e no meio acadêmico, as principais pesquisas ressaltam o desempenho e alguns aspectos do nível de consistência desses SGBDs.

A presente tese de doutorado propõe em um mesmo trabalho a avaliação do desempenho, da disponibilidade, da probabilidade de acesso ao dado mais recente e do consumo de energia. São desenvolvidos modelos que permitem a mensuração de vários cenários, o que contribui no projeto e arquitetura de SGBDs NoSQL baseado em quorum. São aplicadas técnicas de projeto de experimentos para avaliação dos efeitos dos fatores avaliados e suas interações. Adicionalmente estudos de caso demonstram o uso dos modelos propostos.

Nesse contexto, a consistência representa um único estado válido para todas as instâncias do banco de dados e significa que todos os servidores redundantes possuem a última versão do dado. Os SGBDs NoSQL adotam, para melhorar o desempenho e a disponibilidade dos seus serviços, a consistência eventual, que permite uma inconsistência temporária enquanto os dados são atualizados nos servidores redundantes. Eventualmente todos os servidores estarão atualizados (consistentes). Diversos SGBDs NoSQL utilizam a consistência baseada em quorum, em que os níveis de consistência são verificados na quantidade de servidores que confirmam uma operação (nível de consistência).

Os resultados demonstram que as configurações dos níveis de consistência possuem significativa influência no desempenho, na disponibilidade, na probabilidade de acesso ao dado mais recente e no consumo de energia. Essas métricas por vezes podem ser conflitantes: por exemplo, elevar o nível de consistência melhora a probabilidade de acesso ao dado mais recente, no entanto, aumenta o tempo das operações, diminui a disponibilidade do sistema e eleva o consumo de energia.

É importante enfatizar que além de mensurar o impacto em cada métrica, os modelos possibilitam que o projetista do sistema modifique outras configurações do sistema para obter o melhor equilíbrio com um menor custo. Por exemplo, sistemas com melhor desempenho sofrem menos impacto na probabilidade de acesso a dados mais recentes. Outro bom exemplo é que altos níveis de consistência dificultam significativamente a alta disponibilidade do sistema mesmo com altos investimentos em infraestrutura física redundante.

Os modelos propostos foram desenvolvidos inicialmente com rede de Petri estocástica. Para verificar os componentes da infraestrutura na avaliação da disponibilidade dos sistemas, utilizamos Diagrama de Blocos de Confiabilidade. Esses modelos foram validados por experimentos reais e técnicas estatísticas descritas nesta tese. O modelo de consumo de energia foi concebido por coleta de dados e análise do funcionamento do SGBD baseado em quorum. Descrevemos nas próximas seções as principais contribuições desta tese bem como suas limitações. Por fim, apresentamos algumas possibilidades de trabalhos futuros derivados deste trabalho.

7.1 CONTRIBUIÇÕES

O trabalho desenvolvido nesta tese de doutorado conduz a um conjunto de contribuições ao estado da arte do conhecimento sobre SGBD NoSQL baseado em quorum, além da possibilidade de aplicação dessas técnicas em organizações. As principais contribuições são apresentadas a seguir.

- **Modelos** – Proposição de modelos – O primeiro modelo estima o desempenho e a disponibilidade, em conjunto ou separadamente, considerando a capacidade do SGBD e a demanda do cliente. O segundo modelo é de probabilidade de acesso ao dado mais recente que, por meio da chegada de novas atualizações e da capacidade do sistema, mensura a possibilidade de acesso ao dado atualizado. O terceiro, permite a avaliação

da disponibilidade, possibilitando a alteração de componentes de sua infraestrutura. Por fim, o modelo de consumo de energia, que estima o consumo de energia das operações do SGBD NoSQL baseado em quorum em função de seu perfil de utilização. Todos os modelos são baseados nas configurações de replicação e nível de consistência do NoSQL baseado em quorum.

- **Experimentos** – Realizou-se uma série de projetos de experimentos que evidenciaram importantes fatores e interações para cada métrica avaliada nos modelos. Esses experimentos demonstram a importância da escolha do nível de consistência e disponibilizam na literatura valores-referência para estudos complementares.

- **Estudos de casos** – A realização de estudos de caso permitiu exemplificar a utilização dos modelos em um provedor de serviço DBaaS, comparando possibilidades específicas. Outra contribuição importante dos estudos de caso foi demonstrar uma relação entre configurações, por exemplo, melhor desempenho do SGBD está relacionado a uma melhor probabilidade de acessar um dado mais recente, em razão do menor tempo de propagação das atualizações de dados. Outro exemplo é a evidência do alto custo na utilização da consistência forte que nesse nível de consistência implicou maiores indisponibilidades e consumo de energia das operações.

A importância da contribuição desta tese de doutorado dá-se com o provimento de ferramentas e exemplos para mensurar o impacto das métricas: desempenho, disponibilidade, probabilidade de acesso ao dado mais recente e consumo de energia, conforme o nível de consistência. O ambiente NoSQL é utilizado principalmente em escalas maiores; a otimização desses sistemas pode prover uma melhora significativa na aquisição e manutenção de seus recursos.

Contribuições importantes nesta pesquisa foram as publicações:

- **SysCon:** *Carlos Gomes, E. Borba, E. Tavares and M. N. d. O. Junior, Performability Model for Assessing NoSQL DBMS Consistency, 2019 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 2019, pp. 1-6. doi:10.1109/SYSCON.2019.8836757*

- **Journal of Supercomputing:** *Carlos Gomes, E. Borba, E. Tavares and M. N. d. O. Junior, Bruno Nogueira. Cloud storage availability and performance assessment: a study based on NoSQL DBMS. The Journal of Supercomputing (DORDRECHT. ONLINE), v. 1, p. 1-21, 2021. doi:10.1109/SYSCON.2019.8836757*

Outros dois artigos foram submetidos em *Journal* e estamos aguardando as decisões.

7.2 LIMITAÇÕES

Algumas limitações verificadas no decorrer dessa pesquisa são descritas a seguir.

- **Versão do SGBD Cassandra** – No decorrer da pesquisa, após a publicação do primeiro artigo, o SGBD Cassandra em sua versão (2.1), que foi a opção utilizada na validação, foi descontinuado. Em razão desse fato, introduzimos o SGBD Riak KV em sua última versão (2.2) para reforçar a adequação das propostas ao SGBD baseado em quorum usando outro SGBD.

- **Consumo de energia** – Para a obtenção do resultado do modelo de consumo de energia, atualmente é necessário medir o consumo médio das operações do SGBD. Essa tarefa não é trivial e sua proposta será detalhada em trabalhos futuros.

- **Escopo** – A principal limitação dessa pesquisa refere-se à proposição específica dos modelos para SGBDs NoSQL que utilizam a consistência baseada em quorum; outras abordagens não foram verificadas.

- **Técnica e modelos propostos** – Um requisito para utilização dos modelos propostos nesta tese de doutorado é que o arquiteto do sistema tenha conhecimento básico do SGBD avaliado, assim como da infraestrutura na qual o sistema está implementado. Por exemplo, para avaliar sistemas disponíveis em DBaaS, é necessário acesso às informações de sua infraestrutura.

7.3 TRABALHOS FUTUROS

Apesar desta tese de doutorado ter apresentado avanços significativos na avaliação do desempenho, da disponibilidade, da probabilidade de acesso ao dado mais recente e do consumo de energia de SGBDs NoSQL baseados em quorum, foram identificadas algumas pesquisas complementares que podem adicionar ainda mais relevância ao estado da arte. Elas são citadas a seguir.

- **Consumo de energia** – A principal oportunidade de pesquisa observada é o desenvolvimento de técnicas e modelos que possam estimar o consumo individual das operações

em SGBD NoSQL. Sugerimos a decomposição do modelo de consumo de energia proposto em modelos que contenham parâmetros e comportamentos contidos nos sistemas, sem a necessidade de medição do consumo de energia.

- **Extensão de outras características** – Os modelos propostos nesta tese generalizam as características da consistência baseada em quorum e permitem que características peculiares de SGBDs NoSQL baseado em quorum sejam estendidas a outros modelos, permitindo a avaliação de outras características dos SGBDs.

- **Consistência** – O trabalho realizado teve escopo definido na consistência baseada em quorum; observou-se que outros modelos de consistência NoSQL podem ser comparados e modelados utilizando-se técnicas semelhantes às descritas nesta tese.

- **Disponibilidade orientada a capacidade** – Os modelos desenvolvidos consideram a disponibilidade d sistemas nos quais a demanda dos servidores suporta as requisições do clientes. Uma oportunidade identificada é o desenvolvimento de modelos de disponibilidade que considerem a disponibilidade orientada a capacidade desses SGBDs NoSQL.

- **Novas aplicações** – O desenvolvimento de aplicativos que forneçam as métricas propostas nos modelos com a simples parametrização destes sistemas.

- **Novas aplicações** – A criação de mecanismos autônomos que configurem dinamicamente os sistemas em função da qualidade do serviço solicitada.

REFERÊNCIAS

AHSANULLAH, M.; KIBRIA, B. G.; SHAKIL, M. Normal distribution. In: *Normal and Student st Distributions and Their Applications*. [S.l.]: Springer, 2014. p. 7–50.

AL-REFAI, M.; HAYA, A. et al. Database as a service (dbaas) challenges and solutions. In: IEEE. *2021 22nd International Arab Conference on Information Technology (ACIT)*. [S.l.], 2021. p. 1–6.

ANDRAE, A. S. Comparison of several simplistic high-level approaches for estimating the global energy and electricity use of ict networks and data centers. *International Journal*, v. 5, p. 51, 2019.

ARAUJO, C. J. M. *Avaliação e modelagem de desempenho para planejamento de capacidade de sistema de transferência eletrônica de fundos utilizando tráfego em rajada*. [S.l.]: Universidade Federal de Pernambuco, 2009.

ARAUJO, E.; DANTAS, J.; MATOS, R.; PEREIRA, P.; MACIEL, P. Dependability evaluation of an iot system: A hierarchical modelling approach. In: IEEE. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. [S.l.], 2019. p. 2121–2126.

ARAUJO, J. M. A.; MOURA, A. C. E. de; SILVA, S. L. B. da; HOLANDA, M.; RIBEIRO, E. d. O.; SILVA, G. L. da. Comparative performance analysis of nosql cassandra and mongodb databases. In: *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2021. p. 1–6.

ARCHIVE, N. *Your Ultimate Guide to the Non-Relational Universe!* 2022. Http://nosql-database.org/. Acesso em: 15 de out. de 2022.

ATTIYA, H.; ELLEN, F.; MORRISON, A. Limitations of highly-available eventually-consistent data stores. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 28, n. 1, p. 141–155, 2016.

BAILIS, P.; VENKATARAMAN, S.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; STOICA, I. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, VLDB Endowment, v. 5, n. 8, p. 776–787, abr. 2012. ISSN 2150-8097.

BAILIS, P.; VENKATARAMAN, S.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; STOICA, I. Quantifying eventual consistency with pbs. *The VLDB Journal*, Springer, v. 23, n. 2, p. 279–302, 2014.

BARON et al. Nosql key-value dbs riak and redis. *Database Systems Journal*, Academy of Economic Studies-Bucharest, Romania, v. 4, p. 3–10, 2016.

BASHIR, E.; LUŠTREK, M. The mercury environment: a modeling tool for performance and dependability evaluation. In: IOS PRESS. *Intelligent Environments 2021: Workshop Proceedings of the 17th International Conference on Intelligent Environments*. [S.l.], 2021. v. 29, p. 16.

BOBBA, R.; GROV, J.; GUPTA, I.; LIU, S.; MESEGUER, J.; ÖLVECZKY, P. C.; SKEIRIK, S. Survivability: design, formal modeling, and validation of cloud storage systems using maude. *Assured Cloud Computing*, John Wiley & Sons, Inc. Hoboken, NJ, USA, p. 10–48, 2018.

BOUAZIZ, S.; NABLI, A.; GARGOURI, F. Nosql big data warehouse: Review and comparison. In: SPRINGER. *International Conference on Intelligent Systems Design and Applications*. [S.l.], 2020. p. 392–401.

BREWER, E. Cap twelve years later: How the"rules"have changed.(2012). *URL: http://www. infoq. com/articles/cap-twelve-yearslater-how-the-rules-have-changed*, 2012.

BUKH, P. N. D.; JAIN, R. *The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling*. [S.l.]: JSTOR, 1992.

BURDAKOV, A.; GRIGOREV, U.; PLOUTENKO, A.; TTSVIASHCHENKO, E. Estimation models for nosql database consistency characteristics. In: *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. [S.l.: s.n.], 2016. p. 35–42.

BUZACOTT, J. A. Finding the mtbf of repairable systems by reduction of the reliability block diagram. *Microelectronics Reliability*, Elsevier, v. 6, n. 2, p. 105–112, 1967.

