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

FILLIPE CELESTINO DIAS SOUZA

Título: Modelagem Estocástica para Estimativa de Consumo de Energia, Desempenho e Disponibilidade em Sistemas de Armazenamento Baseados em NoSQL

Recife

2025

FILLIPE CELESTINO DIAS SOUZA

Título: Modelagem Estocástica para Estimativa de Consumo de Energia, Desempenho e Disponibilidade em Sistemas de Armazenamento Baseados em NoSQL

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Redes de Computadores e Sistemas Distríbuidos

Orientador (a): Eduardo Antonio Guimarães Tavares

Coorientador (a): Carlos Gomes Araújo

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Souza, Fillipe Celestino Dias.
   Modelagem estocástica para estimativa de consumo de energia, desempenho e disponibilidade em sistemas de armazenamento baseados em NoSQL / Fillipe Celestino Dias Souza. - Recife, 2025.
   116f.: il.

   Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciência da Computação,2025.
   Orientação: Eduardo Antonio Guimarães Tavares.
   Coorientação: Carlos Gomes Araújo.
   Inclui referências e apêndices.

   1. NoSQL; 2. SGBD; 3. SPN; 4. Desempenho; 5. Disponibilidade; 6. Consumo de Energia. I. Tavares, Eduardo Antonio Guimarães. II. Araújo, Carlos Gomes. III. Título.

UFPE-Biblioteca Central

Fillipe Celestino Dias Souza

“Modelagem Estocástica para Estimativa de Consumo de Energia, Desempenho e Disponibilidade em Sistemas de Armazenamento Baseados em NoSQL”

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

Aprovado em: 26/02/2025.

BANCA EXAMINADORA

_____

Prof. Dr. Jamilson Ramalho Dantas
Centro de Informática / UFPE

_____

Profa. Dra. Erica Teixeira Gomes de Sousa
Departamento de Computação / UFRPE

_____

Prof. Dr. Eduardo Antônio Guimarães Tavares
Centro de Informática / UFPE
(**orientador**)

Dedico a Deus, meus pais, meu irmão e à minha amada esposa.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por esta grande conquista em minha vida e pela força que me sustentou durante os desafios enfrentados.

À minha amada esposa, Thuany Camilly, cujo amor, compreensão e apoio foram essenciais. Você foi minha fortaleza em todos os momentos deste percurso.

Aos meus pais, Paulo Santana e Deborah Celestino, expresso minha gratidão. Seu apoio constante e incentivo desde o início foram a base que me manteve firme diante das adversidades.

Manifesto também minha sincera gratidão ao meu orientador, Dr. Eduardo Tavares, por sua orientação sólida, paciência e encorajamento ao longo do processo. Sua experiência e dedicação foram cruciais para o desenvolvimento deste trabalho.

Agradeço ao meu coorientador, Dr. Carlos Gomes, cujas contribuições e perspectivas enriqueceram significativamente esta pesquisa. Sua orientação complementar foi fundamental para alcançar os resultados apresentados.

Não posso deixar de mencionar meu primo, Lucas Farias, que esteve sempre ao meu lado, oferecendo suporte, encorajamento e partilhando sua experiência. Sou profundamente grato por seu apoio.

Por fim, agradeço ao SENAI e a todos os amigos que fiz nesta instituição. O suporte técnico e emocional que recebi de vocês foi de imenso valor e contribuiu para que eu continuasse empenhado neste trabalho.

”Nossa maior fraqueza está em desistir. A maneira mais certa de ter sucesso é sempre tentar mais uma vez. -"Thomas Edison”.

RESUMO

Em sistemas de armazenamento de dados em larga escala, os Sistemas de Gerenciamento de Banco de Dados NoSQL são amplamente adotados por sua capacidade de fornecer melhor desempenho e disponibilidade. Uma característica importante é o suporte à consistência eventual, que permite uma inconsistência temporária de dados em algumas réplicas do banco de dados. Diferentes níveis de consistência podem ser adotados, permitindo ao projetista ajustar sua influência no desempenho e na disponibilidade do sistema. Embora algumas pesquisas proponham técnicas para estimar o impacto da consistência eventual no desempenho e na disponibilidade, o consumo de energia geralmente não é considerado. Este trabalho propõe um modelo baseado em redes de Petri estocásticas para estimar o consumo de energia, desempenho e disponibilidade de sistemas NoSQL que adotam a técnica de quorum. A abordagem avalia distintos níveis de consistência, fatores de replicação e operações de leitura e escrita. Os resultados experimentais demonstram que $cons$ e RF podem afetar o consumo de energia em até 10%. Para um fator de replicação de 5 (RF=5), a diminuição do nível de consistência de 3 para 1 reduz o tempo de inatividade em 87,50% (de 0,72 para 0,09 horas) e reduz o consumo de energia em 2,72%. Reduzir o nível de consistência de 3 para 1 resulta em uma diminuição de aproximadamente 9% no consumo de energia, sem comprometer significativamente o desempenho do sistema. Esses resultados demonstram que o modelo proposto permite avaliar o consumo de energia, desempenho e disponibilidade, ajudando o projetista a escolher a configuração mais adequada para o sistema.

Palavras-chaves: NoSQL. SGBD. SPN. Consistência Eventual. Desempenho. Disponibilidade. Consumo de Energia.

ABSTRACT

In large-scale data storage systems, NoSQL Database Management Systems are widely adopted for their ability to provide better performance and availability. An important feature is the support for eventual consistency, which allows for temporary data inconsistency in some database replicas. Different levels of consistency can be adopted, enabling the designer to adjust their influence on system performance and availability. Although some research proposes techniques to estimate the impact of eventual consistency on performance and availability, energy consumption is often overlooked. This study proposes a model based on stochastic Petri nets to estimate the energy consumption of NoSQL systems that adopt the quorum technique. The approach evaluates distinct consistency levels, replication factors, and read and write operations. Experimental results demonstrate that $cons$ and RF can impact energy consumption by up to 10%. For a replication factor of 5 (RF=5), reducing the consistency level from 3 to 1 decreases downtime by 87.50% (from 0.72 to 0.09 hours) and reduces energy consumption by 2.72%. Lowering the consistency level from 3 to 1 results in approximately a 9% reduction in energy consumption without significantly compromising system performance. These results demonstrate that the proposed model allows for the evaluation of energy consumption, downtime, and latency, assisting the designer in selecting the most suitable configuration for the system.

Keywords: NoSQL. DBMS. SPN. Eventual Consistency. Performance. Availability. Energy Consumption.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID Atomicity, Consistency, Isolation, Durability

AWS Amazon Web Services

BASE Basically Available, Soft state, Eventually consistent

CPN Colored Petri Nets

CTMC Cadeia de Markov de Tempo Contínuo

DoE Design of Experiments

GSPN Generalized Stochastic Petri Nets

HDFS Hadoop Distributed File System

HSDC Hyper-Scale Data Centers

IS Infinite Server

MER Modelo de Entidade Relacionamento

MTTF Mean Time to Failure

MTTR Mean Time to Repair

NoSQL Not Only SQL

PBS Probabilistically Bounded Staleness

PCG Probabilistic Consistency Guarantee

PN Petri Nets

QN Queueing Networks

RBD Reliability Block Diagrams

SAN Storage Area Network

SGBD Sistema de Gerenciamento de Banco de Dados

SMSDC Small and Medium-Scale Data Centers

SPN Stochastic Petri Nets

SS Single Server

YCSB Yahoo! Cloud Serving Benchmark

LISTA DE SÍMBOLOS

Δ	Delta
λ	Lambda
Π	Número Pi
$\in$	Pertence
∞	Infinito
$\geq$	Maior ou Igual
$\leq$	Menor ou Igual
$>$	Maior que
$<$	Menor que
$=$	Igualdade
$\mathbb{N}$	Número Natural
$\mathbb{R}^+$	Números Reais Positivos
$\sum$	Somatório

SUMÁRIO

1 INTRODUÇÃO

A expansão das mídias sociais, da computação em nuvem (*cloud computing*) e da Internet das Coisas gerou um aumento significativo na produção e armazenamento de dados. Esses dados, de grande volume e complexidade, apresentam fontes e tipos heterogêneos, sendo caracterizados pelo conceito de *Big Data*, que abrange três dimensões principais. O volume refere-se à enorme quantidade de dados. A variedade está relacionada à diversidade entre dados estruturados, semiestruturados e não estruturados. A velocidade diz respeito à rapidez com que os dados são gerados e processados (DAVOUDIAN; CHEN; LIU, 2018; BOUAZIZ; NABLI; GARGOURI, 2020).

Para lidar com os desafios do *Big Data*, surgiram os Sistema de Gerenciamento de Banco de Dados (SGBD)s Not Only SQL (NoSQL), que possuem esquemas flexíveis e permitem uma abordagem eficiente para lidar com alterações na estrutura dos dados e suas categorias (STOREY; SONG, 2017). Além disso, esses sistemas facilitam a replicação e o escalonamento de servidores, diferentemente dos bancos de dados relacionais (SAHATQIJA et al., 2018; PALANISAMY; SUVITHAVANI, 2020).

Os SGBDs relacionais seguem os princípios Atomicity, Consistency, Isolation, Durability (ACID), que garantem transações seguras e consistentes, mas podem comprometer a disponibilidade e o desempenho em sistemas distribuídos. Já os SGBDs NoSQL adotam o acrônimo Basically Available, Soft state, Eventually consistent (BASE), priorizando a disponibilidade, mesmo com possíveis inconsistências temporárias (CHANDRA, 2015). Provedores como Microsoft Azure e Amazon Web Services (AWS) utilizam soluções NoSQL, como o DynamoDB, para otimizar o desempenho em ambientes distribuídos (KHAN et al., 2023).

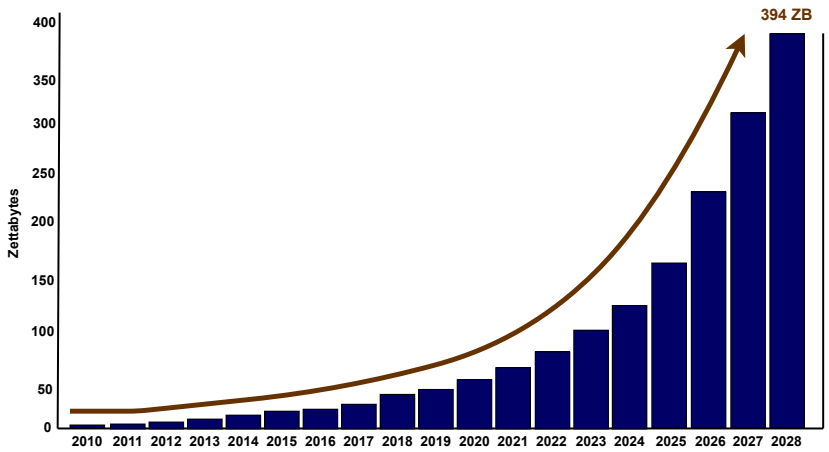
Para atender aos requisitos do sistema, alguns SGBDs NoSQL adotam a consistência baseada em quorum. Essa técnica define o número mínimo de réplicas necessárias para confirmar o sucesso de uma operação (GIFFORD, 1979). Plataformas financeiras, por exemplo, exigem altos níveis de consistência para garantir precisão nas transações, enquanto redes sociais priorizam alta disponibilidade com menor consistência (SHI et al., 2020; HSU; KSHEMKALYANI, 2021).

Embora eficiente, o ajuste dos níveis de consistência em sistemas NoSQL como Cassandra e DynamoDB impacta consumo de energia, disponibilidade e desempenho, exigindo equilíbrio para atender aos requisitos do sistema (NOUDOUST; ADABI; REZAEE, 2022; SOBOL, 2021; GAO et al., 2021).

1.1 MOTIVAÇÃO

Estudos demonstram que a criação, captura e consumo de dados estão crescendo de forma acelerada. Projeta-se que, até 2028, o armazenamento global de dados poderá ultrapassar 394 *zettabytes* (Figura 1). Enquanto para 2019, o total armazenado foi de 41 *zettabytes* (Worldwide, IDC, 2024; ERYUREK et al., 2021).

Figura 1 – Projeção de crescimento da geração de dados



Fonte: adaptado de (ERYUREK et al., 2021; Worldwide, IDC, 2024)

Com a crescente demanda por armazenamento, o consumo de energia em SGBDs NoSQL tornou-se uma preocupação significativa. *Data centers*, responsáveis por esse armazenamento, podem consumir até três vezes mais energia do que a geração total do Japão até 2050 (IIMURA et al., 2015). Até 2030, o consumo de energia desses *Data centers* poderá representar 13% da eletricidade global, com Hyper-Scale Data Centers (HSDC) consumindo 5% e Small and Medium-Scale Data Centers (SMSDC) os 95% restantes (DEEPIKA; DHANYA, 2023). Nos Estados Unidos, SMSDCs abrigam 40% dos 5,17 milhões de servidores, destacando a importância de estratégias para reduzir o consumo energético.

Pesquisas recentes analisam SGBDs NoSQL em relação ao consumo de energia, desempenho e disponibilidade (GUO et al., 2024; GOMES et al., 2023; FALCãO et al., 2024). Na consistência baseada em quorum, níveis mais baixos exigem menos comunicação entre réplicas, reduzindo a latência e o consumo de energia. No entanto, podem comprometer a disponibilidade em caso de falhas, pois menos réplicas participam da operação. Níveis mais altos exigem a participação de mais réplicas, o que aumenta a redundância e a consistência, mas eleva o consumo de energia e pode impactar o desempenho. Assim, equilibrar eficiência energética e qualidade dos

serviços torna-se um desafio (NOUDOUST; ADABI; REZAEE, 2022; GOMES et al., 2023; SHAH; KOTHARI; PATEL, 2022).

1.2 OBJETIVOS

Este trabalho propõe uma metodologia baseada em experimentação, testes e modelagem para avaliar o consumo de energia, desempenho e disponibilidade em SGBDs NoSQL com consistência baseada em quorum, auxiliando projetistas na escolha de configurações que atendam aos requisitos do sistema.

O modelo é generalizado para SGBDs que utilizam consistência quorum, sendo validado no banco de dados Cassandra devido à sua ampla documentação (CASSANDRA, 2024). A abordagem permite avaliar operações de leitura e escrita, considerando tempo de resposta, tempos médios de falhas e reparo das réplicas e servidores. Resultados experimentais indicam que o nível de consistência impacta significativamente o consumo de energia.

Os objetivos específicos desta dissertação são:

- Propor uma metodologia formal baseada em redes de Petri estocásticas para avaliar o consumo de energia, desempenho e a disponibilidade de SGBDs NoSQL que adotam a consistência quorum;
- Investigar os efeitos de fatores que influenciam consumo de energia, desempenho e disponibilidade dos SGBDs NoSQL, utilizando o Design of Experiments (DoE) (MONTGOMERY, 2017);
- Desenvolver um modelo que possibilite estimar e analisar o consumo de energia, desempenho e disponibilidade, considerando diferentes configurações em bancos de dados NoSQL que adotam a consistência quorum;
- Criar e validar cenários experimentais que demonstrem a viabilidade da metodologia proposta, possibilitando que o modelo proposto seja aplicável a diferentes configurações.

1.3 CONTRIBUIÇÕES

As principais contribuições deste trabalho resultam da aplicação da metodologia proposta para avaliar a consistência eventual e reduzir o consumo de energia em infraestruturas de

nuvem que utilizam SGBDs NoSQL com consistência baseada em quorum. Destacam-se:

- Proposição de uma metodologia para avaliar o consumo de energia em conjunto com o desempenho e a disponibilidade em sistemas NoSQL com consistência baseada em quorum. A metodologia permite analisar diferentes configurações e seus impactos sobre as métricas consideradas, apoiando decisões na definição da configuração do sistema;
- Proposição de um modelo formal baseado em redes de Petri estocásticas para estimar consumo de energia, desempenho e disponibilidade. Validado em ambiente real, o modelo considera operações de leitura e escrita, níveis de consistência, redundância de servidores físicos e das réplicas do SGBD NoSQL baseado em quorum;
- Avaliação do impacto da consistência eventual no consumo de energia, desempenho e disponibilidade, aplicando diferentes configurações no sistema;
- Viabilização da análise da qualidade dos serviços em SGBDs NoSQL com adotam a consistência quorum, considerando consumo de energia, desempenho e disponibilidade.