CHANDRA, D. G. Base analysis of nosql database. *Future Generation Computer Systems*, Elsevier, v. 52, p. 13–21, 2015.

CHATTARAJ, D.; SARMA, M.; SAMANTA, D. Stochastic petri net based modeling for analyzing dependability of big data storage system: Proceedings of iemis 2018, volume 2. In: _____. [S.l.: s.n.], 2019. p. 473–484. ISBN 978-981-13-1497-1.

COOPER, B. *Yahoo! Cloud Serving Benchmark*. 2019. Https://github.com/brianfrankcooper/YCSB. Acesso em: 10 de fev. de 2022.

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

DAS, N.; PAUL, S.; SARKAR, B. B.; CHAKRABARTI, S. Nosql overview and performance testing of hbase over multiple nodes with mysql. In: *Emerging Technologies in Data Mining and Information Security*. [S.l.]: Springer, 2019. p. 269–279.

DATASTAX. *Datastax Documentation*. 2022. Https://docs.datastax.com/en/cassandra-oss/2.1/cassandra/tools/toolsCFstats.html. Acesso em: 12 de fev. de 2022.

DAVOUDIAN, A.; CHEN, L.; LIU, M. A survey on nosql stores. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 2, p. 1–43, 2018.

DIPIETRO, S. Performance modelling and optimisation of nosql database systems. *ACM SIGMETRICS Performance Evaluation Review*, ACM New York, NY, USA, v. 47, n. 3, p. 10–13, 2020.

D'AGOSTINO, R. B. Tests for the normal distribution. In: *Goodness-of-fit techniques*. [S.l.]: Routledge, 2017. p. 367–420.

FRANK, P. *Introduction to sensitivity analysis*. [S.l.]: Academic Press London, 1978.

GAO, G.-S.; KONWAR, K.; MANTICA, J.; PAN, H.; KISH, D. R.; TSENG, L.; WANG, Z.; WU, Y. Practical experience report: Cassandra+: Trading-off consistency, latency, and fault-tolerance in cassandra. In: *International Conference on Distributed Computing and Networking 2021*. [S.l.: s.n.], 2021. p. 191–195.

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

GIFFORD, D. K. Weighted voting for replicated data. In: ACM. *Proceedings of the seventh ACM symposium on Operating systems principles*. [S.l.], 1979. p. 150–162.

GOMES, C.; TAVARES, E.; JUNIOR, M. N. d. O. Energy consumption evaluation of nosql dbmss. 2016.

GOMES, V. B. F.; KLEPPMANN, M.; MULLIGAN, D. P.; BERESFORD, A. R. Verifying strong eventual consistency in distributed systems. *Proc. ACM Program. Lang.*, ACM, v. 1, n. OOPSLA, p. 109:1–109:28, 2017. ISSN 2475-1421.

GORBENKO, A.; ROMANOVSKY, A.; TARASYUK, O. Interplaying cassandra nosql consistency and performance: A benchmarking approach. In: SPRINGER. *European Dependable Computing Conference*. [S.l.], 2020. p. 168–184.

GRAY, J.; SIEWIOREK, D. P. High-availability computer systems. *Computer*, IEEE, v. 24, n. 9, p. 39–48, 1991.

HAHN, S. M. L.; CHEREJA, I.; MATEI, O. Evaluation of transformation tools in the context of nosql databases. In: SPRINGER. *Proceedings of SAI Intelligent Systems Conference*. [S.l.], 2021. p. 146–165.

HAMBY, D. M. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment*, Springer, v. 32, n. 2, p. 135–154, 1994.

HAUGHIAN, G.; OSMAN, R.; KNOTTENBELT, W. J. Benchmarking replication in cassandra and mongodb nosql datastores. In: HARTMANN, S.; MA, H. (Ed.). *Database and Expert Systems Applications*. Cham: Springer International Publishing, 2016. p. 152–166. ISBN 978-3-319-44406-2.

HUANG, X.; WANG, J.; QIAO, J.; ZHENG, L.; ZHANG, J.; WONG, R. K. Performance and replica consistency simulation for quorum-based nosql system cassandra. In: SPRINGER. *International Conference on Application and Theory of Petri Nets and Concurrency*. [S.l.], 2017. p. 78–98.

HUANG, X.; WANG, J.; YU, P. S.; BAI, J.; ZHANG, J. An experimental study on tuning the consistency of nosql systems. *Concurrency and Computation: Practice and Experience*, v. 29, n. 12, p. e4129, 2017.

IEA. *Data Centres and Data Transmission Networks*. 2020. Http://www.iea.org/reports/data-centres-and-data-transmission-networks. Accessed: 2021-09-04.

INSIGHTS, H. *Find out what companies are using*. 2021. Https://discovery.hgdata.com. Acesso em: 2021-09-09.

IOOSS, B.; LEMAÎTRE, P. A review on global sensitivity analysis methods. In: *Uncertainty management in simulation-optimization of complex systems*. [S.l.]: Springer, 2015. p. 101–122.

IT solid. *DB-Engines Ranking*. 2022. Http://db-engines.com/en/ranking. Acesso em: 12 de fev. de 2022.

KALID et al. Big-data nosql databases: A comparison and analysis of “big-table”,“dynamodb”, and “cassandra”. In: IEEE. *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*. [S.l.], 2017. p. 89–93.

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

KHAN, M. F.; AZAM, M.; KHAN, M. A.; ALGARNI, F.; ASHFAQ, M.; AHMAD, I.; ULLAH, I. A review of big data resource management: Using smart grid systems as a case study. *Wireless Communications and Mobile Computing*, Hindawi, v. 2021, 2021.

KHELAIFA, A.; BENHARZALLAH, S.; KAHLOUL, L. A new adaptive causal consistency approach in edge computing environment. *International Journal of Computing and Digital Systems*, University of Bahrain, v. 12, n. 1, p. 945–960, 2022.

KHELAIFA, A.; BENHARZALLAH, S.; KAHLOUL, L.; EULER, R.; LAOUID, A.; BOUNCEUR, A. A comparative analysis of adaptive consistency approaches in cloud storage. *Journal of Parallel and Distributed Computing*, Elsevier, v. 129, p. 36–49, 2019.

KLEIJNEN, J. P. Sensitivity analysis and optimization in simulation: design of experiments and case studies. In: IEEE. *Winter Simulation Conference Proceedings, 1995.* [S.l.], 1995. p. 133–140.

KLEIN, J.; GORTON, I.; ERNST, N.; DONOHOE, P.; PHAM, K.; MATSER, C. Performance evaluation of nosql databases: A case study. In: *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*. New York, NY, USA: ACM, 2015. (PABS ’15), p. 5–10. ISBN 978-1-4503-3338-2.

KOOLI, M.; NATALE, G. D. A survey on simulation-based fault injection tools for complex systems. In: IEEE. *2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. [S.l.], 2014. p. 1–6.

KOOT, M.; WIJNHOVEN, F. Usage impact on data center electricity needs: A system dynamic forecasting model. *Applied Energy*, v. 291, p. 116798, 2021. ISSN 0306-2619.

KRECHOWICZ, A.; DENIZIAK, S.; ŁUKAWSKI, G. Highly scalable distributed architecture for nosql datastore supporting strong consistency. *IEEE Access*, v. 9, p. 69027–69043, 2021.

KUMAR, S. P.; LEFEBVRE, S.; CHIKY, R.; HERMANT, O. Consistency-latency trade-off of the libre protocol: A detailed study. In: PINAUD, B.; GUILLET, F.; CREMILLEUX, B.; RUNZ, C. de (Ed.). *Advances in Knowledge Discovery and Management: Volume 7*. Cham: Springer International Publishing, 2018. p. 83–108. ISBN 978-3-319-65406-5.

KUMAR, V.; Thakur, R. S. Big data: Techniques, tools, and technologies–nosql database. In: *Pattern and Data Analysis in Healthcare Settings*. [S.l.]: IGI Global, 2017. p. 158–179.

LENKA, R. K.; PADHI, S.; NAYAK, K. M. Fault injection techniques-a brief review. In: IEEE. *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. [S.l.], 2018. p. 832–837.

LILJA, D. J. *Measuring Computer Performance A practitioner’s guide*. 1th. ed. [S.l.]: Cambridge Univerty Press, 2004.

LIU, L.; ZHOU, K. Ptierdb: Building better read-write cost balanced key-value stores for small data on ssd. In: IEEE. *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. [S.l.], 2021. p. 796–801.

MACIEL, P.; LINS, R.; CUNHA, P. Introduction of the petri net and applied. *X Escola de Computaçao, Campinas, SP*, 1996.

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

MACIEL, P.; TRIVEDI, K.; MATIAS, R.; KIM, D. Dependability modeling. In: _____. [S.l.]: IGI Publishing, 2011. cap. 3, p. 53–97.

MALHOTRA, S.; DOJA, M.; ALAM, B.; ALAM, M. Bigdata analysis and comparison of bigdata analytic approches. In: IEEE. *2017 International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.], 2017. p. 309–314.

MARSAN, M. A.; BALBO, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. Modelling with generalized stochastic petri nets. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 26, n. 2, p. 2–, ago. 1998. ISSN 0163-5999.

MARTINS, P.; ABBASI, M.; SÁ, F. A study over nosql performance. In: SPRINGER. *World Conference on Information Systems and Technologies*. [S.l.], 2019. p. 603–611.

MARUSSY, K.; KLENIK, A.; MOLNÁR, V.; VÖRÖS, A.; MAJZIK, I.; TELEK, M. Efficient decomposition algorithm for stationary analysis of complex stochastic petri net models. In: SPRINGER. *International Conference on Applications and Theory of Petri Nets and Concurrency*. [S.l.], 2016. p. 281–300.

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

MELO, C.; DANTAS, J.; OLIVEIRA, D.; FÉ, I.; MATOS, R.; DANTAS, R.; MACIEL, R.; MACIEL, P. Dependability evaluation of a blockchain-as-a-service environment. In: IEEE. *2018 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2018. p. 00909–00914.

MOHAMED, M. A.; ALTRAFI, O. G.; ISMAIL, M. O. Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, v. 3, n. 03, p. 598–601, 2014.

MOLLOY, M. K. Performance analysis using stochastic petri nets. *IEEE Transactions on computers*, IEEE, n. 9, p. 913–917, 1982.

MONTGOMERY, D. C. *Design and analysis of experiments*. 9th. ed. [S.l.]: John wiley & sons, 2017.

MONTGOMERY, D. C.; RUNGER, G. C. *Applied statistics and probability for engineers*. 6. ed. [S.l.]: John Wiley & Sons, 2013.

MORO, M. P. et al. Eficiência energética e qualidade de serviço em centros de dados. Universidade Federal de Mato Grosso do Sul, 2019.

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

NAVATHE, S. B.; ELMASRI, R. *Sistemas de Banco de Dados*. 6. ed. [S.l.: s.n.], 2010.

NOUDOUST, N. N. S.; ADABI, S.; REZAEE, A. A quorum-based data consistency approach for non-relational database. *Cluster Computing*, Springer, p. 1–26, 2022.

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.

PADHY, S.; KUMARAN, G. M. M. A quantitative performance analysis between mongodb and oracle nosql. In: *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.: s.n.], 2019. p. 387–391.

PATTERSON, D. A.; GIBSON, G.; KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). In: *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 1988. p. 109–116.

PAZ, J. R. G. Introduction to azure cosmos db. In: *Microsoft Azure Cosmos DB Revealed: A Multi-Model Database Designed for the Cloud*. Berkeley, CA: Apress, 2018. p. 1–23. ISBN 978-1-4842-3351-1.

PRAJAPATI, M.; PATEL, S. A review on big data with data mining. In: *Data Science and Intelligent Applications*. [S.l.]: Springer, 2021. p. 155–160.

PRITCHETT, D. Base: An acid alternative. *Queue*, ACM, New York, NY, USA, v. 6, n. 3, p. 48–55, 2008. ISSN 1542-7730.

PULIAFITO, A.; TELEK, M.; TRIVEDI, K. S. The evolution of stochastic petri nets. Citeseer, 1997.

RODRIGUES, L.; ENDO, P. T.; SILVA, F. A. Stochastic model for evaluating smart hospitals performance. In: IEEE. *2019 IEEE Latin-American Conference on Communications (LATINCOM)*. [S.l.], 2019. p. 1–6.

RODRIGUES, M.; VASCONCELOS, B.; GOMES, C.; TAVARES, E. Evaluation of nosql dbms in private cloud environment: An approach based on stochastic modeling. In: *2019 IEEE International Systems Conference (SysCon)*. [S.l.: s.n.], 2019. p. 1–7.

SAHATQIJA, K.; AJDARI, J.; ZENUNI, X.; RAUFI, B.; ISMAILI, F. Comparison between relational and nosql databases. In: IEEE. *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)*. [S.l.], 2018. p. 0216–0221.

SATHAYE, A.; RAMANI, S.; TRIVEDI, K. S. Availability models in practice. In: *Proc. of intl. workshop on fault-tolerant control and computing (FTCC-1)*. [S.l.: s.n.], 2000. v. 52, p. 54.

SHAH, M.; KOTHARI, A.; PATEL, S. A comprehensive survey on energy consumption analysis for nosql. *Scalable Computing: Practice and Experience*, v. 23, n. 1, p. 35–50, 2022.

SINGLA, P.; SINGH, S. S.; GOPINATH, K.; SARANGI, S. Probabilistic sequential consistency in social networks. In: IEEE. *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. [S.l.], 2018. p. 102–111.

SOBOL, V. Enhanced wars model proposal for advancing reasoning consistency based on probabilistically bounded staleness. 2021.

SOFTWARE, M. S. *Minitab*. 2021. Https://www.minitab.com. Acesso em: 17 de nov. de 2021.

SONG, N.; QIN, J.; PAN, X.; DENG, Y. Fault injection methodology and tools. In: IEEE. *Proceedings of 2011 International Conference on Electronics and Optoelectronics*. [S.l.], 2011. v. 1, p. V1–47.

SOUSA, E. T. G. de; LINS, F. A. A. Modeling strategies to improve the dependability of cloud infrastructures. *Dependability Engineering*, BoD–Books on Demand, p. 7, 2018.

TAPEKHIN, A.; BOGOMOLOV, I.; VELIKANOV, O. Analysis of consistency for in memory data grid apache ignite. In: IEEE. *2019 Ivannikov Memorial Workshop (IVMEM)*. [S.l.], 2019. p. 46–50.

TIAN, X.; HAN, R.; WANG, L.; LU, G.; ZHAN, J. Latency critical big data computing in finance. *The Journal of Finance and Data Science*, Elsevier, v. 1, n. 1, p. 33–41, 2015.

VERMA, A. K.; AJIT, S.; KARANKI, D. R. et al. *Reliability and safety engineering*. [S.l.]: Springer, 2010. v. 43.

WAHID, A.; KASHYAP, K. Cassandra—a distributed database system: An overview. In: ABRAHAM, A.; DUTTA, P.; MANDAL, J. K.; BHATTACHARYA, A.; DUTTA, S. (Ed.). *Emerging Technologies in Data Mining and Information Security*. Singapore: Springer Singapore, 2019. p. 519–526.

YAO, X.; WANG, C.-L. Probabilistic consistency guarantee in partial quorum-based data store. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 31, n. 8, p. 1815–1827, 2020.

YOO, J.; LEE, K.-H.; JEON, Y.-H. Migration from rdbms to nosql using column-level denormalization and atomic aggregates. *Journal of Information Science & Engineering*, v. 34, n. 1, 2018.

YUAN, X.; ZHOU, X.; PAN, Y.; KOSONEN, R.; CAI, H.; GAO, Y.; WANG, Y. Phase change cooling in data centers: a review. *Energy and Buildings*, Elsevier, v. 236, p. 110764, 2021.

ZAKARYA, M.; GILLAM, L. Energy efficient computing, clusters, grids and clouds: A taxonomy and survey. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 14, p. 13–33, 2017.

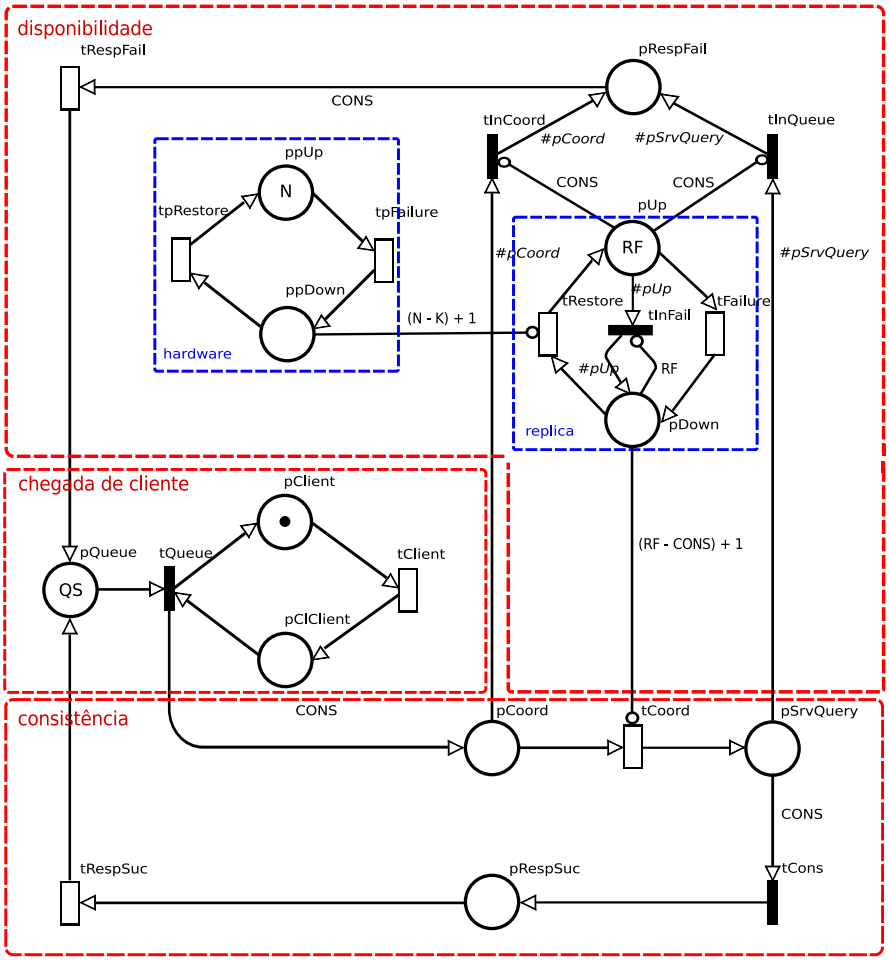
ZAKHAROV, N.; KLEPIKOV, V.; PODKHVATILIN, D. Network redundant dcs configuration management. *Advances in systems science and applications*, v. 20, n. 1, p. 83–90, 2020.

ANEXO A – EXEMPLO DE EXECUÇÃO DO MODELO DE DESEMPENHO E DISPONIBILIDADE

A seguir, apresenta-se um exemplo de execução do modelo de desempenho e disponibilidade. É considerada, inicialmente, uma requisição bem-sucedida; em seguida, apresentamos uma requisição com falha nos servidores que deveriam atender ao nível de consistência solicitado.

A.1 FLUXO REQUISIÇÃO BEM-SUCEDIDA

Figura 21 – (Anexo A) Modelo de desempenho e disponibilidade - 01

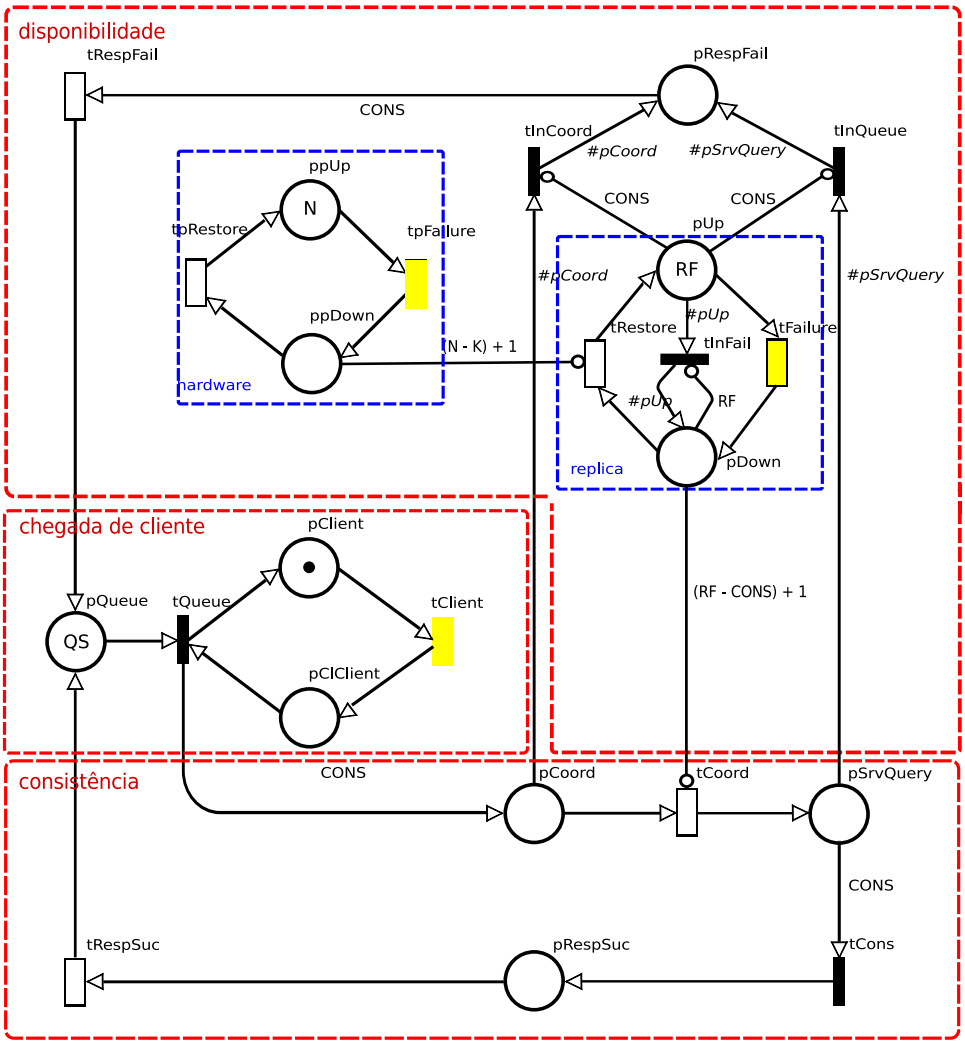


Fonte: Elaboração Própria.

A Figura 21 apresenta a marcação inicial do modelo de desempenho e disponibilidade. Considera-se nesse exemplo, uma operação com nível de consistência dois e fator de replicação

igual a três. Esse exemplo, considera uma requisição bem-sucedida, ou seja, a requisição será executada no nível de consistência solicitado. Em seguida, retornará ao cliente.

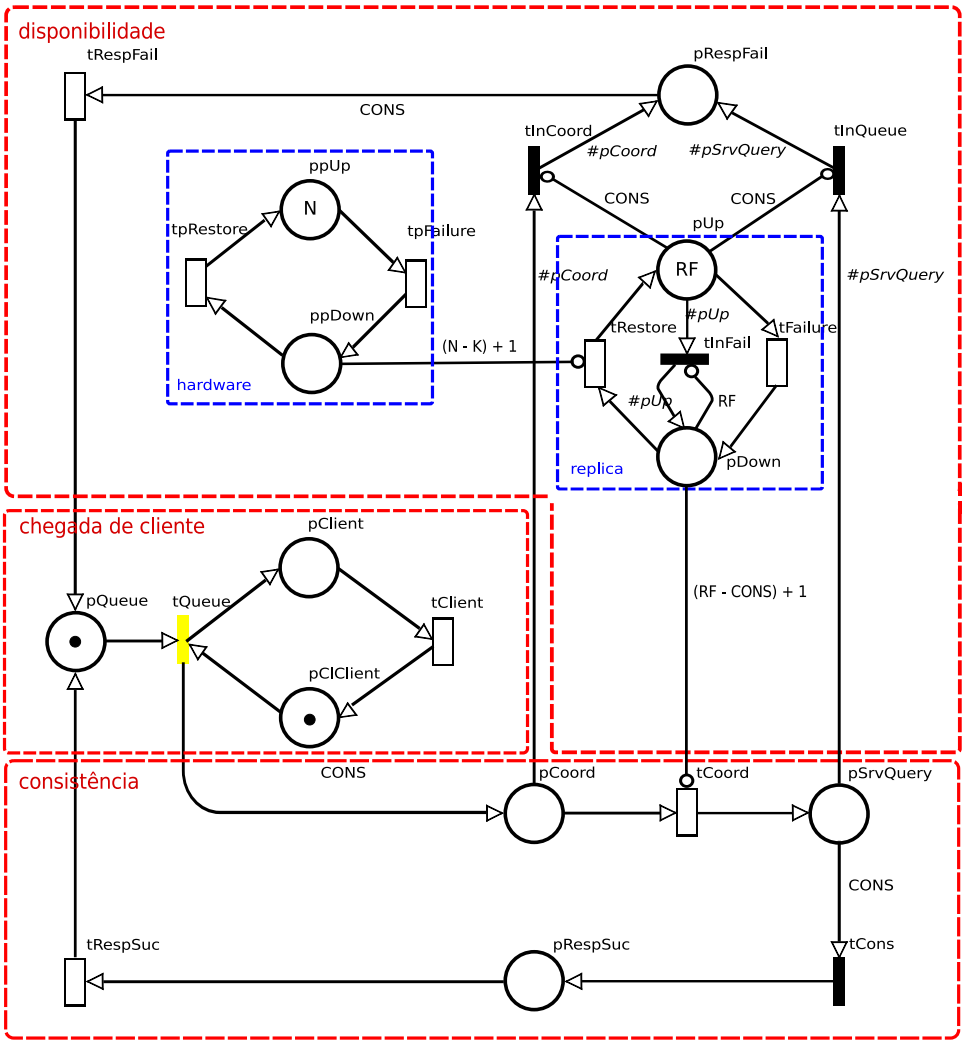
Figura 22 – (Anexo A) Modelo de desempenho e disponibilidade - 02



Fonte: Elaboração Própria.