1.4 ESTRUTURA DO DOCUMENTO

Este documento está organizado em sete capítulos:

- **Capítulo 1 - Introdução:** Contextualiza o objeto de pesquisa, apresentando a motivação, os problemas identificados, os objetivos gerais e específicos, e as contribuições do trabalho;
- **Capítulo 2 – Referencial Teórico:** Aborda os conceitos fundamentais para a compreensão do estudo, incluindo SGBDs NoSQL, consistência eventual, disponibilidade, desempenho, *Benchmark*, redes de Petri e intervalo de confiança;
- **Capítulo 3 – Trabalhos Relacionados:** Descreve estudos relacionados e diferenciações em relação ao presente trabalho;
- **Capítulo 4 – Método Proposto:** Detalha o método de análise, desenvolvimento do modelo e avaliação dos resultados, incluindo as ferramentas utilizadas;
- **Capítulo 5 – Modelo proposto**: Apresenta o modelo desenvolvido com redes de Petri estocásticas;

- **Capítulo 6 – Resultados Experimentais:** Exibe e analisa os resultados experimentais por meio da técnica DoE;

  Capítulo 7 – Conclusão: Resume as principais contribuições do estudo e sugere trabalhos futuros.

2 REFERÊNCIAL TEÓRICO

Este capítulo apresenta os conceitos essenciais para esta pesquisa. Inicialmente, são explorados os fundamentos de SGBDs NoSQL, com ênfase na consistência eventual e no uso de quorum. Em seguida, discute-se a avaliação de dependabilidade, que inclui confiabilidade e disponibilidade, além de desempenho, projeto de experimentos e redes de Petri estocásticas. Por fim, descrevem-se o *benchmark* utilizado na geração de carga e as considerações finais do capítulo.

2.1 SGBD NOSQL

Durante a década de 2000, o termo NoSQL ganhou relevância com a rápida expansão da internet (MEIER et al., 2019). Novos paradigmas e tecnologias de armazenamento surgiram para lidar com grandes volumes de dados estruturados, semiestruturados e não estruturados gerados globalmente. Esses dados são originados de fontes como redes sociais, serviços de armazenamento em nuvem e dispositivos conectados à internet. Para gerenciar esse volume crescente, os sistemas precisam se adaptar, processar consultas de forma eficiente e escalar conforme a demanda. Assim, a prioridade é garantir disponibilidade, desempenho e escalabilidade para atender às demandas do sistema projetado (STOREY; SONG, 2017).

Para alcançar esses objetivos, é essencial utilizar soluções eficientes e flexíveis na gestão de grandes volumes de dados. Os SGBDs relacionais, apesar de priorizarem a integridade e proteção das transações, podem não ser adequados devido à complexidade do controle de integridade e ao alto consumo de processamento. Em cenários de *Big Data*, esses bancos podem atingir seus limites rapidamente (MEIER et al., 2019).

Além disso, os SGBDs relacionais enfrentam desafios relacionados à disponibilidade e ao desempenho, especialmente em transações distribuídas. Isso pode dificultar o atendimento de requisitos de desenvolvimento ágil e iterativo, comuns no contexto de *Big Data* (DAVOUDIAN; CHEN; LIU, 2018). Quando esses requisitos não podem ser plenamente atendidos, torna-se necessário priorizar as demandas. Nesse cenário, os SGBDs NoSQL oferecem uma alternativa viável, pois são capazes de lidar com as três principais características do *Big Data*: volume, velocidade e variedade (BOUAZIZ; NABLI; GARGOURI, 2020).

2.1.1 Teorema CAP

Os bancos de dados NoSQL geralmente possuem uma estrutura distribuída e tolerante a falhas, baseada na redundância de dados. Contudo, essa arquitetura compromete frequentemente a consistência forte. Essa característica é explicada pelo Teorema CAP (BREWER, 2012), desenvolvido para sistemas distribuídos sujeitos a falhas.

O Teorema CAP estabelece que um sistema distribuído não pode garantir simultaneamente as três propriedades seguintes:

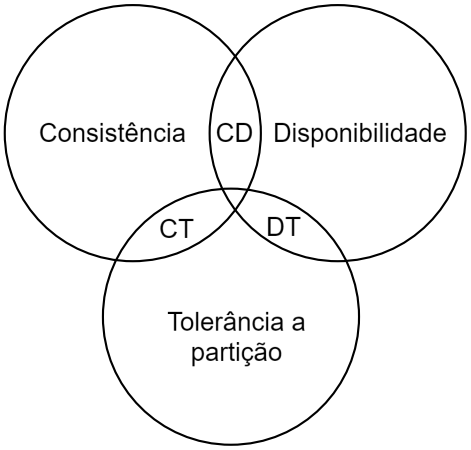
- Consistência (*Consistency*): Todos os dados acessados são sempre os mais atualizados;
- Disponibilidade (*Availability*): Todas as operações de leitura ou gravação recebem uma resposta, mesmo se um nó falhar;
- Tolerância a Partições (*Partition tolerance*): O sistema continua funcionando, mesmo com falhas de rede.

A consistência assegura que todos os dados acessados sejam sempre os mais atualizados. Disponibilidade significa que todas as operações de leitura ou gravação recebem uma resposta bem-sucedida, mesmo em caso de falha de um nó. Já a tolerância a partições garante que o sistema continue operando, mesmo diante de falhas na rede. Com base nessas propriedades, os sistemas distribuídos podem ser caracterizados em três categorias (ver Figura 2):

- Consistente e Tolerante a Falhas (CT): O sistema é capaz de resistir a falhas, mas não garante a disponibilidade;
- Consistente e Disponível (CD): O sistema assegura consistência e disponibilidade, porém não é tolerante a partições ou falhas;
- Disponível e Tolerante a Falhas (DT): Prioriza disponibilidade e tolerância a falhas, mas não garante consistência.

Considerando a tolerância a partições, os projetistas precisam optar entre consistência e disponibilidade. Alguns SGBDs priorizam a consistência, o que pode aumentar o tempo de inatividade. Já priorizar a disponibilidade pode comprometer a consistência, resultando em dados desatualizados em algumas consultas. Nesse contexto, os SGBDs NoSQL geralmente focam

Figura 2 – Teorema CAP



Fonte: Próprio autor

na disponibilidade e no desempenho, equilibrando essas métricas e relegando a consistência a uma preocupação secundária (BREWER, 2012; KUMAR et al., 2018).

Diferentemente dos bancos de dados relacionais, que exigem esquemas tabulares rígidos para a organização dos dados, os sistemas NoSQL apresentam uma estrutura flexível (STOREY; SONG, 2017). Essa flexibilidade torna os SGBDs NoSQL mais adequados para armazenamento em nuvem.

A ausência de um esquema definido (*schemaless*) nos SGBDs NoSQL proporciona maior versatilidade. Outras características que facilitam o gerenciamento de grandes volumes de dados incluem a diversidade arquitetural. Em vez de utilizarem *pools* de armazenamento centralizados, como o Storage Area Network (SAN), esses sistemas empregam armazenamento local, permitindo velocidades de acesso comparáveis às de discos rígidos. A elasticidade é outra característica importante: ao adicionar novos servidores, subconjuntos de dados são replicados para esses servidores adicionais. Contudo, essa replicação não é instantânea, podendo causar indisponibilidade temporária de alguns dados (CHANDRA, 2015).

2.2 CONSISTÊNCIA EVENTUAL

A consistência em bancos de dados garante que os dados permaneçam válidos e íntegros após a execução de operações pelos aplicativos clientes. Bancos de dados relacionais geralmente utilizam transações com as propriedades ACID, que asseguram (CHANDRA, 2015):

- Atomicidade: Todas as operações de uma transação devem ser concluídas com sucesso, caso contrário as mudanças são revertidas;

- Consistência: As transações devem levar o banco de dados de um estado consistente a outro, respeitando regras e restrições;
- Isolamento: Transações simultâneas não devem interferir entre si, sendo executadas como se fossem únicas;
- Durabilidade: Uma vez confirmada, a transação é permanente, garantindo que mudanças sejam mantidas mesmo após falhas do sistema.

Essas características proporcionam forte consistência ao final das transações, assegurando que operações de leitura retornem os dados mais recentes e que gravações sejam replicadas em todos os nós do banco (AHMED et al., 2023).

Nos SGBDs NoSQL, o acrônimo BASE substitui as propriedades ACID, priorizando disponibilidade e tolerância a partições. BASE significa:

- Basicamente Disponível: O sistema está sempre disponível para consultas, mesmo que nem todas as operações de escrita sejam finalizadas;
- Estado flexível: O estado do sistema pode mudar ao longo do tempo, mesmo sem novas requisições, devido à propagação assíncrona das atualizações;
- Consistência Eventual: Apesar de inconsistências temporárias, os dados tornam-se consistentes com o tempo, garantindo que todas as réplicas apresentem eventualmente a versão mais recente.

Essa abordagem prioriza a disponibilidade e desempenho em sistemas distribuídos, embora garantir consistência durante as operações possa ser desafiador (KRECHOWICZ; DENIZIAK; ŁUKAWSKI, 2021).

Em sistemas distribuídos, a consistência pode impactar diretamente o desempenho. Para gravações, é necessário confirmar que a operação foi replicada em todos os servidores requisitados. Em leituras, o SGBD deve assegurar que os dados retornados estejam atualizados. A consistência eventual melhora a disponibilidade ao permitir que nem todas as réplicas sejam atualizadas imediatamente, aumentando a tolerância a falhas. Contudo, em SGBDs baseados em quorum, a disponibilidade pode ser reduzida, já que mais servidores precisam confirmar as operações (ARAÚJO et al., 2024).

2.2.1 Consistência Quorum

Para minimizar requisitos distintos e reduzir o intervalo de inconsistência, SGBDs NoSQL amplamente utilizados, como Riak, Amazon DynamoDB e Cassandra, adotam a abordagem de quorum para gerenciar a consistência entre réplicas (BARON et al., 2016; KALID et al., 2017; WAHID; KASHYAP, 2019). O quorum é o número mínimo de réplicas necessário para validar uma operação antes de considerá-la concluída. O sucesso de operações, como leitura ou gravação, é confirmado somente quando o número suficiente de réplicas reconhece a conclusão bem-sucedida da solicitação (GIFFORD, 1979; YAO; WANG, 2020). O nível de consistência pode ser ajustado explicitamente (por exemplo, 1, 2 ou 3), dependendo das demandas do sistema, mas é limitado pelo fator de replicação do SGBD, ou seja, o número total de réplicas disponíveis (YAO; WANG, 2020).

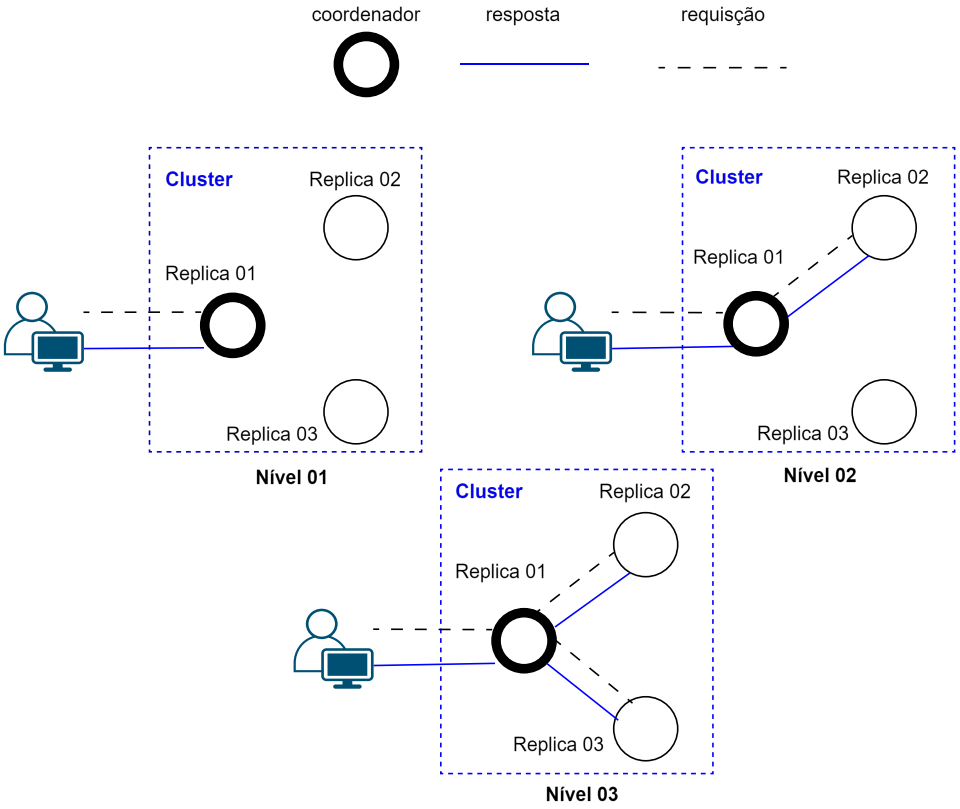
Para coordenar essas operações e assegurar que o quorum necessário seja alcançado, a réplica que recebe uma solicitação de operação pode também ser responsável por coordenar a comunicação com outras réplicas redundantes, dependendo do nível de consistência. Esta réplica é chamada coordenador e qualquer uma das réplicas pode assumir essa função, dependendo do balanceamento de carga do sistema (HUANG et al., 2017).

A Figura 3 ilustra um *cluster* de SGBD com três réplicas e níveis distintos de consistência. No nível 1, uma operação, como uma gravação, exige confirmação de apenas uma réplica. No nível 2, a operação deve ser confirmada por duas réplicas e, no caso de leituras, os dados são comparados para garantir a versão mais recente. Este nível corresponde à política de *QUORUM*, onde mais da metade das réplicas garantem a consistência da escrita (KUMAR et al., 2018). No nível 3, todas as réplicas precisam confirmar a operação, seguindo a política de *ALL*, assegurando a consistência forte e o acesso à versão mais atual dos dados (DIPIETRO, 2020).

O nível de consistência em operações de SGBDs NoSQL é definido por três variáveis principais (YAO; WANG, 2020; AHMED et al., 2023):

- Fator de replicação (RF): Número total de réplicas no cluster;
- Quorum de escrita (W): Número de réplicas para confirmar uma operação de escrita;
- Quorum de leitura (R): Número de réplicas para confirmar uma operação de leitura.

Figura 3 – Exemplo níveis de consistencia (níveis 1, 2 e 3)



Fonte: adaptado de (ARAÚJO et al., 2024)

Em sistemas baseados em quorum, a consistência eventual é alcançada quando $W + R \leq RF$. Já a consistência forte ocorre quando $W + R > RF$. Por exemplo:

- Com $W = 1$, $R = 1$ e $RF = 3$, temos consistência eventual, pois é possível ler dados de uma réplica ainda não atualizada;
- Com $W = 2$, $R = 2$ e $RF = 3$, temos consistência forte, garantindo que a leitura sempre incluirá ao menos uma réplica atualizada.

A consistência eventual permite concluir operações de leitura antes que todas as réplicas sejam atualizadas, oferecendo flexibilidade ao sistema. Essa capacidade de alternar entre consistência eventual e forte é essencial para que os projetistas ajustem o sistema conforme os requisitos específicos (KHELAIFA et al., 2019).

2.3 MODELOS PARA AVALIAÇÃO DE DEPENDABILIDADE E DESEMPENHO

A dependabilidade e o desempenho são aspectos essenciais na avaliação de sistemas distribuídos, especialmente em ambientes de armazenamento em nuvem que utilizam bancos de dados NoSQL. A dependabilidade abrange confiabilidade e disponibilidade, sendo fundamental para garantir o funcionamento do sistema mesmo diante de falhas. O desempenho refere-se à eficiência do sistema na execução de operações, considerando métricas como tempo de resposta e duração do processamento (MACIEL, 2023).