No início da execução do modelo de desempenho e disponibilidade, são possíveis os disparos das transições: *tClient*, que representa a chegada de um cliente; *tpFailure*, que representa a falha de um servidor físico; e *tFailure*, que representa a falha de uma réplica do SGBD (Figura 22).

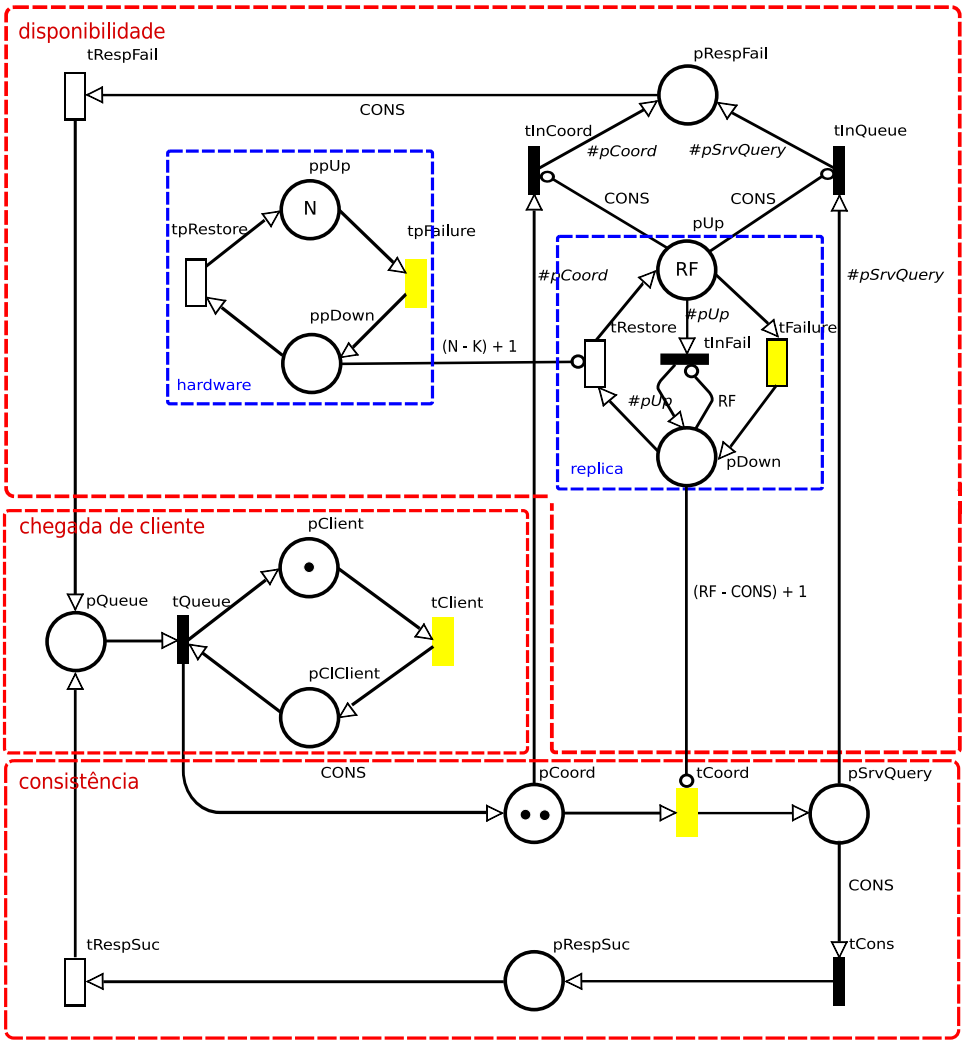
Figura 23 – (Anexo A) Modelo de desempenho e disponibilidade - 03



Fonte: Elaboração Própria.

Ocorreu o disparo da transição *tClient* (Figura 23). Dessa forma, é habilitada a transição *tQueue*. Essa transição é do tipo imediata, possui prioridade sobre as demais transições. Por isso, nenhuma outra transição pode disparar nesse momento.

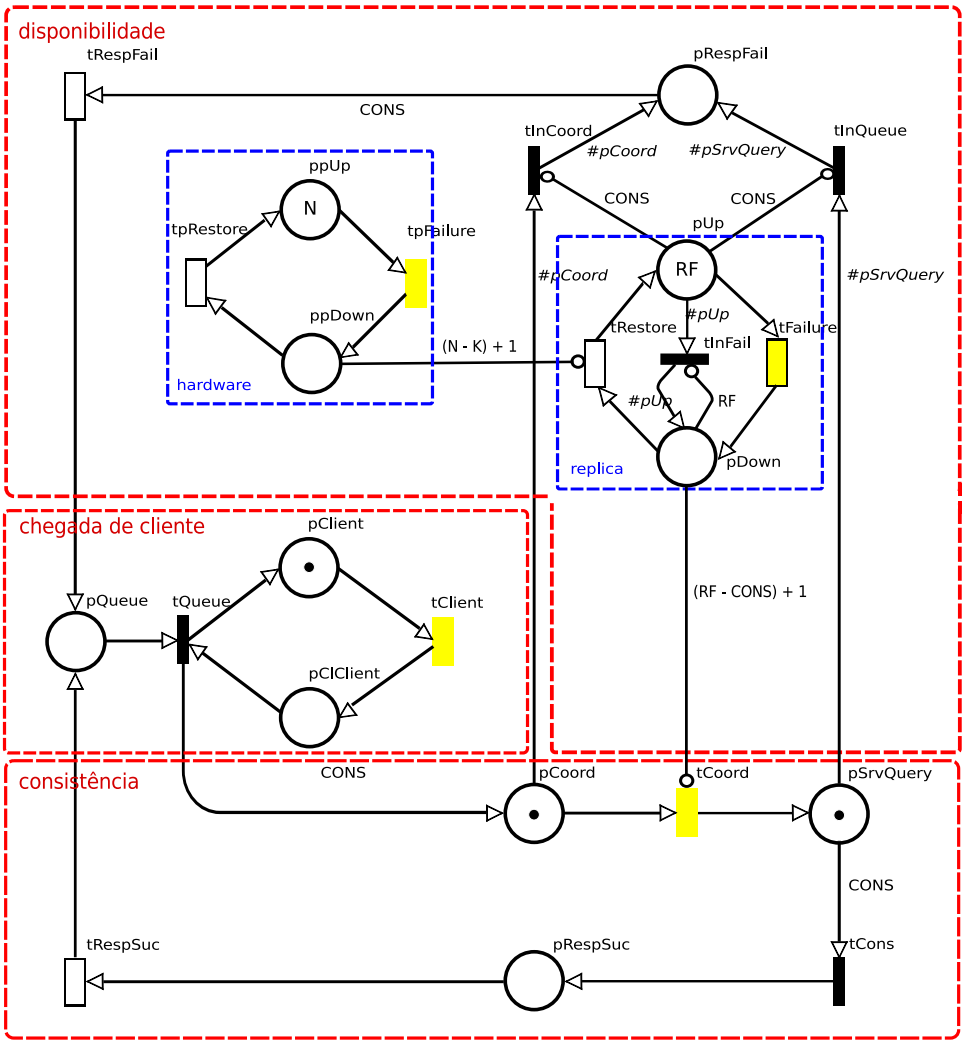
Figura 24 – (Anexo A) Modelo de desempenho e disponibilidade - 04



Fonte: Elaboração Própria.

Após disparo da transição *tQueue*, o lugar *pCoord* recebeu o número de *tokens* correspondente ao nível de consistência da requisição, que neste exemplo foi igual a dois (Figura 24). Agora as possibilidade são: o disparo da transição *tCoord*; o disparo da transição *tClient*; ou falha de um servidor físico ou réplica do SGBD por meio das transições *tpFailure* e *tFailure*.

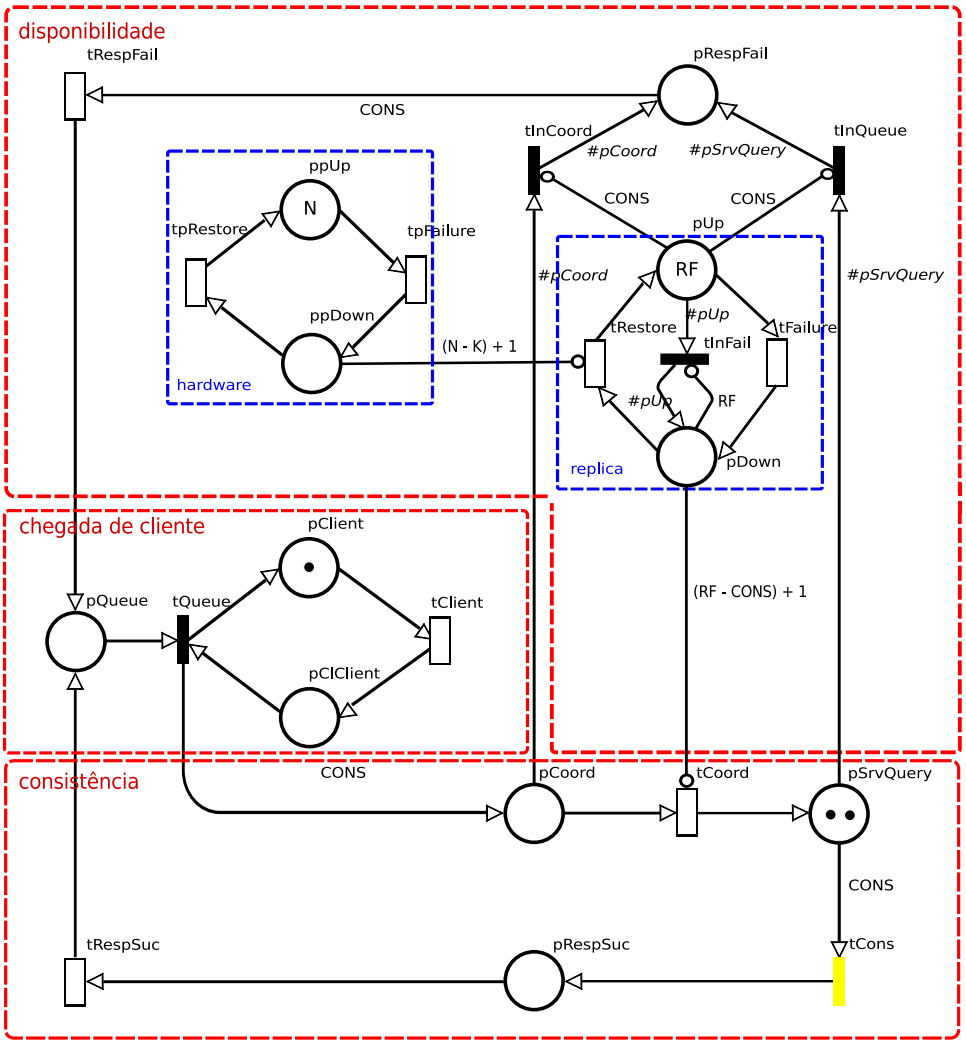
Figura 25 – (Anexo A) Modelo de desempenho e disponibilidade - 05



Fonte: Elaboração Própria.

A transição *tCoord*, que representa a comunicação entre as réplicas do SGBD, efetuada pelo servidor coordenador da requisição, realizou um disparo. O lugar *pSrvQuery* recebeu um *token*, significa que uma operação foi confirmada. Não houve mudança nas transições habilitadas (Figura 25).