A avaliação conjunta dessas métricas permite identificar o equilíbrio entre a qualidade dos serviços oferecidos e os custos associados ao uso dos recursos computacionais. Para realizar essa avaliação, diversos modelos formais e técnicas experimentais podem ser aplicados, proporcionando resultados precisos e confiáveis. Entre as principais abordagens destacam-se as redes de Petri estocásticas (Stochastic Petri Nets (SPN)), que permitem representar formalmente comportamentos concorrentes e eventos aleatórios, e o Projeto de Experimentos (*Design of Experiments* - DoE), que oferece uma metodologia estruturada para condução dos testes e análise dos resultados obtidos (MURATA, 1989; MONTGOMERY, 2017).

Nesta seção, serão abordados os fundamentos teóricos relacionados à dependabilidade, desempenho, Projeto de Experimentos e redes de Petri estocásticas.

2.3.1 Confiabilidade e Disponibilidade

Os serviços em nuvem são amplamente utilizados e exigem altos níveis de disponibilidade. Garantir essa disponibilidade é um desafio que frequentemente requer o uso de servidores redundantes (DAVOUDIAN; CHEN; LIU, 2018). Falhas nesses serviços podem degradar o desempenho ou causar interrupções, devido à indisponibilidade dos recursos solicitados. Essas interrupções podem ser resultado de falhas de *hardware*, *software*, conexões de rede ou infraestrutura.

Falhas frequentes afetam negativamente a confiança dos usuários no serviço contratado, levando-os a procurar alternativas (LI et al., 2021). Além disso, a recuperação do sistema após falhas pode ser custosa, especialmente quando envolve a sincronização de ambientes.

A confiabilidade representa a probabilidade de o sistema operar sem falhas durante um intervalo de tempo definido (MACIEL, 2023). A Equação (2.1) apresenta essa métrica, onde $R(t)$ é definida como a probabilidade de um sistema realizar sua função predefinida sem falhas para um intervalo de tempo específico. T é a variável aleatória que representa o tempo até

a falha do sistema ou de um único componente. A confiabilidade também pode ser expressa pela Equação (2.2), relacionada à função distribuição cumulativa dada pela Equação (2.3), indicando que uma falha ocorreu antes do tempo T.

$$R(t) = P\{T \geq t\} \tag{2.1}$$

$$R(t) = 1 - F(t) \tag{2.2}$$

$$F(t) = P\{T < t\} \tag{2.3}$$

A disponibilidade é a probabilidade de o sistema estar operacional em um dado momento, considerando a alternância entre estados operacionais e de falha (MACIEL, 2023). As Equações (2.4) e (2.5) são comumente adotadas, onde A representa a disponibilidade, $Uptime$ é o tempo em que o sistema esteve ativo e $Downtime$ é o período de inatividade.

$$A = \frac{Uptime}{Downtime + Uptime} \tag{2.4}$$

$$A = \frac{MTTF}{MTTF + MTTR} \tag{2.5}$$

O tempo médio até ocorrer uma falha é representado por Mean Time to Failure (MTTF), enquanto o tempo médio necessário para reparo é dado por Mean Time to Repair (MTTR). Esses valores são obtidos pelas Equações (2.6) e (2.7), onde $M(t)$ é a função de distribuição cumulativa que expressa a probabilidade de um reparo ocorrer em um determinado tempo t, e $R(t)$ é a função de confiabilidade mencionada anteriormente.

$$MTTF = \int_0^\infty R(t)\, dt \tag{2.6}$$

$$MTTR = \int_0^\infty (1 - M(t))\, dt \tag{2.7}$$

Para melhorar a disponibilidade e a confiabilidade, estratégias como redundância e replicação de componentes são comumente adotadas (MACIEL, 2023). A redundância envolve a adição de componentes para que, em caso de falha, um componente redundante assuma

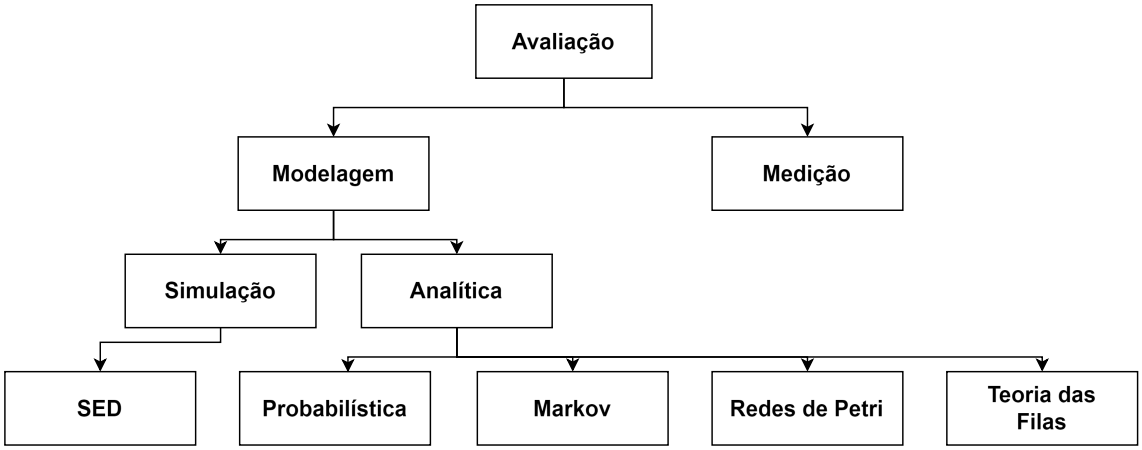
as funções do principal. Por exemplo, a redundância dinâmica utiliza componentes substitutos, enquanto falhas também podem ser gerenciadas por técnicas de codificação, capazes de detectar e corrigir erros em *hardware* e *software*.

2.3.2 Avaliação de Desempenho

A avaliação de desempenho é essencial em todas as etapas do ciclo de vida de um sistema, influenciando seu projeto, aquisição e operação eficiente (CARVALHO; COSTA, 2005; JAIN, 1991). Essa prática ajuda a prevenir custos elevados e otimizar o funcionamento dos sistemas. Para isso, é necessário conhecer claramente o sistema a ser avaliado e selecionar metodologias, cargas de trabalho e ferramentas adequadas (JAIN, 1991; BUKH, 1992).

A definição de métricas apropriadas é fundamental na avaliação de desempenho. Uma abordagem eficiente é listar os serviços oferecidos pelo sistema e considerar os possíveis resultados para cada requisição. Por exemplo, em um banco de dados, o serviço de responder consultas pode resultar em respostas corretas, incorretas ou ausência de resposta. Quando o sistema executa o serviço corretamente, seu desempenho pode ser medido por tempo de resposta, taxa de execução e recursos consumidos (BUKH, 1992; ARAÚJO, 2022).

Figura 4 – Avaliação de desempenho



Fonte: (CALLOU et al., 2011)

Os objetivos da avaliação de desempenho podem incluir comparação de alternativas, análise de impacto, ajuste do sistema e definição de expectativas (LILJA, 2005). Para uma análise de desempenho eficaz, é necessário considerar os custos associados a decisões incorretas. Técnicas como medição e modelagem analítica desempenham papel fundamental nesse contexto. Métodos de análise de desempenho, ilustrados na Figura 4, podem ser aplicados. A combinação

dessas técnicas, ajustada ao tempo e aos recursos disponíveis, proporciona um entendimento mais detalhado do sistema. O uso de ferramentas de modelagem, simulação e medição oferece informações que facilitam a tomada de decisões mais informadas e precisas.

O uso de técnicas de medição permite obter resultados mais precisos, pois monitoram o sistema sob carga de trabalho, aproximando-se do ambiente real. Para resultados representativos, a carga de trabalho, real ou sintética, deve ser cuidadosamente selecionada (ARAUJO, 2009; BUKH, 1992). Embora a carga real represente fielmente o sistema, nem sempre é ideal, pois pode não ser significativa, sofrer interferências ou apresentar dificuldades de acesso aos dados. Por isso, cargas sintéticas são frequentemente usadas em *benchmarks* e programas (ARAÚJO, 2022).

A modelagem analítica constitui uma abordagem versátil para avaliação de desempenho, permitindo representar matematicamente o sistema e calcular métricas de interesse a partir de valores de entrada (RODRIGUES; ENDO; SILVA, 2019). Esses modelos podem capturar características complexas como concorrência e paralelismo. Para garantir que representem adequadamente os sistemas reais, é comum validá-los por meio de comparações estatísticas com medições do sistema em funcionamento.

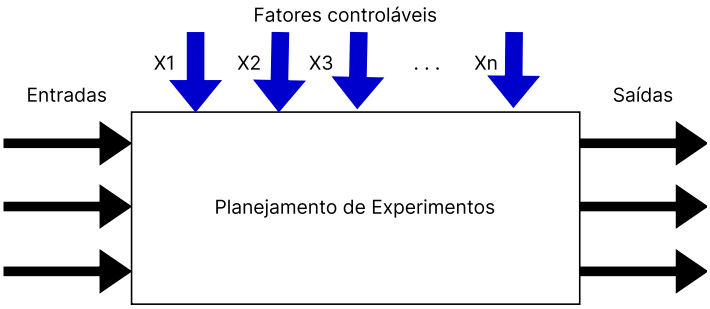
Esta dissertação adota a modelagem analítica fundamentada em medições para criar modelos com alto grau de confiança. A abordagem foi escolhida por sua capacidade de estimar métricas relevantes para SGBDs NoSQL baseados em consistência quorum, como consumo de energia, desempenho e disponibilidade. Além disso, permite usar dados reais para validar e ajustar os modelos, tornando os resultados confiáveis e aplicáveis em projetos reais.

2.3.3 Projeto de Experimentos (Design of Experiments)

O Projeto de Experimentos (*Design of Experiments* – DoE) é importante para compreender sistemas ou processos na pesquisa científica (MONTGOMERY, 2017). Essa metodologia permite identificar quais fatores influenciam significativamente as variáveis de resposta do sistema avaliado e validar os resultados de forma sistemática (MONTGOMERY; RUNGER, 2020).

A etapa inicial consiste na definição do problema, seguida da identificação das variáveis de resposta e dos fatores que podem influenciá-las. Cada fator é avaliado em diferentes níveis, representando variações a serem analisadas. Com base nessas definições, realizam-se experimentos controlados para a coleta de dados. Posteriormente, aplica-se uma análise estatística para identificar os fatores e compreender seus efeitos sobre as variáveis de resposta.

Figura 5 – Modelo básico de um projeto de experimentos



Fonte: adaptado de (MONTGOMERY; RUNGER, 2010)

A Figura 5 representa o processo de experimentação e destaca os fatores controláveis. Durante os experimentos, as variáveis de entrada são modificadas para observar seus efeitos nas variáveis de saída (MONTGOMERY; RUNGER, 2020). Um fator corresponde a uma variável controlável que influencia diretamente os resultados. Os valores específicos atribuídos a esses fatores são denominados níveis. Mudanças nas médias dos resultados devido à alteração dos níveis são denominadas efeitos. Portanto, é importante definir claramente as variáveis de resposta, os fatores e seus níveis antes da realização dos experimentos (MASON; GUNST; HESS, 2003).

As entradas são representadas pelas cargas de trabalho executadas no sistema, enquanto os fatores do ambiente afetam seu desempenho. Cada fator pode apresentar diversos níveis, aumentando a complexidade e a quantidade necessária de experimentos. Por isso, o planejamento fatorial é recomendado para avaliar o impacto combinado dos fatores, em vez de analisá-los isoladamente. Este planejamento pode ser completo, com todos os fatores possuindo dois níveis (2^k), ou fracionado, quando há restrições de recursos disponíveis (MONTGOMERY, 2017).

2.3.4 Redes de Petri

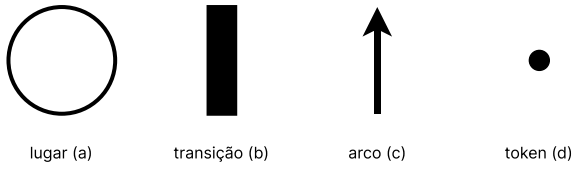
Redes de Petri, ou Petri Nets (PN), são um formalismo matemático e gráfico que permite a modelagem, análise e avaliação de sistemas concorrentes, distribuídos e complexos (MURATA, 1989). A principal vantagem dessa técnica é a capacidade de representar, de forma visual e intuitiva, aspectos como concorrência, sincronização e compartilhamento de recursos. Uma Rede de Petri é definida por um conjunto de lugares, transições e arcos, permitindo visualizar o fluxo de informações ou operações do sistema.

Uma PN é formalmente definida por uma tupla $PN = (P, T, F, W, M_0)$.

- P é o conjunto finito de lugares;
- T é o conjunto finito de transições;
- F representa os arcos, ou a relação de fluxo entre lugares e transições;
- W é a função de atribuição de pesos aos arcos;
- M_0 define a marcação do estado inicial do modelo;

Os elementos fundamentais das Redes de Petri são ilustrados na Figura 6. Os lugares (representados por círculos) indicam estados ou condições do sistema. As transições (barras ou retângulos) simbolizam eventos ou mudanças no sistema. Os arcos direcionados conectam lugares a transições (e vice-versa) indicando o fluxo e a dependência lógica entre estados e eventos. Os *tokens* (fichas ou marcas) são elementos dinâmicos que ocupam os lugares, representando recursos ou o estado atual do sistema. A distribuição desses *tokens* define o estado atual da rede (marcação), que é alterada conforme ocorrem disparos das transições.

Figura 6 – Elementos de uma rede de Petri

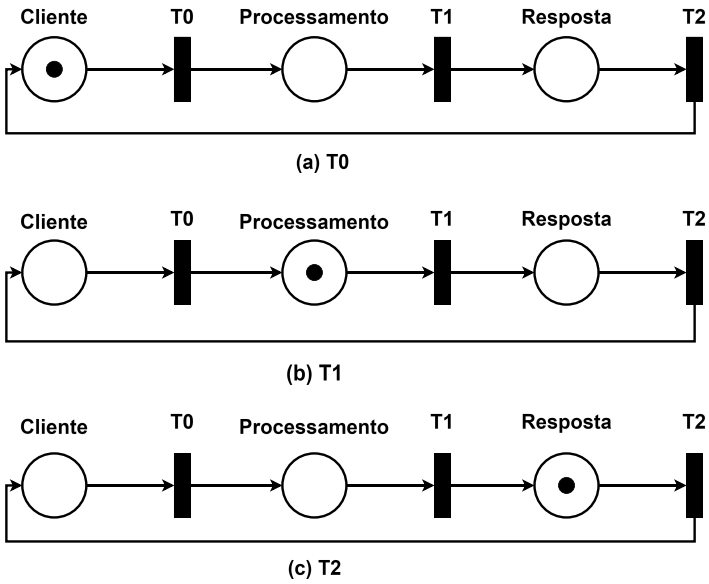


Fonte: adaptado de (MURATA, 1989)

As Redes de Petri possibilitam realizar análises estruturais e comportamentais, permitindo avaliar propriedades como alcançabilidade (*reachability*), limitação (*boundedness*), vivacidade (*liveness*), cobertura (*coverability*), reversibilidade e *home state* (MURATA, 1989).

Um exemplo de PN é apresentado na Figura 7, ilustrando um sistema cliente-servidor. Nesse modelo, o cliente envia uma requisição ao servidor, o servidor processa a requisição e responde ao cliente. Inicialmente, o *token* está no lugar Cliente, permitindo que o cliente faça uma requisição. As transições T0, T1 e T2 representam a requisição, o processamento e o envio da resposta, respectivamente. Na Figura 7 (a), apenas a transição T0 está habilitada, pois o *token* está no lugar Cliente e um arco conecta esse lugar à transição T0. Quando T0 é disparada, o *token* move-se de Cliente para Processamento, desabilitando T0 e habilitando

Figura 7 – Estados de um modelo que representa um sistema



T1 (Figura 7 (b)). Ao disparar T1, o *token* vai para o lugar Resposta, ativando apenas T2 (Figura 7 (c)). Finalmente, ao disparar T2, o *token* retorna ao lugar Cliente.