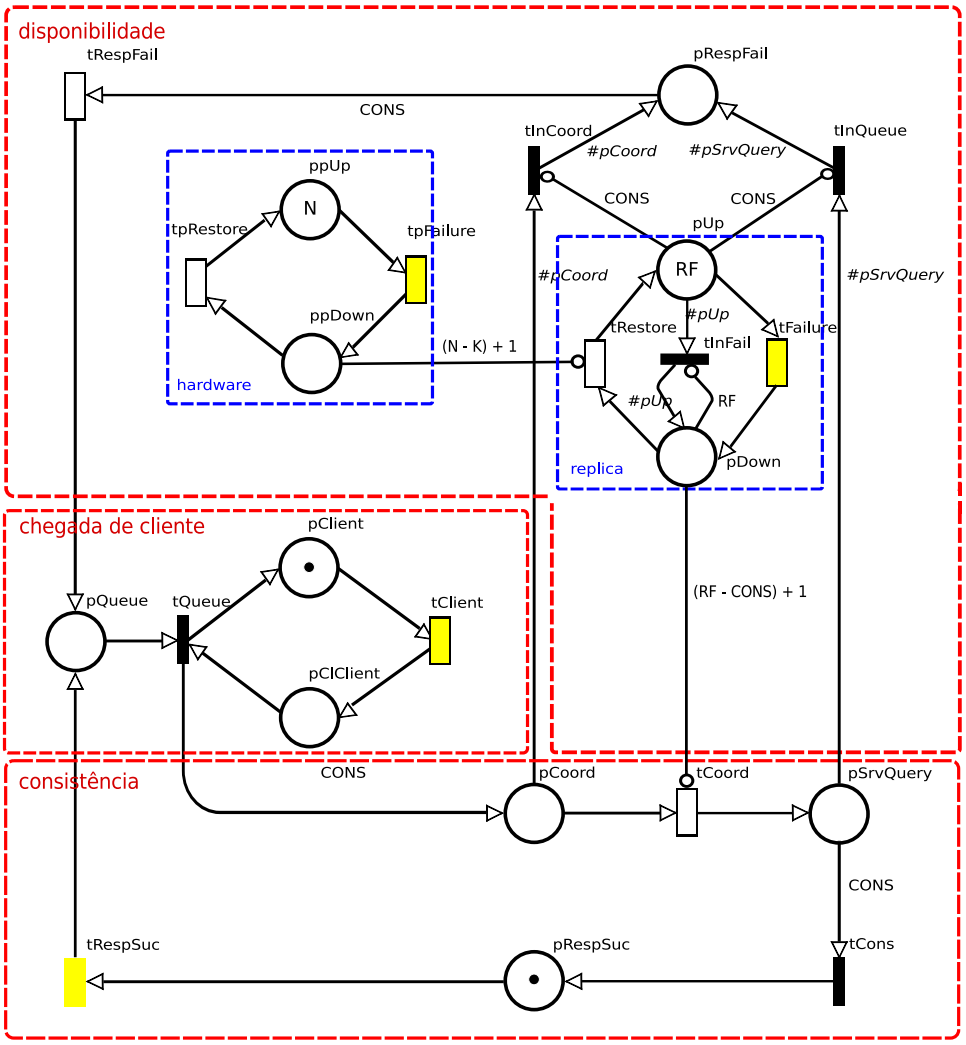
Figura 26 – (Anexo A) Modelo de desempenho e disponibilidade - 06



Fonte: Elaboração Própria.

Com o segundo disparo da transição *tCoord* (Figura 26), o lugar *pSrvQuery* recebeu o segundo *token* e somente agora a transição imediata *tCons* foi habilitada. Isso ocorreu, porque há *tokens* no lugar *pSrvQuery* igual ao nível de consistência da operação, ou seja, a requisição foi concluída conforme o nível de consistência solicitado. *tCons* é uma transição do tipo imediata, somente ela pode ser disparada nesse momento.

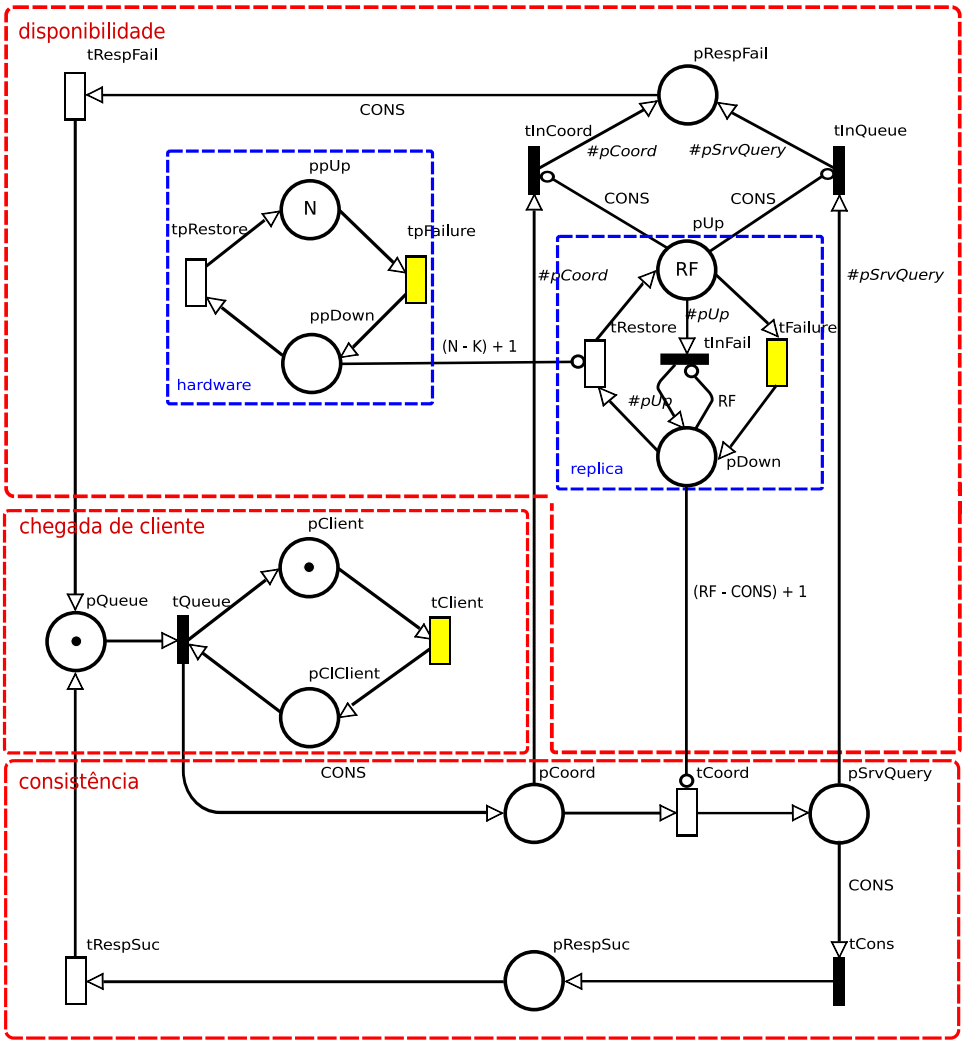
Figura 27 – (Anexo A) Modelo de desempenho e disponibilidade - 07



Fonte: Elaboração Própria.

Após o disparo da transição *tCons*, somente um *token* é enviado ao lugar *pRespSuc*. Isso representa a resposta da requisição a ser enviada ao cliente (Figura 27).

Figura 28 – (Anexo A) Modelo de desempenho e disponibilidade - 08

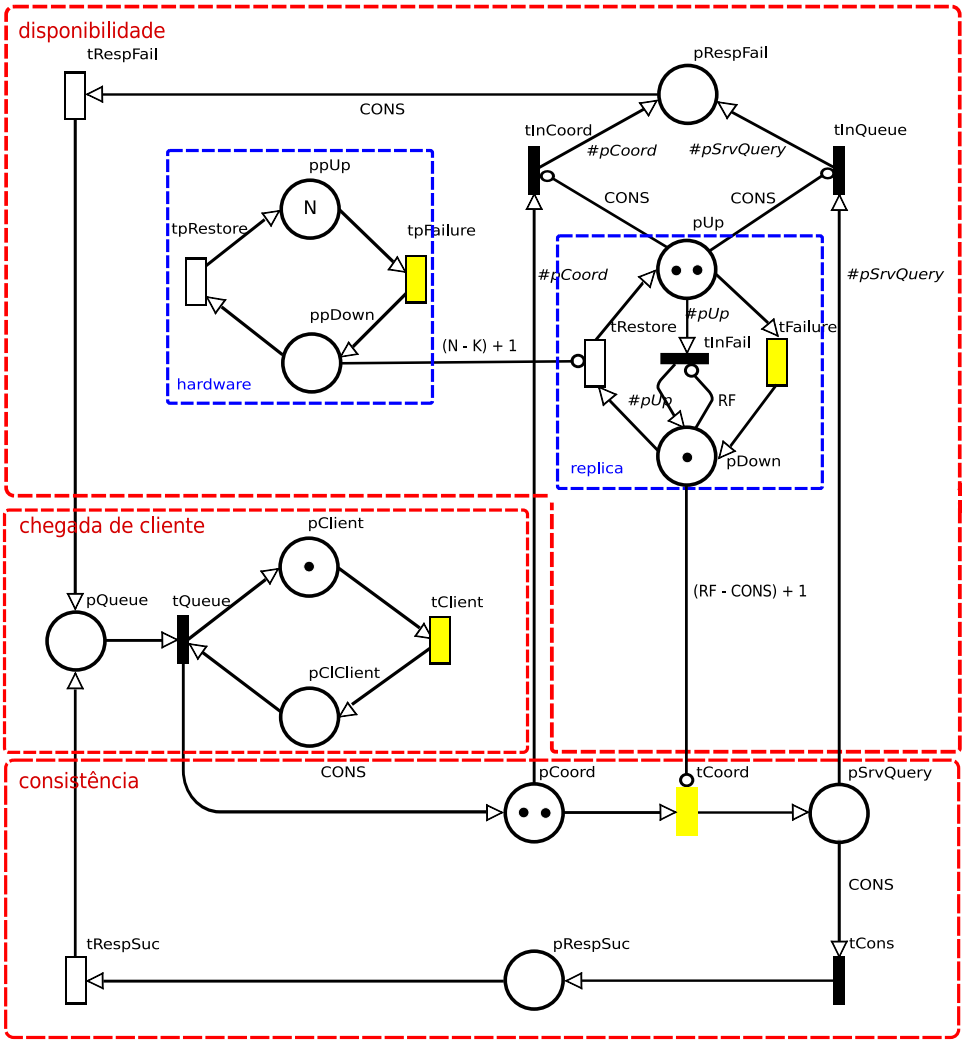


Fonte: Elaboração Própria.

A transição *tRespSuc* realizou o disparo, o tempo associado e ela é o tempo de comunicação entre as réplicas e o cliente, aferido no ambiente do SGBD. Após esse disparo o modelo retorna sua a marcação inicial (Figura 28).

A.2 FLUXO REQUISIÇÃO COM FALHA

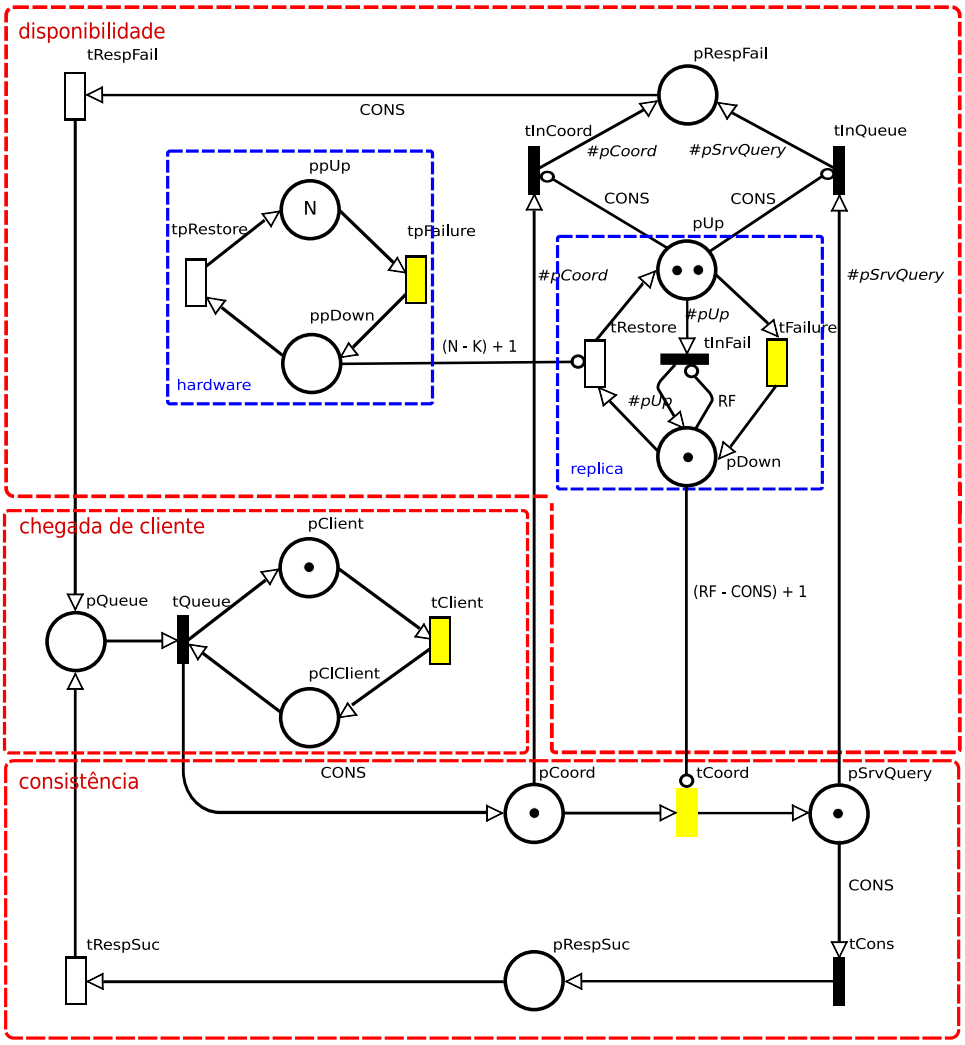
Figura 29 – (Anexo A) Modelo de desempenho e disponibilidade - 09



Fonte: Elaboração Própria.