Nesta dissertação, escolheu-se a análise estacionária que permite explorar todas as possibilidades do modelo para obter os resultados. Quando não é possível avaliar todas as possibilidades do modelo, utiliza-se a análise transiente, onde a avaliação depende do tempo. A análise estacionária foi realizada porque o modelo proposto apresenta propriedades comportamentais e estruturais adequadas para essa abordagem (MARUSSY et al., 2016; MACIEL, 2023).

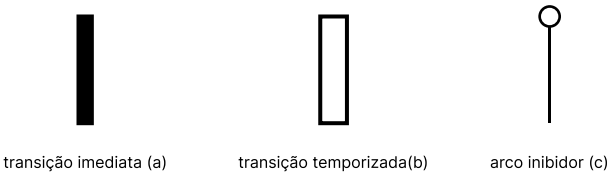
2.3.4.1 Redes de Petri Estocásticas

As Redes de Petri Estocásticas, ou SPN, são uma extensão das PN voltada para especificar sistemas e realizar análises probabilísticas. As SPNs associam tempos às transições do modelo, seguindo uma distribuição exponencial. As transições podem representar o tempo necessário para que uma máquina execute uma tarefa ou serem imediatas, indicando que não há tempo associado à ação da transição (MURATA, 1989). Isso permite avaliar o desempenho e a confiabilidade dos sistemas, considerando o tempo de disparo das transições como variáveis aleatórias (MACIEL, 2023). Assim, as SPNs são adequadas para modelar sistemas em que o tempo é um fator relevante, proporcionando uma análise mais completa.

Além disso, as SPNs podem ser mapeadas para uma Cadeia de Markov de Tempo Contínuo (CTMC), o que possibilita a análise de diversas métricas de desempenho. Esse mapeamento converte o espaço de estados da SPN em uma CTMC por meio do grafo de alcançabilidade.

Por exemplo, uma SPN com estados de processamento e espera pode ser convertida em uma CTMC, na qual os estados e as taxas de transição refletem tempos de atraso com distribuição exponencial. Esse processo permite a estimativa de métricas como tempo de resposta e *throughput* (ARAUJO, 2009; MACIEL, 2023). Com uma base matemática sólida, as SPNs também possibilitam a representação e análise de sistemas paralelos e heterogêneos que envolvem simultaneidade e sincronização (MACIEL, 2023).

Figura 8 – Elementos de uma rede de Petri estocástica



Fonte: Próprio autor

As SPNs possuem dois tipos de transições: imediatas e temporizadas. As transições imediatas (Figura 8 (a)), representadas por retângulos pretos finos, têm atraso igual a zero. Por outro lado, as transições temporizadas (Figura 8 (b)) são representadas por retângulos brancos e apresentam um atraso distribuído exponencialmente. Além dos arcos tradicionais, as SPNs utilizam arcos inibidores (Figura 8 (c)). Esses arcos, que conectam lugares e transições como nas PN tradicionais, são identificados por um círculo branco em uma das extremidades. Uma transição é desativada se houver *tokens* no lugar de origem do arco inibidor e ativada quando não houver *tokens*. Assim como os arcos tradicionais, os arcos inibidores também podem ter um peso associado.

Uma SPN é definida pela 9-tupla $SPN = (P, T, I, O, H, \Pi, G, M_0, ATTS)$ (GERMAN, 2000), em que:

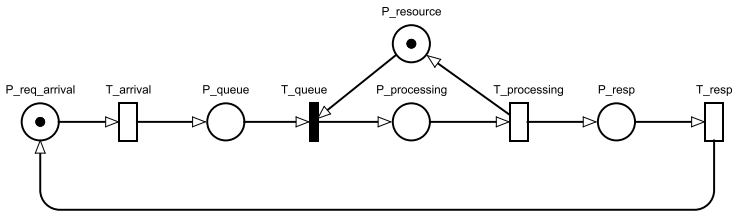
- $P = \{p1, p2, ..., p_n\}$ é o conjunto finito de lugares;
- $T = \{t1, t2, ..., t_n\}$ é o conjunto de transições, que podem ser imediatas, determinísticas ou exponenciais;
- $I \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ é a matriz que descreve os arcos de entrada;
- $O \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ denota a matriz que descreve os arcos de saída;
- $H \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ representa a matriz correspondente aos arcos inibidores;

- $\Pi \in \mathbb{N}^m$ é um vetor que atribui níveis de prioridade a cada transição;
- $G \in (\mathbb{N}^n \rightarrow \{true, false\})^m$ é o vetor que contém a função de guarda associada à marcação dos lugares para cada transição;
- $M_0 \in \mathbb{N}^m$ é o vetor que indica o estado inicial de cada lugar;
- $ATTs = (Dist, W, Markdep, Policy, Concurrency)^m$ compreende o conjunto de atributos das transições, onde:
  - $Dist \in \mathbb{N}^m \rightarrow \mathcal{F}$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, onde o domínio de $\mathcal{F}$ é $[0, \infty)$ (a distribuição pode depender da marcação);
  - $W \in \mathbb{N}^m \rightarrow \mathbb{R}^+$ é a função de peso, que atribui um peso (w_t) às transições imediatas e uma taxa λ_t às transições temporizadas, onde $W(t) = w_t \geq 0$ se *t* é uma transição imediata ou $W(t) = \lambda_t > 0$ caso contrário;
  - $Markdep \in \{constant, enabledep\}$ determina se a distribuição de probabilidade associada ao tempo de uma transição é constante (*constant*) ou depende da marcação (*enabledep*);
  - $Policy \in \{prd, prs\}$ define a política de preempção adotada pela transição, onde *prd (preemptive repeat different)* corresponde ao resampling e *prs (preemptive resume)* é equivalente à *age memory policy*;
  - $Concurrency \in \{SS, IS\}$ representa o grau de concorrência das transições, onde Infinite Server (IS) indica a semântica de servidor infinito e Single Server (SS) representa a semântica de servidor único, similar aos modelos de fila.

As transições temporizadas admitem distintos graus de concorrência, como semântica de servidor único (SS) ou semântica de servidor infinito (IS). Em geral, a semântica IS denota disparos de transição paralelo, nas quais a taxa de disparo de uma transição aumenta linearmente com base em seu grau de habilitação. Para a semântica SS, uma taxa de disparo permanece constante (MARSAN et al., 1998; MACIEL, 2023).

Na Figura 9, encontra-se um exemplo de SPN que modela um sistema de gerenciamento de banco de dados. O lugar $P\_req\_arrival$ representa a chegada de requisições ao sistema, o *token* é enviado para P_queue ao habilitar a transição temporizada $T\_arrival$, que simboliza a fila de requisições. Em P_queue, o sistema aguarda até que o recurso esteja disponível para

Figura 9 – Exemplo de SPN



Fonte: Próprio autor

iniciar a operação, representado pelo *token* em $P\_resource$. Esse *token* habilita a transição imediata T_queue, que permite iniciar o processamento da operação em $P\_processing$, com o tempo determinado pela transição temporizada $T\_processing$. Ao final do processamento, o *token* é transferido para P_resp, que indica o envio de resposta e libera o recurso em $P\_resource$. A transição imediata T_resp então envia a resposta ao usuário, retornando o *token* para $P\_req\_arrival$.

2.4 BENCHMARK

Para avaliar sistemas, utiliza-se a técnica de medição, que gera uma carga de trabalho para ser processada pelo sistema (BUKH, 1992). A seleção da carga de trabalho deve ser criteriosa, pois uma escolha inadequada pode gerar dados que não refletem o comportamento típico do sistema ou não avaliem corretamente a métrica desejada. Ferramentas de *Benchmark* são frequentemente usadas para realizar essas análises, executando um conjunto de operações e monitorando o comportamento do sistema (BUKH, 1992). O *benchmarking* é amplamente utilizado para gerar cargas de trabalho sintéticas e avaliar sistemas computacionais em diversos níveis de abstração (COOPER et al., 2010).

O aumento no número de sistemas e a ausência de padrões claros de comparação de desempenho tornam mais complexa a compreensão das relações entre sistemas e suas respectivas cargas de trabalho. Contudo, o uso de *Benchmark* facilita uma comparação eficiente entre diferentes produtos (BORAL; DEWITT, 1984). Para ser eficiente, um *Benchmark* deve ser relevante para a aplicação que representa, compatível com diferentes arquiteturas e escalável para execução em múltiplos sistemas computacionais (CIFERRI, 1995).

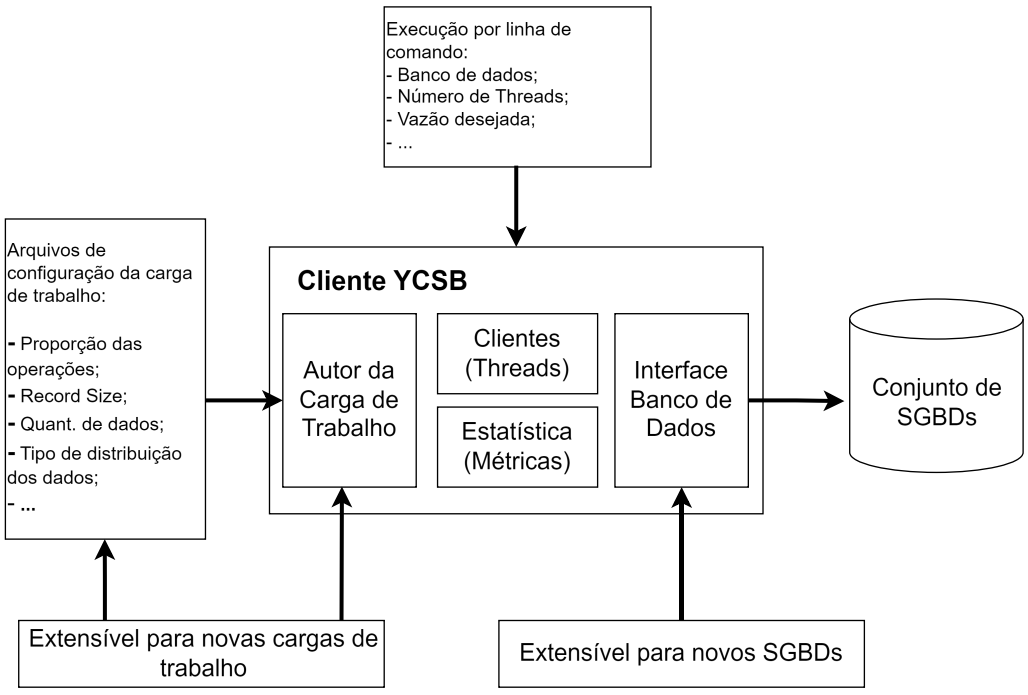
Nesta pesquisa, foi utilizada a ferramenta Yahoo! Cloud Serving Benchmark (YCSB) (COOPER et al., 2010). Essa ferramenta foi empregada para gerar cargas de trabalho representando operações de leitura e escrita no sistema, validando o modelo proposto.

2.4.1 Yahoo! Cloud Serving Benchmark (YCSB)

O *Benchmark* YCSB é uma ferramenta de código aberto que suporta diversos bancos de dados. Sua principal característica é a extensibilidade, permitindo a criação de cargas de trabalho por meio de arquivos de configuração e a avaliação de diferentes SGBDs por meio de novas classes e interfaces existentes. A ferramenta facilita a comparação de desempenho entre sistemas, criando um padrão de referência para avaliar diferentes sistemas em nuvem (COOPER et al., 2010). O YCSB possui duas fases: *load* e *transaction*. Na fase *load*, os dados são inseridos no SGBD, com a quantidade definida previamente. Na fase *transaction*, executada com o parâmetro *run*, realizam-se operações de leitura e escrita.

O comando *load* carrega dados iniciais no banco de dados, criando, por exemplo, 1.000 registros no Cassandra com a instrução $-p\ recordcount = 1000$. Já o comando *run* executa a carga de trabalho configurada, realizando 1.000 operações transacionais conforme definido pela instrução $-p\ operationcount = 1000$. A proporção das operações é configurada no arquivo de carga de trabalho *workloada*. Em ambas as fases, é possível definir a carga de trabalho, a vazão e o número de usuários simultâneos. Após cada fase, o YCSB computa métricas como tempo de execução, vazão média, tempo de resposta (latência) das requisições e a quantidade de operações bem-sucedidas e falhas.

Figura 10 – Arquitetura do Yahoo! Cloud Serving Benchmark



Fonte: adaptado de (COOPER et al., 2010)

Conforme mostrado na Figura 10, a estrutura do YCSB é organizada em quatro camadas: autor da carga de trabalho, clientes (*threads*), estatísticas e interface do banco de dados. A camada de autor da carga de trabalho é configurada por arquivos que especificam a quantidade de dados, o tamanho dos registros e outros parâmetros necessários para as operações. Esta camada é extensível, permitindo a criação de novos arquivos de configuração conforme necessário.

A camada de clientes define, através da linha de comando, atributos como o número de *threads*, a vazão e os parâmetros de conexão do banco de dados. A camada de estatísticas calcula métricas de desempenho, como tempos de execução e resposta. A interface do banco de dados é composta por classes e métodos que facilitam a comunicação com diferentes bancos de dados, sendo também extensível para integrar novos bancos de dados utilizando os métodos existentes (COOPER et al., 2010).

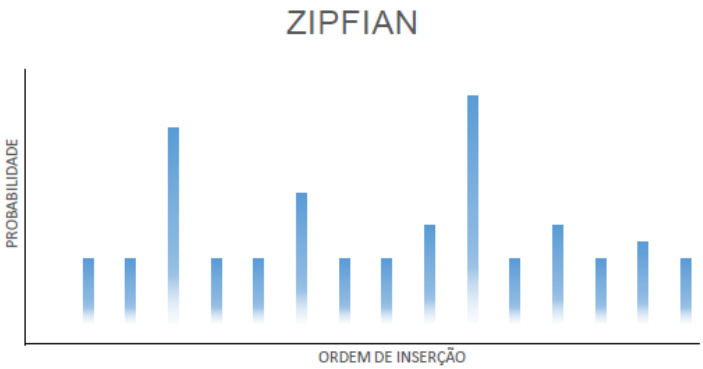
A métrica desempenho foi utilizada para avaliar o consumo de energia do SGBD. O nível de desempenho do *Benchmark* está relacionado à latência das solicitações sob carga. Para validar o modelo proposto, esta pesquisa avalia o consumo de energia elétrica com base no tempo de execução da carga de trabalho, priorizando o consumo de energia em vez da latência isoladamente.

2.4.2 Distribuições do YCSB

O gerador de carga de trabalho realiza escolhas aleatórias para criar dados de Benchmark, sejam de escrita ou leitura. Essas escolhas seguem distribuições probabilísticas, introduzindo aleatoriedade nos testes. O YCSB oferece diversas distribuições para modelar os padrões de acesso aos dados durante os testes. Neste trabalho, foi utilizada a distribuição *Zipfian*, adaptada pelo Yahoo!, que distribui itens populares ao longo de todo o *keyspace* do banco de dados (COOPER et al., 2010).

A Figura 11 ilustra a distribuição *Zipfian*, com os itens representados no eixo horizontal e a probabilidade de seleção no eixo vertical. Essa distribuição simula picos de acesso, onde alguns itens são acessados com maior frequência. No YCSB, isso reflete cenários em que uma pequena parcela dos dados é frequentemente acessada, comportamento observado em muitos aplicativos reais.

Figura 11 – Distribuição Zipfian



Fonte: adaptado de (COOPER et al., 2010)

2.5 CONSUMO DE ENERGIA

O consumo de energia não deve ser negligenciado, pois pode influenciar diretamente os custos operacionais e o desempenho de sistemas distribuídos (MALMODIN, 2023). Em projetos que adotam SGBDs NoSQL, os dados comumente são replicados em múltiplos servidores para melhorar o desempenho e a disponibilidade. Entretanto, essa replicação gera maior demanda por comunicação entre os servidores, que pode aumentar o consumo de energia devido à maior utilização de recursos do sistema (SHAH; KOTHARI; PATEL, 2022).

Para estimar o consumo de energia nesses sistemas, mede-se geralmente a potência (em watts ou joules por segundo) utilizada pelos recursos computacionais durante um período específico. Considerando que 1 *watt* (W) equivale a 1 *Joule* por segundo (J/s), a energia total consumida pode ser estimada por meio da Equação 2.8, que representa a energia consumida ao longo do tempo (NILSSON; RIEDEL, 2015).

$$E = \sum_{i}^{T} P_i \times \Delta t \tag{2.8}$$

em que:

- E é o consumo total de energia em joules (J);
- P_i é a potência média no intervalo i em watts (W) ou joules por segundo (J/s);
- Δt é o intervalo de tempo considerado em segundos (s).

2.6 CONSIDERAÇÕES

Este capítulo apresentou os conceitos fundamentais para a compreensão deste trabalho. Inicialmente, foram abordados os conceitos de SGBD NoSQL, consistência eventual e consistência baseada em quorum. Em seguida, discutiram-se as definições de confiabilidade e disponibilidade, além dos princípios de avaliação de desempenho. Também foram abordadas as redes de Petri e sua extensão, redes de Petri estocásticas a técnica de DoE. Por fim, discutiu-se o uso da ferramenta de *benchmark* utilizado na pesquisa e introduziu-se a análise do consumo de energia como métrica para avaliação do sistema considerado. Esses conceitos fornecem a base para o desenvolvimento do modelo proposto e para a condução dos experimentos descritos nos capítulos seguintes.

3 TRABALHOS RELACIONADOS

Esta seção apresenta os trabalhos relacionados à presente dissertação. O objetivo é avaliar métodos e resultados de estudos sobre modelagem, desempenho, disponibilidade e consumo de energia em sistemas de armazenamento. Foram considerados artigos publicados entre 2016 e 2024, selecionados a partir de bases científicas como *IEEE*, *ACM*, *Elsevier* e *Springer*. Além disso, foram empregadas palavras-chave como "banco de dados NoSQL", "consumo de energia", "avaliação de desempenho", "rede de Petri estocástica" e "consistência quorum".

Após a pesquisa sistemática e a aplicação dos critérios de inclusão, foram selecionados estudos que investigam o consumo de energia, a consistência eventual e o desempenho de sistemas NoSQL (NOUDOUST; ADABI; REZAEE, 2022; GOMES et al., 2023). Algumas abordagens propuseram modelos estocásticos para estimar o impacto da consistência eventual na disponibilidade e no desempenho (RODRIGUES et al., 2019; DIPIETRO, 2020). A avaliação conjunta do impacto da consistência eventual no consumo de energia, desempenho e disponibilidade ainda é pouco explorada. A seguir, os trabalhos relacionados são apresentados de acordo com o tipo de avaliação realizada.

3.1 MODELAGEM, AVALIAÇÃO DE DESEMPENHO E DISPONIBILIDADE

Avaliar o desempenho e a disponibilidade é uma tarefa essencial, e diversos estudos têm sido realizados com o objetivo de analisar essas métricas em sistemas computacionais.

Os autores (GAUR; JOSHI; SRIVASTAVA, 2017) apresentam um modelo baseado em Colored Petri Nets (CPN) para avaliar o desempenho de um servidor de banco de dados. O modelo considera o impacto do número de usuários simultâneos e do tipo de consulta. Os resultados experimentais indicam que o modelo é eficaz para determinar a quantidade ideal de servidores, além de identificar o número máximo de requisições que o banco de dados suporta. Embora relevante, o trabalho avalia apenas o desempenho, sem abordar a disponibilidade do sistema.

Em (RODRIGUES et al., 2019), os autores avaliam o desempenho e a disponibilidade em ambientes de nuvem privada que utilizam SGBDs NoSQL. A abordagem propõe um modelo baseado em Generalized Stochastic Petri Nets (GSPN), considerando diferentes cargas de trabalho, mecanismos de redundância e o *Pool Size*, que representa threads executando requisições simultâneas. Os resultados mostram que o *Pool Size* impacta o desempenho, com

variações superiores a 300%. O estudo também destaca que técnicas de redundância podem gerar tempos de inatividade variando entre 45 e 229 horas anuais. Apesar de avaliar desempenho e disponibilidade, a consistência não foi analisada.

Os autores (CHATTARAJ; SARMA; SAMANTA, 2019) propõem um modelo baseado emSPN para representar o Hadoop Distributed File System (HDFS), com foco na análise de confiabilidade e disponibilidade. Os resultados indicam que a quantidade de blocos de dados e o número de nós no HDFS influenciam significativamente a disponibilidade. Além disso, a replicação dos blocos aumenta a tolerância a falhas, e o fator de replicação padrão de 3 foi considerado ideal. No entanto, o desempenho do sistema não foi avaliado nesse estudo.

Em (ARAUJO et al., 2021) avaliam o desempenho dos SGBDs NoSQL Cassandra e MongoDB em uma infraestrutura de nuvem com *clusters* de três servidores cada. Para reduzir as diferenças de arquitetura, o Cassandra foi configurado sem replicação de dados. No primeiro experimento, realizado com três cargas de trabalho distintas usando o YCSB, o Cassandra apresentou melhor desempenho. No segundo experimento, que simulou operações em um banco de dados de investidores do Tesouro Nacional, o MongoDB teve desempenho superior, exceto nas operações de atualização. As avaliações de disponibilidade e consistência dos dados não foram consideradas.

Os autores (NOUDOUST; ADABI; REZAEE, 2022) avaliam o desempenho e a disponibilidade do Cassandra NoSQL com diferentes níveis de consistência baseados em quorum. Foram realizadas operações de leitura e escrita em conjuntos de dados de tamanhos variados. Os resultados mostram que o nível de consistência 1 resultou em cinco vezes mais operações de leitura desatualizadas do que o nível 2. Níveis mais altos de consistência reduziram significativamente a disponibilidade, enquanto níveis menores apresentaram o maior número de operações por segundo na avaliação de desempenho. Apesar da relevância do estudo, a análise focou apenas em medições, sem utilização de técnicas de modelagem.

(BANSAL; SACHDEVA; AWASTHI, 2023) avaliam os bancos de dados NoSQL MongoDB, Cassandra e Redis. O estudo analisa as arquiteturas de esquema desses SGBDs, utilizando um Modelo de Entidade Relacionamento (MER) para projetar possíveis escolhas de esquema para cada modelo de dados. As arquiteturas são testadas quanto ao desempenho das consultas, considerando tempo de resposta e tamanho da base de dados. Os resultados mostram que, para esquemas desnormalizados, o MongoDB teve um bom desempenho no tempo de resposta das consultas. Embora o trabalho forneça *insights* valiosos, a disponibilidade não é avaliada.

3.2 MODELAGEM, AVALIAÇÃO DE CONSISTÊNCIA E CONSUMO DE ENERGIA

Nos últimos anos, diversos estudos propuseram abordagens para lidar com a consistência em sistemas de gerenciamento de bancos de dados NoSQL. Pesquisadores têm analisado configurações que minimizam o consumo de energia, considerando o impacto dessa escolha no desempenho e na disponibilidade. Contudo, um dos maiores desafios é compreender como os diferentes níveis de consistência influenciam essas métricas de maneira integrada. Até o momento, não foram identificados estudos que abordem a modelagem avaliando essas variáveis simultaneamente.

A pesquisa conduzida por (BURDAKOV et al., 2016) analisa o impacto do número de réplicas e servidores na confirmação de operações de leitura e gravação em ambientes de replicação NoSQL. Um modelo matemático é proposto para avaliar a consistência eventual e estimar a probabilidade de leituras desatualizadas. Além disso, é apresentado um modelo para consistência forte, baseado na sincronização completa do registro. A pesquisa conclui que modelos matemáticos são eficazes para avaliar a consistência em bancos de dados replicados, embora não contemplem a disponibilidade do sistema.

(HUANG et al., 2017) adotaram a modelagem em CPN, permitindo a análise e ajuste do SGBD NoSQL que utiliza a técnica de quorum. Nos experimentos e validação do modelo, o SGBD Cassandra foi implementado em uma infraestrutura de nuvem. Os autores concentraram-se nas métricas de desempenho e no tempo necessário para a sincronização entre réplicas. Os resultados experimentais indicaram a eficiência do modelo na adequação ao sistema. No entanto, embora o estudo avalie consistência e desempenho, não aborda a disponibilidade e o consumo de energia do sistema.

Os autores (GORBENKO; ROMANOVSKY; TARASYUK, 2020) investigaram o impacto das configurações de consistência baseadas em quorum no desempenho do SGBD NoSQL Cassandra. Utilizando o YCSB, eles simularam cargas de trabalho em um ambiente com três servidores replicados no Amazon EC2. Os testes incluíram diferentes proporções de operações de leitura e escrita, demonstrando que as configurações de consistência influenciam significativamente o tempo de resposta do Cassandra. A consistência forte reduziu o desempenho em até 26% quando comparada a outros níveis. A configuração ideal para consistência forte depende da proporção de operações de leitura e escrita, além da carga de trabalho específica. Apesar de explorar os níveis de consistência em relação ao desempenho, o estudo não analisa a disponibilidade do sistema.

Os autores (YAO; WANG, 2020) analisaram o modelo Probabilistic Consistency Guarantee (PCG) em um SGBD NoSQL para estimar o quorum necessário para alcançar a consistência após a gravação mais recente, com menor latência. Caso uma réplica esteja indisponível, o modelo redimensiona o quorum e lê os valores das réplicas disponíveis, reduzindo a desatualização nas operações de leitura. Os resultados indicaram que é possível diminuir a latência em até 48,9% e obter previsões 77,7% mais precisas em comparação com o modelo Probabilistically Bounded Staleness (PBS).

Em (DIPIETRO, 2020), os autores apresentaram um modelo de Queueing Networks (QN) para avaliar o desempenho de SGBDs NoSQL em sistemas de nuvem privada e pública. O estudo focou na consistência e replicação dos dados no Cassandra, avaliando operações de leitura e o impacto no tempo de resposta, *throughput*, fatores de replicação e níveis de consistência. Os resultados mostraram que o modelo estimou o *throughput* e o tempo de resposta com erro abaixo de 10%. Contudo, o estudo não abordou o consumo de energia.

(GAO et al., 2021) conduziram um estudo que implementou alterações na replicação do Cassandra NoSQL, incluindo abordagens de consistência eventual, quorum e forte, baseadas em um algoritmo bizantino de tolerância a falhas. Essas mudanças foram avaliadas quanto ao impacto na latência das operações. O *Benchmark* YCSB foi utilizado para simular cargas de trabalho em uma infraestrutura de nuvem. Os resultados indicaram que o nível de consistência eventual padrão do Cassandra apresentou a menor latência, enquanto customizações de consistência resultaram em desempenho inferior. A disponibilidade do sistema não foi avaliada.

Em (EVANGELISTA, 2021) comparou estudos que abordam modelagens para avaliar o desempenho e a disponibilidade de SGBDs NoSQL. A pesquisa destacou o desafio de encontrar configurações ideais para resolver problemas como baixo desempenho, alto custo e baixa qualidade. Vários modelos foram desenvolvidos para estimar os efeitos de alterações nos parâmetros de tempo de execução. O estudo abordou o atraso de processamento, o nível de consistência e o fator de replicação, destacando que o QN apresenta limitações em relação ao fator de replicação. Consumo de energia e consistência baseada em quorum não foram considerados

Em (SHAH; KOTHARI; PATEL, 2022) investigaram o consumo de energia em bancos de dados NoSQL, comparando diferentes bases de dados, tipos de operações, métricas, ferramentas de medição e *Benchmark*. A pesquisa destacou que os níveis de consistência e a latência das operações influenciam significativamente o consumo energético. Os resultados mostraram que a consistência eventual consome mais energia em cargas de trabalho de leitura intensiva, enquanto a consistência forte apresenta maior consumo em operações de escrita. A escolha do

SGBD e a otimização das operações são fatores que podem melhorar a eficiência energética.

O estudo conduzido por (KENITAR; ARIOUA; YAHYAOUI, 2023) compara SGBDs relacionais e NoSQL, analisando desempenho, latência e eficiência energética. A pesquisa utilizou Aurora e DynamoDB na infraestrutura da AWS, com o Apache JMeter para simular cargas de trabalho e medir a comunicação entre nós dos dispositivos. Os resultados indicaram que SGBDs NoSQL possuem menor latência e consumo de energia em comparação com os bancos de dados relacionais. Para processar 1.024 instâncias de dados, os SGBDs NoSQL consumiram $27,45J/s$, enquanto os relacionais consumiram $33,96J/s$. Apesar de avaliar o consumo de energia, o estudo não abordou a disponibilidade do sistema.

Os autores (GOMES et al., 2023) avaliaram sistemas de armazenamento em nuvem baseados em SGBDs NoSQL e propuseram dois modelos SPN. O primeiro estimou o desempenho com base no tempo de leitura e escrita, considerando o tempo médio entre falhas (MTTF) e o tempo médio de reparo (MTTR). O segundo modelo calculou a probabilidade de acesso ao dado mais recente, assumindo que o dado estava atualizado em todas as réplicas. Resultados mostraram que aumentar a consistência do nível 1 para o nível 3 elevou em 21% a probabilidade de acesso a dados atualizados. Além disso, foi proposto um modelo matemático para estimar o consumo de energia, no entanto, não foi validado e não considera consumo de energia em conjunto com disponibilidade e desempenho.

Em (GUO et al., 2024) propuseram um modelo matemático para estimar o consumo de energia em operações de banco de dados distribuídos, como Cassandra e Redis. O modelo analisou o perfil de consumo energético de consultas individuais e do sistema como um todo, visando otimizar o projeto e a eficiência energética. Resultados indicaram uma precisão média de 94,63% para consultas distribuídas.

Em (FALCãO et al., 2024) avaliaram desempenho e consumo de energia de SGBDs NoSQL multimodelo, comparando ArangoDB, OrientDB, MongoDB e Redis. Utilizando o *Benchmark* YCSB com cargas de 1.000, 5.000 e 10.000 operações, os resultados mostraram que o OrientDB apresentou o maior consumo de energia e tempo de execução, enquanto o Redis teve o menor. O estudo não considerou a consistência, a disponibilidade ou a modelagem dos sistemas.

Tabela 1 – Comparação entre este trabalho e trabalhos relacionados

	NoSQL	Modelo	Desempenho	Disponibilidade	Consistência	Consumo de Energia
(BURDAKOV et al., 2016)	✓	✓	-	-	✓	-
(GAUR; JOSHI; SRIVASTAVA, 2017)	-	✓	✓	-	-	-
(HUANG et al., 2017)	✓	✓	✓	-	✓	-
(RODRIGUES et al., 2019)	✓	✓	✓	✓	-	-
(CHATTARAJ; SARMA; SAMANTA, 2019)	-	✓	-	✓	-	-
(GORBENKO; ROMANOVSKY; TARASYUK, 2020)	✓	-	✓	-	✓	-
(YAO; WANG, 2020)	✓	✓	✓	-	✓	-
(DIPIETRO, 2020)	✓	✓	✓	-	✓	-
(GAO et al., 2021)	✓	✓	-	-	✓	-
(ARAUJO et al., 2021)	✓	-	✓	-	-	-
(EVANGELISTA, 2021)	✓	✓	✓	✓	-	-
(NOUDOUST; ADABI; REZAEE, 2022)	✓	-	✓	-	✓	-
(SHAH; KOTHARI; PATEL, 2022)	✓	-	✓	-	✓	✓
(BANSAL; SACHDEVA; AWASTHI, 2023)	✓	✓	✓	-	-	-
(KENITAR; ARIOUA; YAHYAOUI, 2023)	✓	-	✓	-	-	✓
(GOMES et al., 2023)	✓	✓	✓	✓	✓	✓
(GUO et al., 2024)	✓	✓	✓	-	-	✓
(FALCãO et al., 2024)	✓	-	✓	-	-	✓
Este trabalho	✓	✓	✓	✓	✓	✓

3.3 COMPARAÇÃO DOS TRABALHOS RELACIONADOS

A Tabela 1 apresenta as contribuições de estudos relacionados a esta dissertação, comparando-os em termos de desempenho, disponibilidade, modelagem, consistência e consumo de energia. Embora esses trabalhos abordem temas relevantes, eles se concentram em aspectos específicos e não abrangem todos os pontos de forma integrada.