A (Figura 29) representa o modelo em uma marcação onde o servidor coordenador já recebeu a requisição do cliente. Mesmo com a falha de uma réplica, representada pela existência de um *token* no lugar *pDown*, o SGBD está apto a realizar as operações. Isso é possível, porque ainda há servidores suficientes para confirmação da operação no nível de consistência solicitado, indicado no lugar *pUp*.

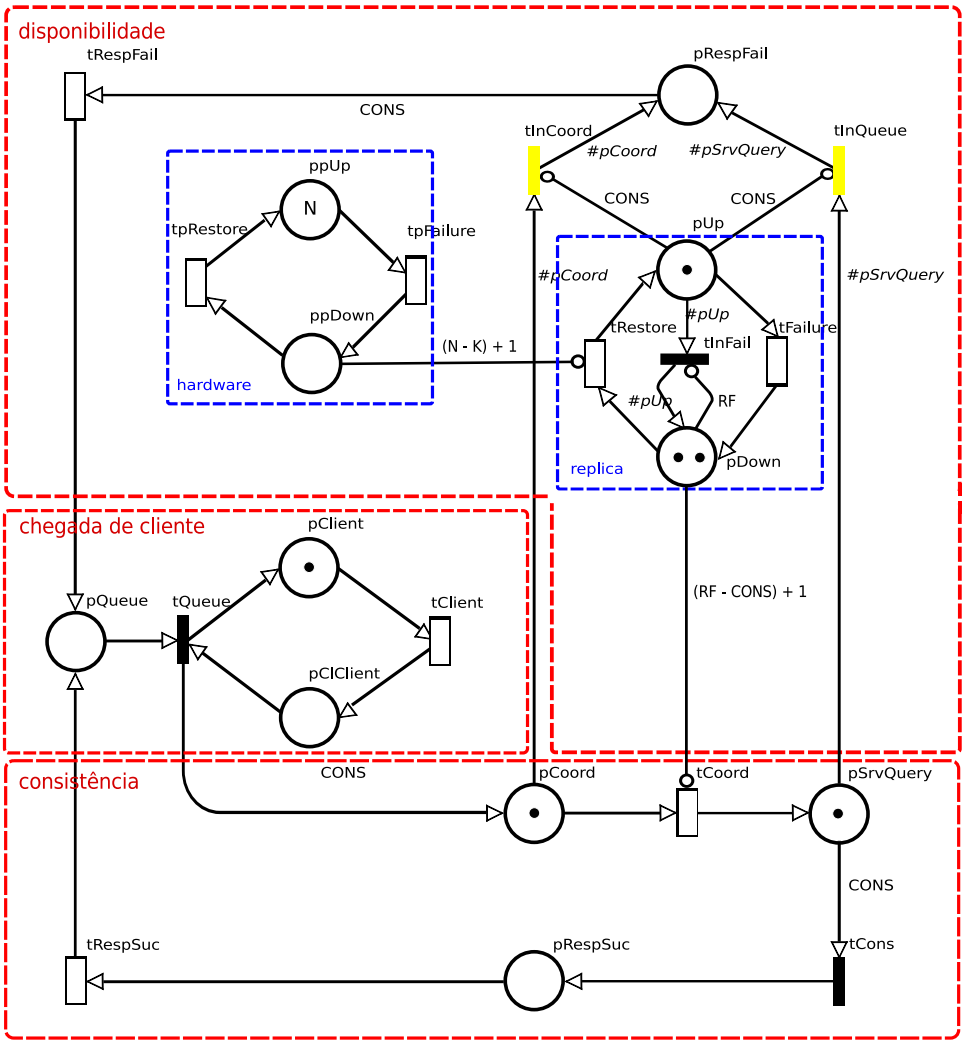
Figura 30 – (Anexo A) Modelo de desempenho e disponibilidade - 10



Fonte: Elaboração Própria.

O coordenador, representado pela transição *tCoord*, efetuou a primeira operação no *cluster* do SGBD (Figura 30).

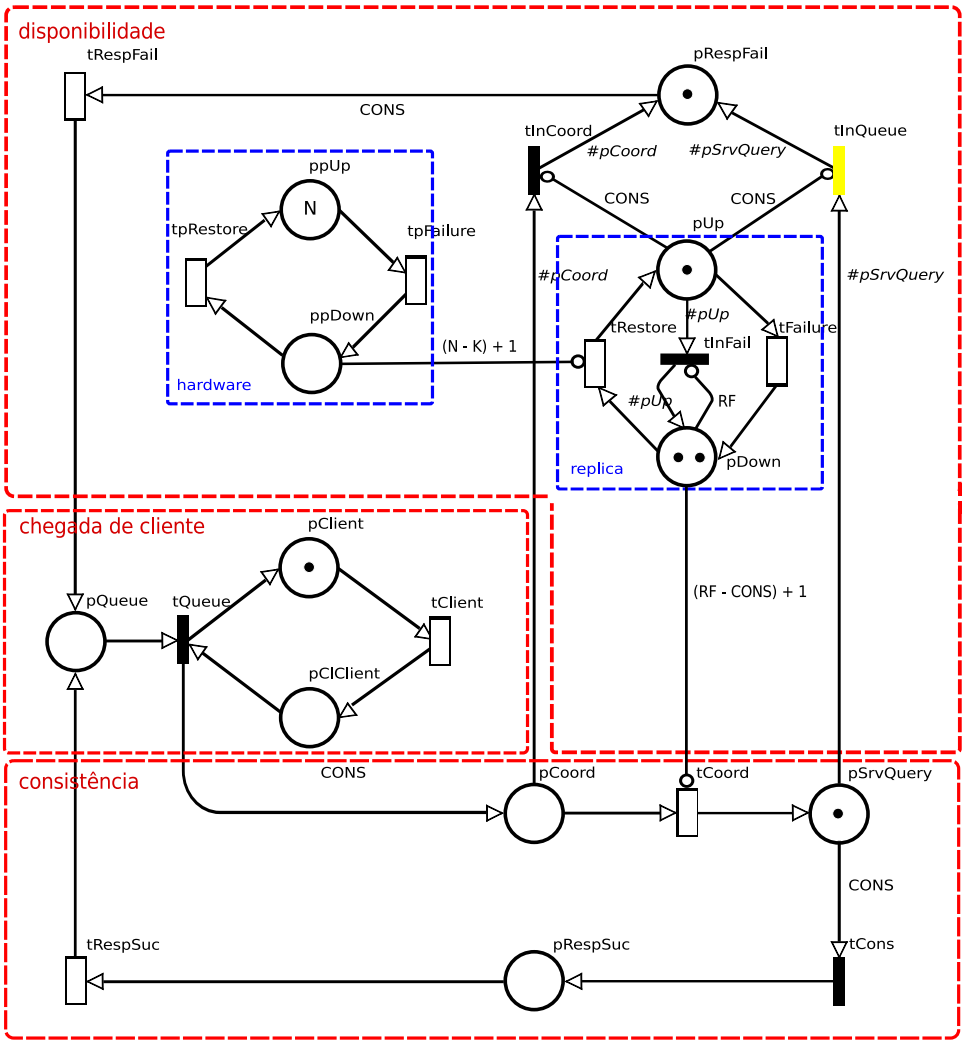
Figura 31 – (Anexo A) Modelo de desempenho e disponibilidade - 11



Fonte: Elaboração Própria.

Agora com a falha da segunda réplica, não há servidores suficientes para executar a operação no *cluster* do SGBD NoSQL (Figura 31). Dessa forma, as transições imediatas *tInCoord* e *tInQueue* são habilitadas por meio de arco inibidor com peso igual ao nível de consistência e de *token* nos lugares *pCoord* e *pSrvQuery*.

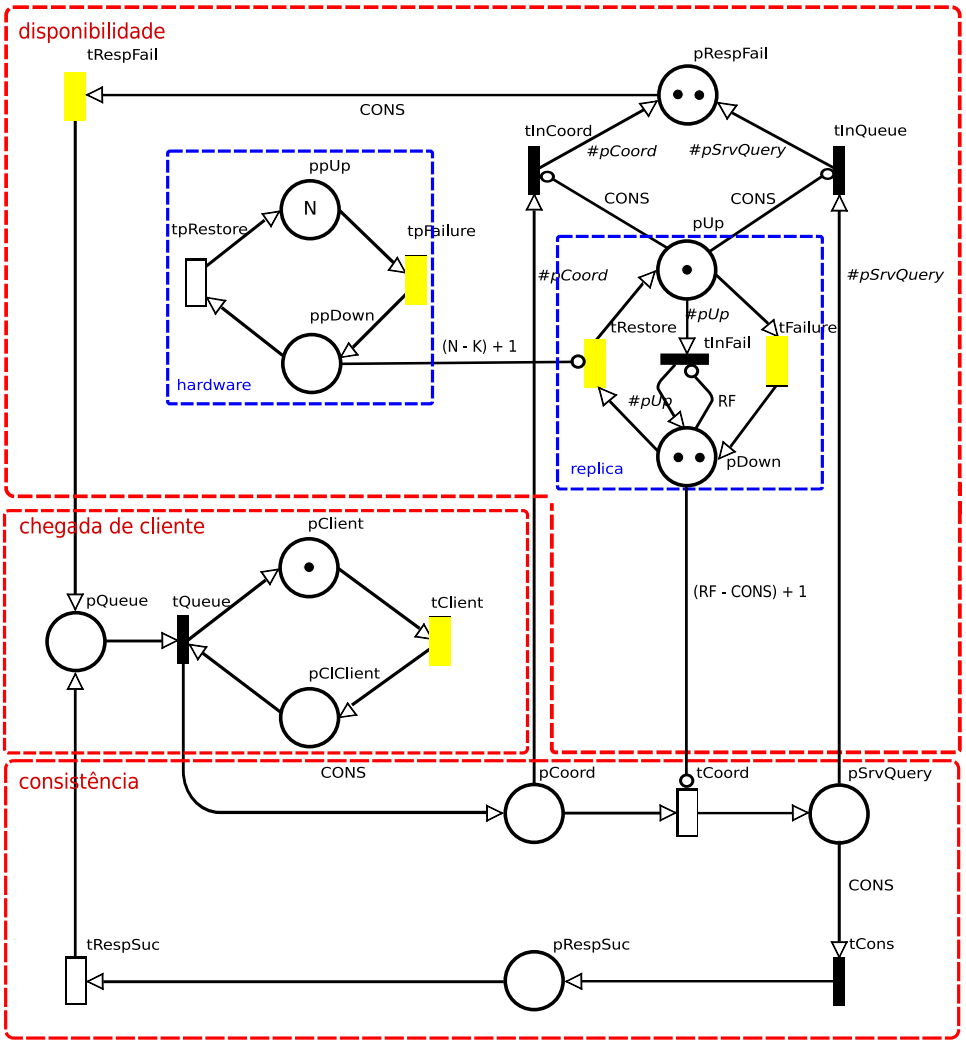
Figura 32 – (Anexo A) Modelo de desempenho e disponibilidade - 12



Fonte: Elaboração Própria.

A transição imediata *tInCoord* dispara e transfere, no mesmo disparo, todos os *tokens*, por meio de arcos com peso igual a *#pCoord* para o lugar *pRespFail*, o que denota a falha da operação (Figura 32).