Por exemplo, (GAUR; JOSHI; SRIVASTAVA, 2017) propõem um modelo para avaliar o desempenho de SGBDs, mas não consideram a disponibilidade e sistemas NoSQL. Os autores (CHATTARAJ; SARMA; SAMANTA, 2019) avaliam a disponibilidade em um ambiente de *big data* usando SPN, mas não abordam o desempenho. Em (RODRIGUES et al., 2019), são apresentados modelos SPN para avaliar disponibilidade e desempenho, mas sem considerar o consumo de energia.

(GOMES et al., 2023) avaliam o desempenho, a disponibilidade e o consumo de energia. No entanto, o modelo matemático proposto para estimar o consumo de energia não foi validado. Por sua vez, (SHAH; KOTHARI; PATEL, 2022) analisam o consumo de energia e a consistência, mas não se concentram na modelagem. Em (FALCãO et al., 2024), são estudados o consumo de energia e o desempenho de SGBDs NoSQL, mas a disponibilidade não é investigada, tampouco são propostos modelos para o sistema.

A abordagem proposta nesta dissertação baseia-se no estudo de Gomes et al. (GOMES et al., 2023), que também utiliza a técnica de quorum. No entanto, diferencia-se por realizar uma

avaliação conjunta do consumo de energia, desempenho e disponibilidade durante operações de leitura e escrita. Enquanto os autores (GOMES et al., 2023) não validam o consumo de energia com medições reais, a presente dissertação utiliza valores coletados em um sistema real por meio do *framework Measurement Server* (ver Subseção 4.2.4.2). Neste trabalho, o desempenho é medido especificamente pela latência, e a disponibilidade é calculada por meio do tempo médio entre falhas (MTTF) e do tempo médio de reparo (MTTR).

3.4 CONSIDERAÇÕES

Este capítulo revisou os principais estudos relacionados à pesquisa desta dissertação. Além de discutir a modelagem de sistemas de armazenamento, foram apresentados estudos sobre avaliação de desempenho e disponibilidade. Também foram analisadas pesquisas focadas na consistência em sistemas NoSQL e no consumo de energia, destacando a comparação de desempenho e a quantificação da consistência.

4 MÉTODO PROPOSTO

Este capítulo descreve o método proposto para construção e validação do modelo para estimar o consumo de energia, desempenho e disponibilidade em SGBDs NoSQL qcom consistência baseada em quorum. Primeiramente, oferece uma visão geral do método, abordando as principais atividades para alcançar os objetivos da pesquisa. Em seguida, detalha cada atividade, iniciando pela definição do problema, as ferramentas utilizadas e o processo de modelagem. Por fim, analisa os resultados dos experimentos.

4.1 VISÃO GERAL

O método proposto é acessível a usuários com conhecimento em modelagem de sistemas e SPN. Ele permite avaliar, de forma integrada, o consumo de energia, o desempenho e a disponibilidade, auxiliando na escolha de configurações que atendam aos requisitos do sistema.

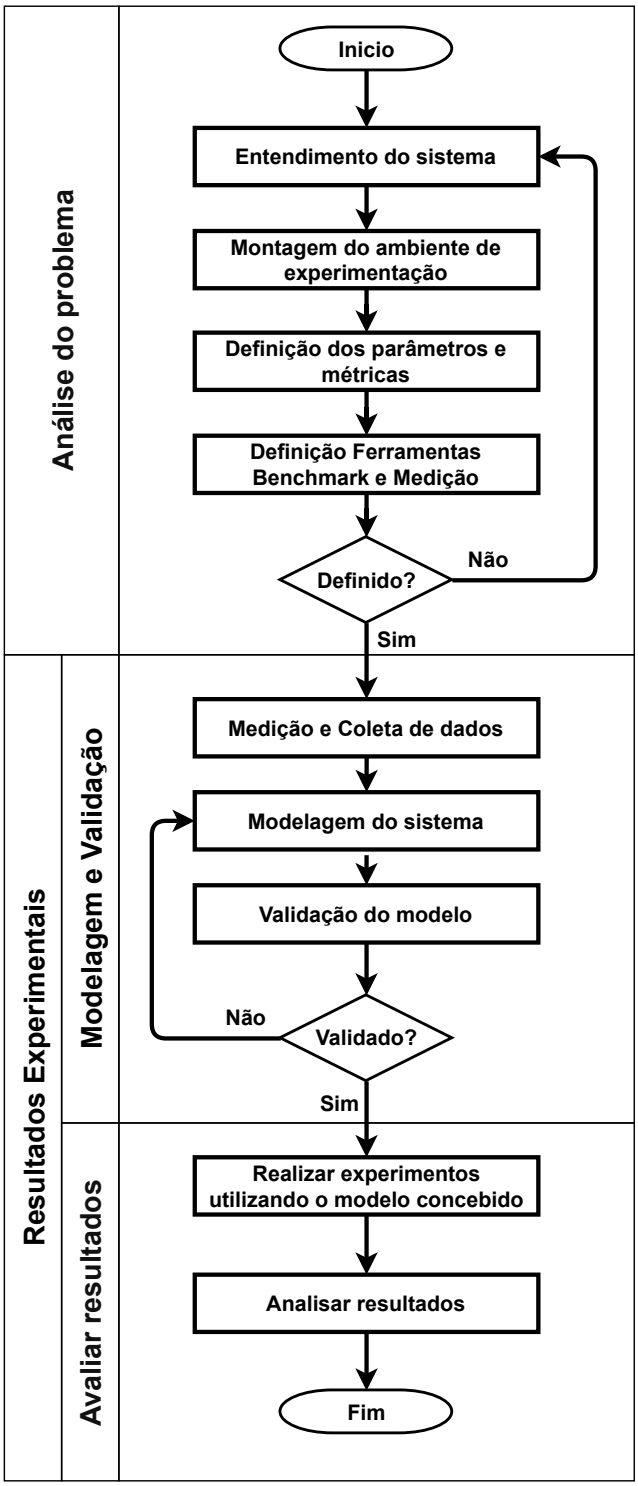
Para validar a abordagem proposta, os experimentos foram realizados em um ambiente controlado, composto por um servidor isolado que executa o banco de dados NoSQL Cassandra. A carga de trabalho gerada e o monitoramento do consumo de energia foram executados na máquina cliente, utilizando, respectivamente, a ferramenta YCSB e o *Measurement Server*. Essa configuração permitiu avaliar o impacto das configurações do banco de dados no consumo de energia do sistema.

O fluxograma na Figura 12 descreve a sequência das atividades realizadas. Cada etapa do método é representada por retângulos e segue uma ordem definida pelas setas, com conjuntos de etapas formando fases específicas. Quando necessário, o fluxo pode retornar a uma fase anterior para revisão de requisitos. Losangos no fluxograma indicam pontos de decisão que podem levar a diferentes caminhos, dependendo dos resultados obtidos. Uma etapa só avança se todas as condições forem atendidas, caso contrário, a etapa é repetida ou ajustes são feitos na fase anterior. O resultado do método proposto é a criação de um modelo validado em um sistema real.

As atividades do método adotado são detalhadas a seguir:

- Análise do problema: Consiste no estudo dos sistemas NoSQL com consistência baseada em quorum, identificando características e fatores que impactam o consumo de energia e a qualidade do serviço. Nesta fase, o ambiente de experimentos é configurado para

Figura 12 – Visão Geral do método proposto



Fonte: Próprio autor

observar o sistema em operação e definir os parâmetros que influenciam as métricas de interesse;

- Modelagem e validação: Envolve a representação formal das principais características operacionais do sistema em estudo. Nesta etapa, os dados do sistema real são coletados, o modelo é desenvolvido e, em seguida, validado por métodos estatísticos para garantir sua proximidade com o comportamento real do sistema;
- Avaliar resultados: Após a validação, é aplicado a técnica DoE ao modelo concebido para avaliar a influência dos fatores na métrica de consumo de energia, tanto de forma isolada quanto em conjunto com o desempenho e a disponibilidade. Nesta etapa, são elaboradas as discussões sobre os resultados obtidos.

4.2 ANÁLISE DO PROBLEMA

Esta seção descreve o estudo realizado para definir o problema, compreender o escopo do sistema, configurar o ambiente e identificar os parâmetros e métricas a serem avaliados.

4.2.1 Entendimento do sistema

Para avaliar a consistência em SGBDs NoSQL baseados em quorum, o projetista deve compreender o funcionamento desses sistemas. Isso inclui a identificação de restrições e componentes, além de entender seu impacto na qualidade do serviço. Também é fundamental conhecer as principais soluções utilizadas, destacando suas vantagens e desvantagens.

Nesta etapa, é essencial compreender as métricas mais relevantes avaliadas na literatura para sistemas de armazenamento com consistência baseada em quorum. Esse entendimento permite alinhar a avaliação a padrões reconhecidos e ajustar o modelo para garantir uma análise abrangente e precisa.

4.2.2 Montagem do ambiente de experimentação

Após compreender o sistema, procede-se à montagem e configuração do ambiente experimental. As atividades realizadas incluem:

- Seleção das tecnologias: Envolve a escolha do sistema operacional, SGBD, ferramentas

de *benchmark*, monitoramento e medição;

- Instalação e configuração do ambiente: Apenas as aplicações essenciais para a execução e o monitoramento do sistema são utilizadas, minimizando a interferência de fatores externos. O SGBD é configurado com as definições padrão;
- Gerenciamento do ambiente: O ambiente deve possibilitar o monitoramento detalhado do sistema, permitindo a avaliação de seu comportamento

Para analisar a técnica de quorum, foram empregadas soluções de código aberto, tais como o banco de dados NoSQL Cassandra e o YCSB para simulação de cargas de trabalho. Essas soluções permitem a replicação do experimento. Um protótipo foi inicialmente montado em um ambiente virtual para avaliar e configurar as o SGBD NoSQL e a ferramenta de *benchmark*. Após a validação da arquitetura, os testes foram realizados em ambiente real.

O ambiente real de experimentação consistiu em um servidor dedicado a hospedar o SGBD, mantendo apenas os serviços essenciais. Além disso, uma máquina cliente foi utilizada para gerar a carga de trabalho, executar o *framework* de medição e coletar os parâmetros necessários à validação do modelo. Essa abordagem minimizou a interferência de processos externos, permitindo que o sistema refletisse de forma mais precisa seu comportamento em um ambiente controlado.

4.2.3 Definição dos parâmetros e métricas

Após o entendimento do sistema e a configuração do ambiente, os parâmetros com maior impacto no consumo de energia e em métricas como desempenho e disponibilidade foram selecionados.

Neste estudo, os parâmetros identificados como mais relevantes para SGBDs NoSQL baseados em quorum incluem: o nível de consistência das operações, o fator de replicação dos dados e a demanda de operações (leitura ou escrita). Para avaliar o consumo de energia e o desempenho, foi considerada a capacidade dos servidores. No que diz respeito à disponibilidade, utilizou-se o tempo médio para falha (MTTF) e o tempo médio para reparo (MTTR) dos componentes do sistema. Assim, os parâmetros definidos contemplam aspectos relacionados ao consumo de energia, desempenho e disponibilidade.

4.2.4 Ferramentas de *Benchmark* e Medição

Para avaliar o desempenho e o consumo de energia de sistemas computacionais, é essencial utilizar ferramentas de *Benchmark* e medição. As ferramentas de *Benchmark* possibilitam realizar testes comparativos e medir o desempenho dos sistemas ao configurar e executar cargas de trabalho específicas. Já as ferramentas de medição capturam dados relacionados ao consumo de energia e outros parâmetros relevantes durante os testes.

As subseções a seguir detalham a configuração adotada para o *Benchmark* YCSB e para o *Measurement Server*, um *framework* utilizado para medir o consumo de energia.

4.2.4.1 Configuração YCSB

Para realizar este *benchmark*, é necessária uma ferramenta compatível com os SGBDs, comandos e a carga de trabalho escolhida. A ferramenta deve atender a dois requisitos: definir e carregar o conjunto de dados no SGBD e executar operações para medir o desempenho (tempo de execução). Foi selecionado o YCSB (ver Seção 2.4), que atende a todos os requisitos do estudo, possui bom suporte técnico (COOPER, 2024).

O consumo de energia foi medido por meio de um *framework* específico, descrito na subseção seguinte. Após a preparação dos componentes do sistema, a carga de trabalho do YCSB foi executada na fase de *load*, validando a conexão entre o *Benchmark* e o SGBD Cassandra. Para carregar os dados, o cliente YCSB foi inicializado e configurado para executar a fase de carregamento.

Durante a coleta, é utilizado o comando *run* para executar operações de leitura e escrita, adotando o padrão do YCSB de dez colunas de texto por registro, com tamanho de $1KB$ por campo. Para o experimento, executaram-se 1.000 operações de leitura e 1.000 operações de escrita, ambas com 100% de proporção. Dez *threads* simularam acessos simultâneos. O parâmetro $-target$ não foi definido, permitindo ao YCSB usar a máxima capacidade de vazão do ambiente.

4.2.4.2 *Measurement server*

O *framework* adotado é denominado *Measurement Server* [1]. Essa ferramenta coleta dados relacionados ao consumo de energia e ao tempo de execução de um sistema computacional. Esta ferramenta permite definir a quantidade de amostras que se deseja coletar e insere os valores em um arquivo de texto. Além disso, o *Measurement Server* foi validado em um estudo anterior (FALCãO et al., 2024).

$$P = V \times I \tag{4.1}$$

O *framework* pode ser utilizado com a plataforma Arduino (MEGA2560, 2023), que usa sensores para coletar valores de corrente (I) e tensão (V). A potência (P) pode então ser determinada (Equação 4.1), e, utilizando o tempo de execução, o consumo de energia (em Joules) é estimado (Seção 2.5) (NILSSON; RIEDEL, 2015).

O *Measurement Server* sincroniza o início e o término de uma amostra. Este *framework* coleta e estima valores extremos ou médios relacionados ao tempo de execução e ao consumo de energia com base nos dados da plataforma Arduino. Após registrar a métrica de consumo de energia e o tempo de execução, um *log* é gerado na máquina cliente (ver Apêndice A).

4.3 MODELAGEM E VALIDAÇÃO

Nesta seção, são explicadas as etapas para o desenvolvimento do modelo que avalia o comportamento do sistema. O modelo representa o cliente enviando requisições de leitura e escrita em um sistema de armazenamento de dados que utiliza SGBDs NoSQL baseados em quorum. As etapas descritas anteriormente são pré-requisitos fundamentais para a modelagem.

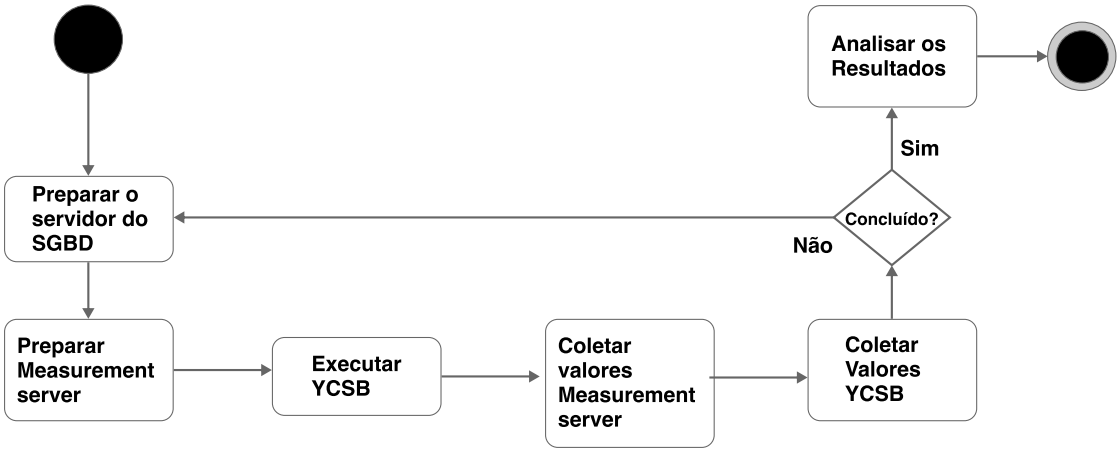
4.3.1 Coleta de dados

A obtenção dos valores de entrada para o modelo ocorre durante a etapa de coleta de dados. Esse processo inicia-se com medições a partir de uma carga de trabalho gerada para o sistema, enquanto seu comportamento é monitorado (BUKH, 1992). A análise do consumo de energia por meio de medições é uma técnica amplamente utilizada na avaliação de sistemas (SHAH; KOTHARI; PATEL, 2022; KENITAR; ARIOUA; YAHYAOUI, 2023; FALCãO et al., 2024). A

[1] Tavares (2023), https://sites.google.com/site/eagtav/measurement-server

escolha adequada da carga de trabalho é fundamental para garantir que os dados reflitam o comportamento típico do sistema. Nesta pesquisa, os valores foram obtidos por medição direta, utilizando parâmetros semelhantes aos de estudos anteriores para avaliar o comportamento do sistema em diferentes cenários.

Figura 13 – Etapas para a coleta de dados



Fonte: Próprio autor

A Figura 13 representa a sequência de passos para a coleta de dados. Primeiramente, o servidor do SGBD é preparado, com a exclusão de todos os *logs* e *caches*. Em seguida, o *Measurement Server* (ver Seção 4.2.4.2) é configurado e iniciado, e os registros de medições anteriores são removidos. O YCSB é executado para gerar a carga de trabalho. Após a execução, os dados do *Measurement Server* e do YCSB são coletados. A coleta é considerada completa quando 100 amostras são obtidas. Caso contrário, a carga de trabalho é executada novamente.

A medição fornece os parâmetros para cada experimento de leitura e escrita, como o tempo de resposta fornecido pelo YCSB, o tempo médio de execução e o consumo médio de energia obtidos pelo *Measurement Server* ao final da coleta das amostras. O número de amostras foi definido com base no teorema do limite central (MACIEL, 2023), que afirma que a distribuição das médias se aproxima de uma curva normal com pelo menos 30 amostras. Escolheram-se 100 amostras para aumentar a precisão estatística e reduzir o impacto de *outliers* (RAVAT et al., 2020). Para o cálculo das médias foi considerado um intervalo de confiança de 95%.

4.3.2 Modelagem do sistema

Nesta pesquisa, o comportamento do sistema é representado por um único modelo. Desenvolver um modelo que reflita com precisão o sistema real exige conhecimento prático. O processo de modelagem inclui etapas de verificação e validação (BUKH, 1992). A verificação assegura que o modelo execute corretamente o fluxo de operações, de maneira semelhante ao sistema real, permitindo a identificação de *deadlocks* e gargalos. A validação, por sua vez, confirma se o modelo representa adequadamente o comportamento do sistema, comparando seus resultados com dados reais.

Com o modelo desenvolvido, é possível aplicar técnicas como análises estacionárias e transientes para obter métricas previamente definidas. *Softwares* especializados, como TimeNet e Mercury (ZIMMERMANN, 2017; BASHIR; LUŠTREK, 2021), facilitam a simulação e análise de modelos baseados em redes de Petri. Esses *softwares* calculam métricas de desempenho utilizando conceitos da Teoria de Filas. Além disso, o Mercury oferece suporte a outras formas de modelagem, como CTMC e Reliability Block Diagrams (RBD) (BASHIR; LUŠTREK, 2021).

Nesta pesquisa, utilizou-se o TimeNet (ZIMMERMANN, 2017), que oferece suporte à modelagem e avaliação de desempenho de redes de Petri estocásticas. A escolha da ferramenta deve-se à sua robustez e à interface gráfica intuitiva, que permite a construção, visualização e refinamento do modelo (GERMAN et al., 1995). Além disso, a interface possibilita a exportação da imagem do modelo, facilitando sua documentação e análise.

4.3.3 Validação

A etapa de validação assegura que o modelo reflete o comportamento do sistema real. Valores coletados do sistema em funcionamento foram utilizados como entrada no modelo. Em seguida, os resultados do modelo foram comparados com os dados medidos. Quando os valores ficaram dentro dos intervalos de confiança das médias obtidas, o modelo foi considerado adequado.

Caso os resultados não estejam alinhados ao comportamento esperado, a modelagem é revisada. Ajustes são realizados e os componentes reavaliados, repetindo-se o fluxo de trabalho. Após a verificação e validação, o modelo pode ser utilizado para analisar situações que seriam economicamente inviáveis em um ambiente de medição em escala reduzida ou que demandariam a interrupção de sistemas em produção.

Para a validação do modelo proposto, utilizou-se o ambiente descrito na Subseção 4.2.2, em conjunto com o *benchmark* (ver Subseção 2.4) e o *framework* selecionado (ver Subseção 4.2.4.2).

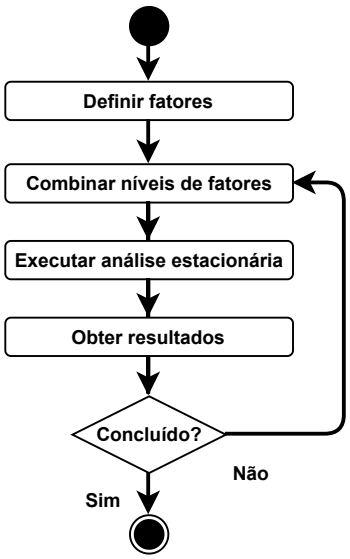
4.4 AVALIAÇÃO DOS RESULTADOS

A análise dos resultados exige um entendimento aprofundado do sistema em estudo e a definição clara dos parâmetros e métricas utilizados nos modelos desenvolvidos. Esta etapa inclui a execução dos experimentos e a análise dos dados obtidos.

4.4.1 Realizar experimentos

Nesta etapa, os experimentos são conduzidos com base no modelo desenvolvido, permitindo a avaliação do comportamento do sistema sem a necessidade de implementar todos os cenários no ambiente real. Após definir as métricas, é essencial planejar os experimentos que serão realizados. Essa atividade identifica os fatores que exercem maior influência no consumo de energia, desempenho e disponibilidade.

Figura 14 – Diagrama de estados UML dos experimentos



Fonte: Próprio autor

Figura 14 apresenta o diagrama das etapas do experimento. Inicialmente, definem-se os fatores e seus respectivos níveis. Em seguida, aplica-se a técnica DoE (ver Subseção 2.3.3), utilizando experimentos fatoriais (l^k), onde (l) são os níveis e (k) os fatores. Os fatores e níveis são combinados em tratamentos, e a análise estacionária é executada para cada combinação. Após a obtenção dos resultados, o processo é repetido para todos os tratamentos definidos no experimento. Essa abordagem possibilita a classificação dos fatores com base no impacto que exercem nas variáveis de resposta.

4.4.2 Analisar resultados

Os resultados obtidos são avaliados por meio da análise da classificação de efeitos dos fatores considerados nos experimentos e pela comparação dos resultados do modelo em relação aos parâmetros, como nível de consistência da operação e fator de replicação.

Entre as questões abordadas nesta análise, destacam-se:

- Como os fatores considerados no projeto de experimentos impactam o consumo de energia, o desempenho e a disponibilidade do sistema;
- Como configurar o sistema para reduzir o consumo de energia;
- Como configurar o sistema para obter melhor desempenho;
- Como configurar o sistema para alcançar maior disponibilidade;
- Como equilibrar desempenho, disponibilidade e consumo de energia para atender às necessidades da solução.

A análise dos resultados compara os valores gerados pelo modelo com diferentes configurações, buscando confirmar o mesmo comportamento em um sistema real. Essa abordagem permite compreender a influência de parâmetros específicos e atuar de forma eficaz nas métricas desejadas. A escolha adequada dos parâmetros contribui para otimizar as características analisadas. Por exemplo, determinar o número ideal de réplicas pode reduzir o consumo de energia sem comprometer a qualidade do serviço oferecido.

4.5 CONSIDERAÇÕES

Este capítulo apresentou o método para avaliar o consumo de energia em SGBDs NoSQL baseados em quorum. Foram detalhadas as etapas do método, incluindo análise do problema, modelagem, validação e avaliação dos resultados. O uso das ferramentas YCSB e *Measurement Server* foi destacado, oferecendo suporte à replicação do estudo por outros pesquisadores. Essa abordagem proporciona uma base sólida para a avaliação e otimização de sistemas, considerando suas características operacionais e métricas de interesse.

5 MODELO PROPOSTO

A modelagem de sistemas é fundamental na análise de parâmetros relevantes para a configuração de ambientes computacionais que visam oferecer alta qualidade de serviço. Essa abordagem possibilita identificar configurações que permitam melhorar o desempenho e a disponibilidade, ao mesmo tempo em que minimizam o consumo de energia. O desenvolvimento de um modelo formal requer a consideração dos componentes envolvidos, dos fluxos de operação e dos eventos que afetam o comportamento do sistema.

Este capítulo apresenta o modelo proposto, baseado em SPN, para estimar o consumo de energia em conjunto com a disponibilidade e o desempenho de bancos de dados NoSQL que utilizam consistência baseada em quorum. O modelo permite analisar diferentes configurações do sistema, sendo relevante para provedores que lidam com altos volumes de transações e variabilidade de carga.

As SPNs foram escolhidas para concepção do modelo por permitirem representar o paralelismo, sincronização e comportamento estocástico, típicos de sistemas distribuídos (ver Subseção 2.3.4.1). Transições imediatas descrevem eventos com tempo zero, enquanto transições temporizadas, com tempos baseados na distribuição exponencial, modelam atrasos de processamento, falhas e reparos (MURATA, 1989). Essas são características comuns em sistemas computacionais.

O modelo é estruturado em blocos, que representam etapas do sistema: chegada do cliente, disponibilidade do sistema, consistência de leitura e consistência de escrita. A explicação segue essa estrutura, detalhando a movimentação dos tokens e o disparo das transições. Também são apresentados os operadores utilizados para estimar as métricas analisadas.

O objetivo do modelo é fornecer uma ferramenta de apoio à análise e comparação de configurações de sistemas NoSQL, contribuindo para o desenvolvimento de soluções mais eficientes e alinhadas às demandas de aplicações no contexto de *Big Data*.

5.1 SISTEMA CONSIDERADO

Para a construção do modelo, considera-se um banco de dados NoSQL que adota a técnica de consistência baseada em quorum. Esses sistemas são amplamente utilizados em aplicações que demandam escalabilidade, desempenho e alta disponibilidade, características presentes em

ambientes de *Big Data* e aplicações em tempo real, como redes sociais e serviços de *streaming*.

O sistema pode ser composto por um ou mais servidores físicos (*hosts*), responsáveis por fornecer os recursos computacionais às réplicas do SGBD. As réplicas podem estar alocadas em um único *host* ou distribuídas entre vários, o que contribui para redundância e tolerância a falhas.

A técnica de quorum (ver Seção 2.2.1) permite gerenciar a consistência entre as réplicas do banco de dados nas operações de leitura e escrita. Essa técnica é definida por três parâmetros principais:

- Fator de Replicação (RF): Número total de réplicas no sistema, que determina a redundância dos dados e o nível de tolerância a falhas;
- Quorum de Leitura (R): Número mínimo de réplicas necessárias para validar uma operação de leitura.
- Quorum de Escrita (W): Número mínimo de réplicas que devem confirmar uma operação de escrita.

A relação entre esses parâmetros permite ajustar a configuração do sistema às suas necessidades. No caso de consistência eventual ($R + W \leq RF$), as operações de leitura podem ser concluídas mesmo que nem todas as réplicas estejam atualizadas. Já para consistência forte ($R+W > RF$), todas as réplicas precisam estar atualizadas antes de confirmar uma operação de leitura, o que pode aumentar a latência e o consumo de energia.

Durante uma operação de leitura, o SGBD compara as versões dos dados e seleciona a mais recente. Na escrita, a operação é considerada bem-sucedida após validação pelo quorum mínimo, e a replicação para as demais réplicas ocorre de forma assíncrona. Esse gerenciamento de consistência possibilita equilibrar métricas como disponibilidade, desempenho e consumo de energia, adaptando o sistema às necessidades das aplicações.

Tanto os servidores físicos quanto as máquinas virtuais estão sujeitos a falhas, que podem impactar diretamente o desempenho e a disponibilidade do sistema. Para modelar esses eventos, pode-se utilizar a latência e os tempos médios de falha e reparo (ver Subseção 2.3.1).

As métricas avaliadas no modelo são:

- Consumo de Energia: Estimado com base no número médio de réplicas envolvidas nas operações e no impacto de falhas e reparos;

- Disponibilidade: Probabilidade de o sistema estar operacional, considerando MTTF e MTTR;
- Desempenho: Estimado a partir da latência e influenciado pelo nível de consistência e fator de replicação.

O modelo analisa o impacto de diferentes configurações sobre essas métricas, permitindo identificar aquelas mais adequadas aos requisitos de cada aplicação.

5.2 MODELO

A modelagem foi realizada utilizando SPN para representar o comportamento de um sistema NoSQL que adota consistência baseada em *quorum*. O modelo permite estimar, de forma conjunta, o consumo de energia, o desempenho e a disponibilidade sob diferentes configurações.

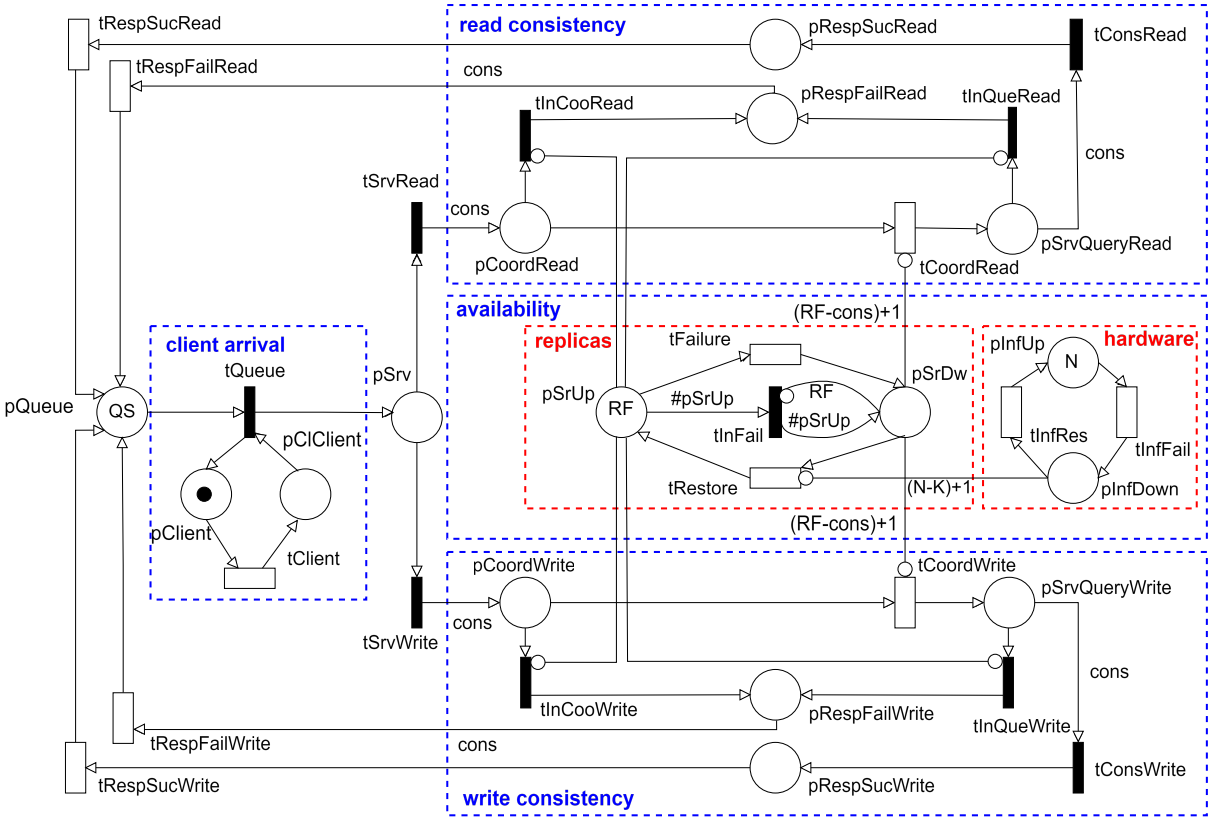
A abordagem considera N servidores físicos que podem hospedar uma ou mais réplicas do SGBD. O número de réplicas é definido pelo Fator de Replicação (RF), enquanto o nível de consistência é representado por $cons$. O sistema é considerado operacional quando, no mínimo, K servidores e $cons$ réplicas estão funcionando, respeitando as condições $K \leq N$ e $cons \leq RF$.