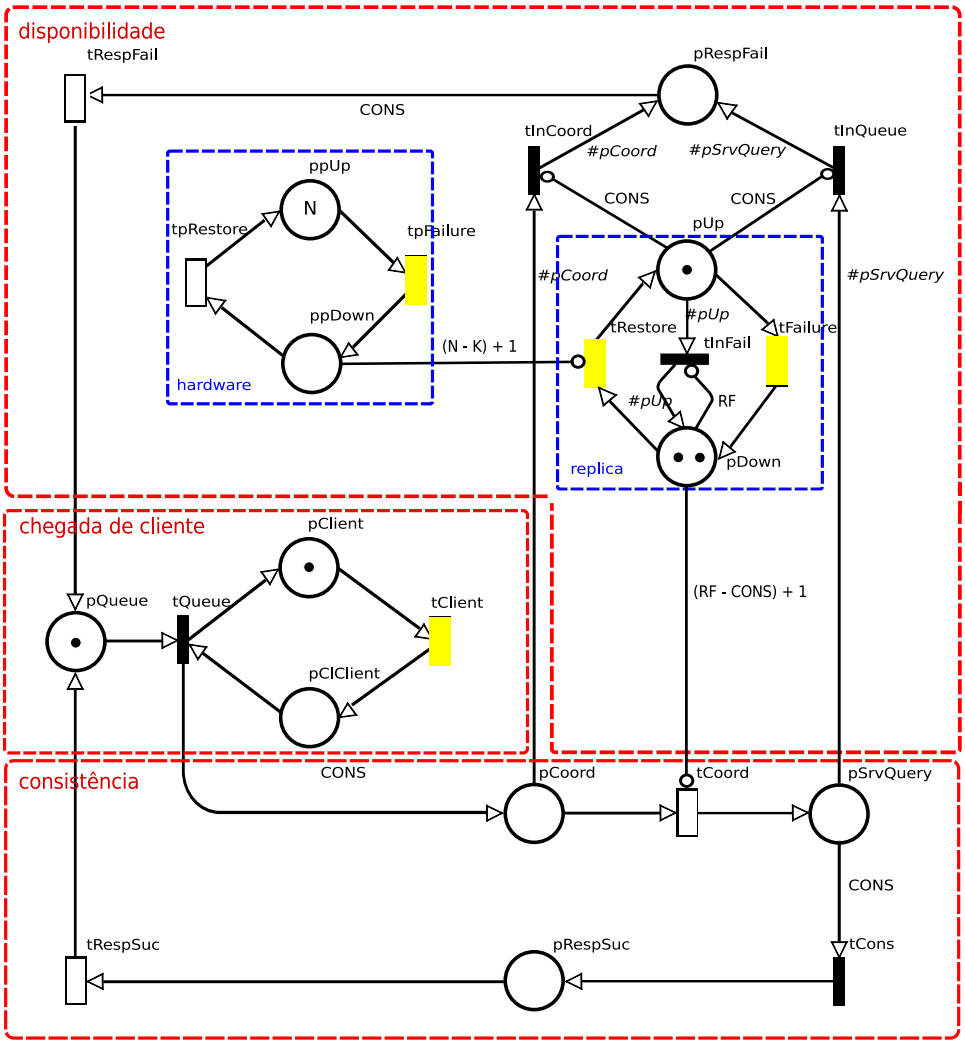
Figura 33 – (Anexo A) Modelo de desempenho e disponibilidade - 13



Fonte: Elaboração Própria.

Da mesma forma, a transição imediata *tInQueue* dispara e transfere, no mesmo disparo, todos os *tokens*, por meio de arcos com peso igual a *#pSrvQuery* para o lugar *pRespFail* (Figura 33). Conclui-se que a transição *tRespFail* foi habilitada, pois *tokens* iguais ao nível de consistência estão no lugar *pRespFail*.

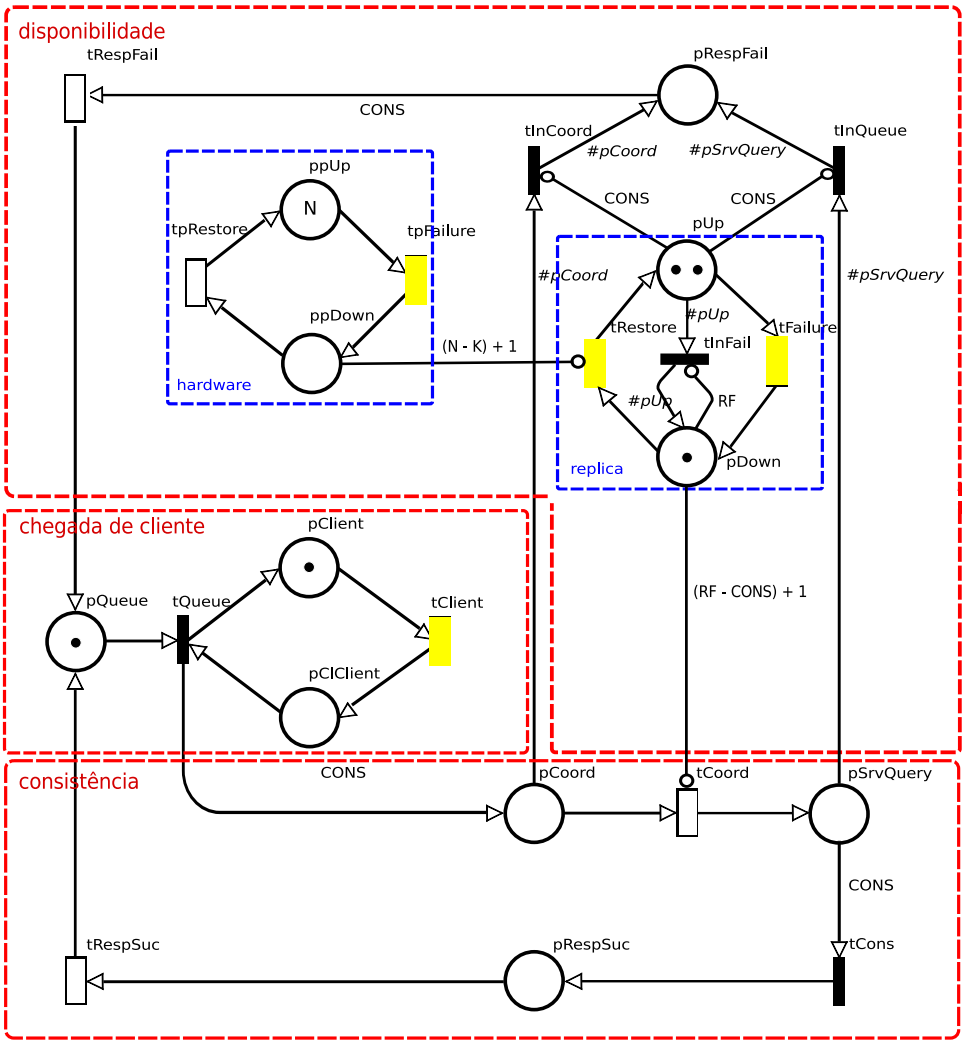
Figura 34 – (Anexo A) Modelo de desempenho e disponibilidade - 14



Fonte: Elaboração Própria.

Após o disparo da transição *tRespFail*, o cliente recebe a resposta de falha da requisição. Uma nova requisição pode ser realizada, uma outra replica pode falhar, ou ainda, uma ou duas réplicas podem ser restauradas pela transição *tRestore* (Figura 34).

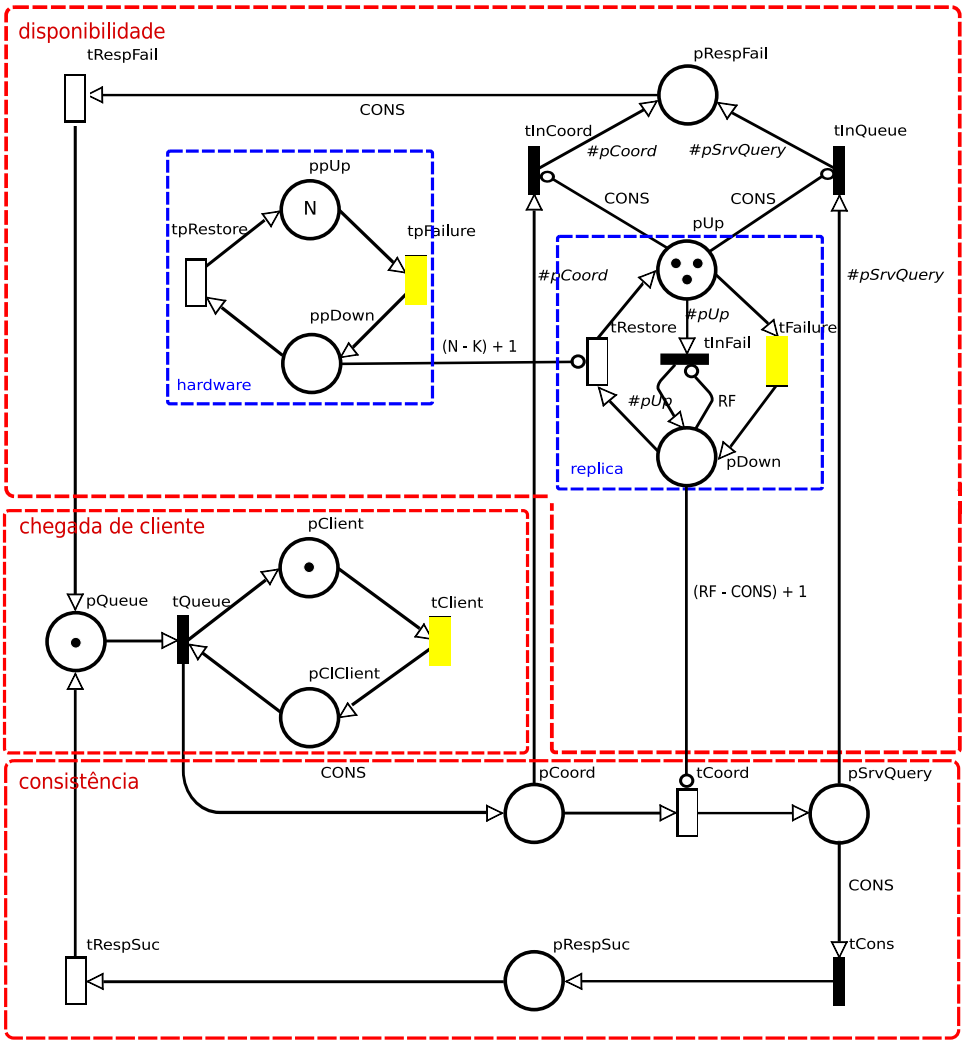
Figura 35 – (Anexo A) Modelo de desempenho e disponibilidade - 15



Fonte: Elaboração Própria.

A transição *tRestore* disparou e restaurou uma réplica, permitindo que o nível de consistência dois da operação fosse atendido (Figura 35).

Figura 36 – (Anexo A) Modelo de desempenho e disponibilidade - 16



Fonte: Elaboração Própria.

Outro disparo da transição *tRestore*, restaurou a réplica remanescente com falha, dessa forma, voltamos à marcação inicial do modelo (Figura 36).

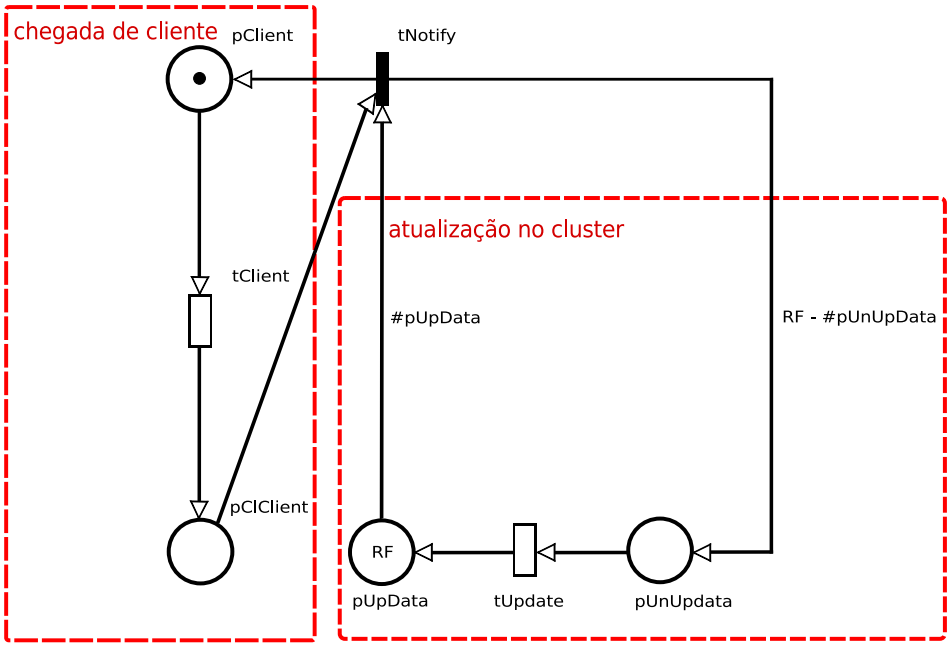
O exemplo demonstrado neste Anexo A apresenta uma possibilidade de fluxo em requisições do NoSQL baseado em quorum; outros níveis de consistência e fatores de replicação podem ser avaliados conforme o capítulo 5, que contém os detalhes do modelo.

ANEXO B – EXEMPLO DE EXECUÇÃO DO MODELO DE ACESSO AO DADO

A seguir, apresenta-se um exemplo de execução do modelo de acesso ao dado. Esse modelo não considera falha no sistema.

B.1 FLUXO EXECUÇÃO DO MODELO DE ACESSO AO DADO

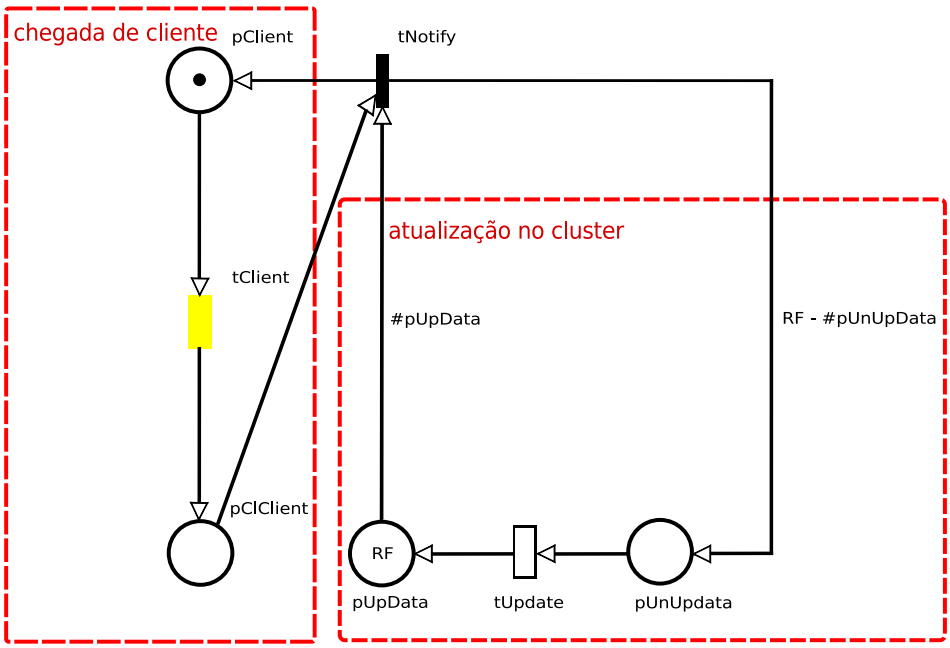
Figura 37 – (Anexo B) Modelo de acesso ao dado - 01



Fonte: Elaboração Própria.

A Figura 37 apresenta a marcação inicial do modelo de acesso ao dado; considera-se, nesse exemplo, uma operação com nível de consistência dois.

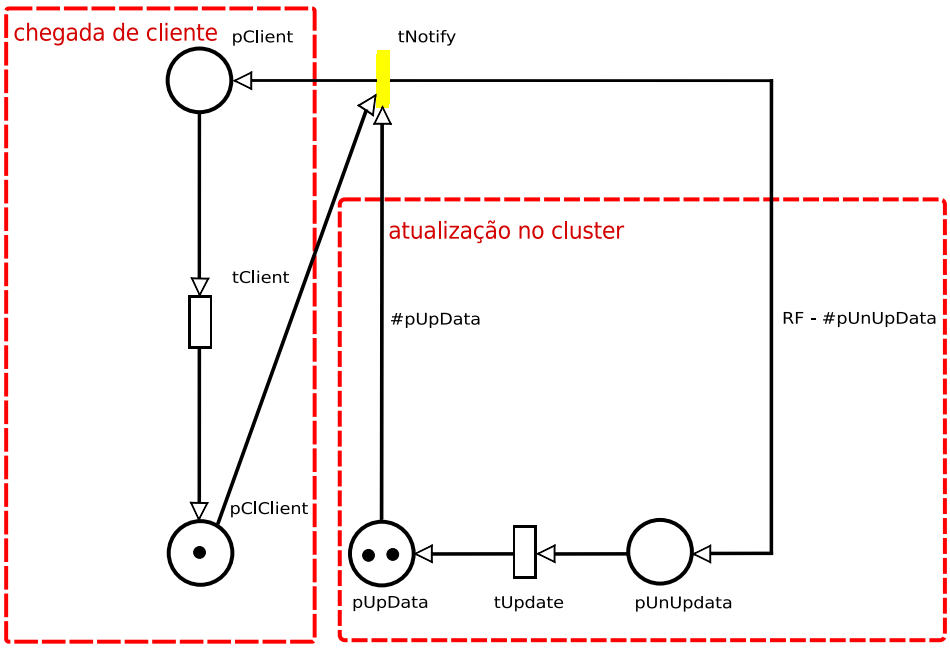
Figura 38 – (Anexo B) Modelo de acesso ao dado - 02



Fonte: Elaboração Própria.

No início da execução deste modelo, apenas é possível o disparo da transição *tClient*, que representa a chegada de um cliente (Figura 38). Considera-se, neste modelo, que inicialmente todos os SGBDs estão atualizados, uma vez que não houve uma requisição prévia.

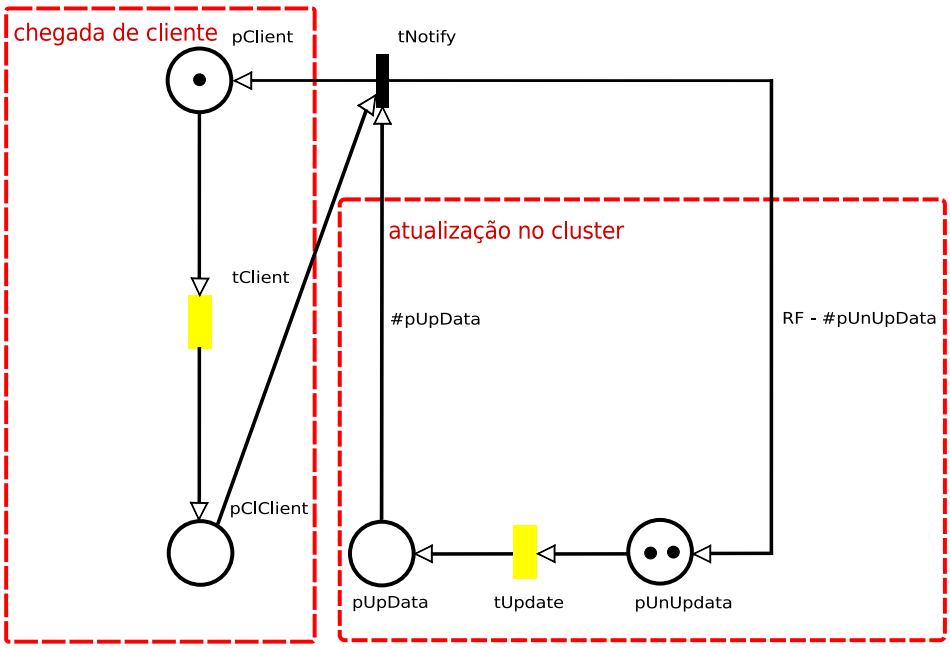
Figura 39 – (Anexo B) Modelo de acesso ao dado - 03



Fonte: Elaboração Própria.

Após o disparo da transição *tClient*, a transição imediata *tNotify* é habilitada, nenhuma outra transição pode efetuar um disparo nesse momento (Figura 39).

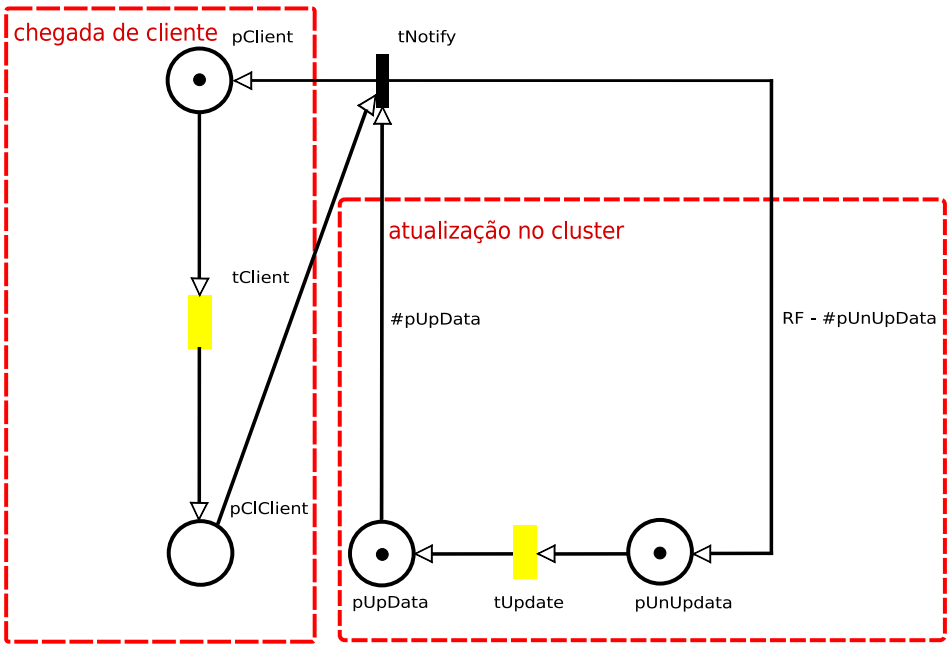
Figura 40 – (Anexo B) Modelo de acesso ao dado - 04



Fonte: Elaboração Própria.

Após o disparo da transição *tNotify*, todos os *tokens* são enviados do lugar *pUpData*, que representa as réplicas atualizadas, para o lugar *pUnUpdata*, que denota as réplicas desatualizadas (Figura 40). Agora uma nova requisição pode chegar, pois o lugar *pClient* também recebeu o *token* do lugar *pClClient* ou o *cluster* do SGBD e pode, assim, iniciar a replicação dos dados.

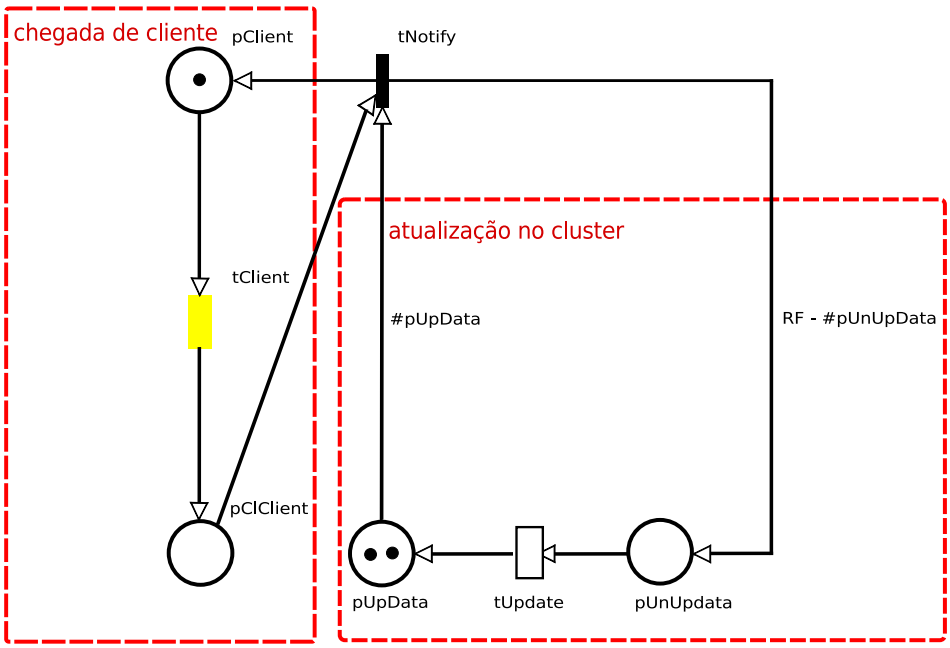
Figura 41 – (Anexo B) Modelo de acesso ao dado - 05



Fonte: Elaboração Própria.

A transição *tUpdate* realiza um disparo que representa a atualização do dado em um servidor (Figura 41). Agora as possibilidade são a replicação do segundo dado ou a chegada de uma nova requisição.

Figura 42 – (Anexo B) Modelo de acesso ao dado - 06



Fonte: Elaboração Própria.

Com o novo disparo da transição *tUpdate*, a atualização do segundo dado em um servidor é realizada (Figura 42). Dessa forma, o modelo está novamente em sua marcação inicial.

O exemplo demonstrado neste Anexo B demonstra uma possibilidade de fluxo do modelo de acesso ao dado em requisições do NoSQL baseado em quorum. Detalhes do modelo estão contidos no capítulo 5.