A estrutura geral do modelo é apresentada na Figura 15. A modelagem é composta por quatro blocos:

- Chegada do cliente (*Client Arrival*): representa a chegada dos clientes e a fila de requisições;
- Disponibilidade (*Availability*): modela o estado operacional dos servidores e das réplicas, incluindo falhas e reparos;
- Consistência de Leitura (*Read Consistency*): representa as operações de leitura, verificando o quorum necessário;
- Consistência de Escrita (*Write Consistency*): representa as operações de escrita, verificando o quorum necessário.

Esses blocos interagem de forma integrada para representar o funcionamento do sistema sob diferentes configurações. As próximas subseções descrevem cada bloco, detalhando os

Figura 15 – Modelo para avaliar consumo de energia, desempenho e disponibilidade



Fonte: Próprio autor

componentes da SPN, o fluxo de disparo das transições e os operadores utilizados para a avaliação das métricas.

5.2.1 Chegada do cliente (*Client Arrival*)

O bloco chegada do cliente representa a chegada de solicitações ao sistema. A transição temporizada *tClient* retrata o intervalo entre as requisições geradas por um cliente (cl), registradas no lugar *pClient*. Quando a transição *tClient* dispara, o token é movido para o lugar *pClClient*. Se o número de tokens em *pQueue* for maior que zero, a transição *tQueue* dispara e o token é movido para o lugar *pSrv*, representando que o sistema aceitou a requisição. Caso contrário, o sistema aguarda a liberação de espaço na fila, o que reflete a limitação em processar solicitações quando a capacidade é excedida.

Esse bloco permite controlar o fluxo de entrada e observar os efeitos do número de requisições aceitas pelo sistema.

5.2.2 Disponibilidade (*Availability*)

O bloco de disponibilidade representa o estado operacional dos servidores físicos e das réplicas do sistema. O lugar *pInfUp* armazena os *tokens* que representam os servidores disponíveis, enquanto o lugar *pSrUp* representa as réplicas ativas. A transição temporizada *tInfFail* modela falhas em servidores, removendo *tokens* de *pInfUp*, enquanto *tFailure* representa falhas em réplicas, removendo *tokens* de *pSrUp*. Ambas consideram o tempo médio de falha (MTTF).

Os reparos são modelados pelas transições temporizadas *tInfRes* e *tRestore*, que movem os *tokens* novamente nos lugares *pInfUp* e *pSrUp*, respectivamente, com parâmetros definidos pelo respectivo tempo médio de reparo (MTTR). O número de *tokens* nos lugares *pInfUp* e *pSrUp* determina a disponibilidade do sistema para atender às requisições das operações de leitura e escrita. Esse bloco influencia diretamente o comportamento dos demais blocos, pois a indisponibilidade de servidores ou réplicas pode influenciar o processamento de operações, impactando o desempenho e o consumo de energia.

5.2.3 Consistência de Leitura (*Read Consistency*)

O bloco de consistência de leitura representa a execução das operações de leitura do sistema. Esse bloco é ativado quando um *token* é encaminhado ao nó coordenador de leitura (*pCoordRead*), habilitando a transição temporizada *tCoordRead*. Essa transição representa o tempo de comunicação entre o nó coordenador e as demais réplicas envolvidas na operação.

O disparo da transição *tCoordRead* é condicionado à verificação do número de réplicas disponíveis no lugar *pSrUp*, por meio de um arco inibidor. Esse arco impede o disparo caso a quantidade de réplicas seja insuficiente para satisfazer o quorum de leitura. O peso do arco é calculado como $(RF - cons) + 1$, garantindo que a operação ocorra apenas se ao menos $cons$ réplicas estiverem ativas.

Se o *quorum* de leitura for atendido, a operação é considerada bem-sucedida, e a transição imediata *tRespSucRead* é disparada. Caso contrário, a transição *tRespFailRead* é ativada, indicando falha na leitura por indisponibilidade de réplicas suficientes. Esse bloco permite avaliar o impacto das réplicas no consumo de energia e na latência durante a execução de leituras sob diferentes configurações de quorum.

5.2.4 Consistência de escrita (Write Consistency)

O bloco de consistência de escrita modela o processamento das operações de escrita no sistema. A transição temporizada *tCoordWrite* é habilitada quando um *token* alcança o lugar *pCoordWrite*, indicando que uma requisição de escrita foi encaminhada ao nó coordenador. Essa transição representa o tempo necessário para o coordenador comunicar-se com as réplicas envolvidas na operação.

Antes do disparo de *tCoordWrite*, um arco inibidor conectado ao lugar *pSrUp* verifica a disponibilidade de réplicas ativas. O peso do arco é definido como $(RF - W) + 1$, impedindo que a transição seja disparada caso o número de réplicas disponíveis seja inferior ao exigido pelo *quorum* de escrita.

Quando o quorum é atendido, a operação prossegue com o disparo da transição imediata *tRespSucWrite*, sinalizando a conclusão da escrita. Caso contrário, a transição *tRespFailWrite* é ativada, indicando falha devido à indisponibilidade de réplicas suficientes. Esse bloco permite analisar como diferentes configurações de escrita afetam a latência e o consumo de energia no sistema durante a respectiva operação.

5.2.5 Estimativas das métricas

A avaliação do modelo proposto foi realizada com base em três métricas principais: consumo de energia, disponibilidade e desempenho. A estimativa dessas métricas utiliza operadores e expressões matemáticas, fundamentadas no comportamento do sistema modelado por SPN.

Os operadores utilizados representam expressões auxiliares para análise e estimativa das métricas consideradas neste trabalho:

- $\#p$ indica o número de *tokens* em um lugar p;
- $P\{exp\}$ representa a probabilidade de uma expressão lógica exp ser verdadeira. No presente trabalho, esse operador é calculado como a soma das probabilidades de estado em cada CTMC que atendem à condição definida por $exp.$;
- $E\{\#p\}$ fornece o número médio de *tokens* em um lugar p;

Tabela 2 – Atributos das transições do modelo proposto

Transição	**Tipo**	**Atraso**	**Concorrencia**
tFailure	exponencial	MTTF	infinite server
tInFail	exponencial	MTTF	infinite server
tRestore	exponencial	MTTR	infinite server
tInRes	exponencial	MTTR	infinite server
tInCooRead (*tInCooWrite*)	imediata	-	-
tInQueRead (*tInQueWrite*)	imediata	-	-
tClient	exponencial	CL	single server
tQueue	imediata	-	-
tSrvRead (*tSrvWrite*)	imediata	-	-
tCoordRead (*tCoordWrite*)	exponencial	CC	infinite server
tConsRead (*tConsWrite*)	imediata	-	-
tRespFailRead (*tRespFailWrite*)	exponencial	resp	infinite server
tRespSucRead (*tRespSucWrite*)	exponencial	resp	infinite server

A Tabela 2 apresenta as transições do modelo, com suas respectivas características. O consumo de energia para requisições de leitura (E_{read}) e escrita (E_{write}) é estimado com base no número médio de réplicas envolvidas em cada operação e nas respostas bem-sucedidas enviadas ao cliente, ao longo de um período T. A expressão $E\{\#pCoordRead\}$ representa a média de réplicas que processam a leitura, enquanto $E\{\#pRespSucRead\}$ refere-se às réplicas que retornam uma resposta bem-sucedida.

Da mesma forma, $E\{\#pCoordWrite\}$ e $E\{\#pRespSucWrite\}$ indicam, respectivamente, o número médio de réplicas envolvidas no processamento e nas respostas da operação de escrita. As variáveis P_{read} e P_{write} correspondem à potência média consumida por uma réplica durante as operações de leitura e escrita, respectivamente, podendo ser obtidas por medição ou documentação técnica.

Para operações de leitura, o consumo de energia é estimado pela Equação 5.1, considerando o número médio de réplicas que participam do processamento e das respostas bem-sucedidas multiplicado por P_{read} e pelo período T. Assim, E_{read} representa o consumo de energia para operações de leitura no período analisado.

$$E_{read} = [E\{\#pCoordRead\} + E\{\#pRespSucRead\}] \times P_{read} \times T \tag{5.1}$$

Para operações de escrita, o consumo de energia é calculado pela Equação 5.2. A expressão $(E\{\#pRespSucWrite\} \times (RF - cons))$ representa o impacto das réplicas adicionais devido

à consistência eventual. A expressão $(RF - cons)$ indica o número de réplicas que executam a escrita após a confirmação de sucesso. Por exemplo, se o fator de replicação (RF) for 3 e o nível de consistência $(cons)$ for 2, então 1 réplica adicional processará a escrita após a confirmação.

$$E_{write} = [E\{\#pCoordWrite\} + E\{\#pRespSucWrite\} + (E\{\#pRespSucWrite\} \times (RF - cons))] \times P_{write} \times T \quad (5.2)$$

O consumo total do sistema (E_{sys}) é a soma do consumo de energia para operações de leitura e escrita, conforme a Equação (5.3).

$$E_{sys} = E_{read} + E_{write} \quad (5.3)$$

A disponibilidade do sistema é calculada pela Equação 5.4, que determina a probabilidade de o número médio de réplicas operacionais ser maior ou igual ao nível de consistência $(cons)$ definido. Uma probabilidade mais alta de réplicas ativas indica maior disponibilidade do sistema.

A disponibilidade do sistema é calculada pela Equação 5.4, que determina a probabilidade de o número médio de réplicas operacionais ser maior ou igual ao nível de consistência $(cons)$ definido. Uma probabilidade mais alta de réplicas ativas indica maior disponibilidade do sistema

$$A = P\{\#pSrUp \geq cons\} \quad (5.4)$$

A taxa de transferência de leitura é estimada em 5.5 e a de escrita em 5.6, onde $resp_{read}$ e $resp_{write}$ são os atrasos associados às transições $tRespSucRead$ e $tRespSucWrite$ (respostas bem-sucedidas), respectivamente. A latência de leitura é $(thput_{read})^{-1}$, enquanto a de escrita é $(thput_{write})^{-1}$. A taxa de transferência do sistema é estimada na Equação 5.7, e a latência respectiva é $(thput_{sys})^{-1}$.

$$thput_{read} = P\{\#pRespSucRead > 0\} \times 1/resp_{read} \quad (5.5)$$

$$thput_{write} = P\{\#pRespSucWrite > 0\} \times 1/resp_{write} \quad (5.6)$$

$$thput_{sys} = thput_{read} + thput_{write} \quad (5.7)$$

5.3 EXEMPLO DE FLUXO OPERAÇÃO DO MODELO

Para ilustrar o funcionamento do modelo proposto, apresenta-se o fluxo de execução de operações de leitura e escrita, que podem ocorrer de forma paralela ou isolada, conforme a necessidade. O modelo considera tanto requisições bem-sucedidas quanto falhas devido a indisponibilidades. Os parâmetros iniciais incluem: nível de consistência ($cons$) igual a dois, fator de replicação (RF) igual a três e número de servidores (N) igual a um. As Figuras do Apêndice D mostram o fluxo de execução com base na marcação da rede, permitindo visualizar o disparo das transições e a movimentação dos *tokens*.

Esses exemplos ajudam a reforçar a compreensão das regras implementadas e da lógica de avaliação das métricas. São apresentados casos de operações bem-sucedidas e falhas em requisições de leitura e escrita, de acordo com as configurações de quorum e disponibilidade do sistema.

5.3.1 Exemplo de fluxo operação de leitura bem-sucedida

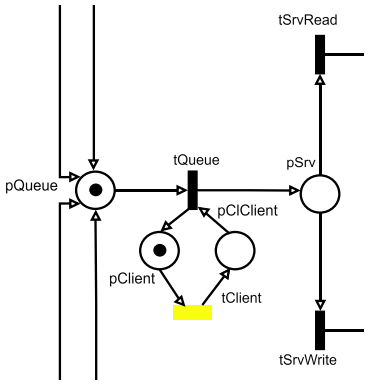


Figura 16 – *token* posicionado no lugar *pClient*

A Figura 16 mostra o início da operação de leitura, com a chegada da requisição pelo cliente. O *token* é posicionado no lugar *pClient*, e, após o disparo da transição temporizada *tClient*, é transferido para *pClClient* (Figura 17), habilitando a transição imediata *tQueue*.

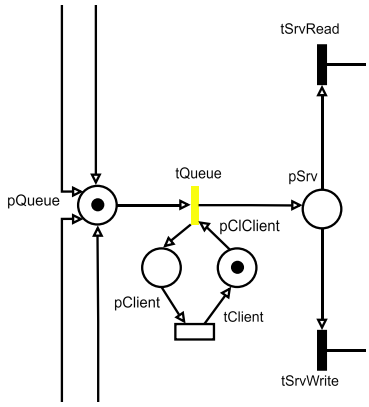


Figura 17 – *token* posicionado no lugar *pClClient*

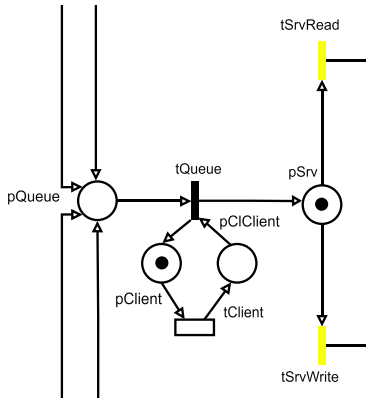


Figura 18 – *token* posicionado no lugar *pSrv*.

Na Figura 18, a transição *tQueue* é disparada, e o token segue para *pSrv*, representando que o sistema aceitou a requisição. As transições temporizadas *tSrvRead* e *tSrvWrite* são ativadas. Para a operação de leitura ao disparar a transição *tSrvRead* o *token* é movido para o lugar *pCoordRead*, indicando que a requisição foi encaminhada ao nó coordenador.

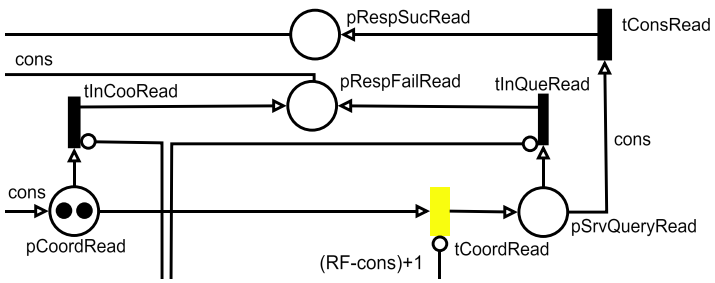


Figura 19 – *tokens* posicionados no lugar *pCoordRead*.

A Figura 19 mostra o momento em que a transição temporizada *tCoordRead* é habilitada. Seu disparo depende da disponibilidade de réplicas suficientes para satisfazer o *quorum* configurado. Uma vez atendida essa condição, o *token* é transferido para *pSrvQueryRead*, indicando o fim do processamento da operação (Figura 20).

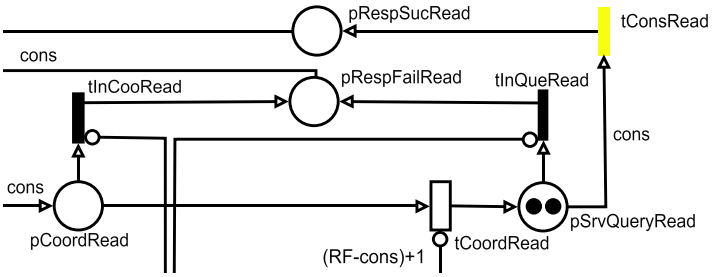


Figura 20 – *tokens* posicionados no lugar *pSrvQueryRead*

Com a finalização do processamento por todas as réplicas envolvidas, o *token* é encaminhado para *pRespSucRead*, que representa o envio da resposta bem-sucedida ao cliente (Figura 21). Em seguida, a transição imediata *tRespSucRead* é disparada, concluindo a operação de leitura.

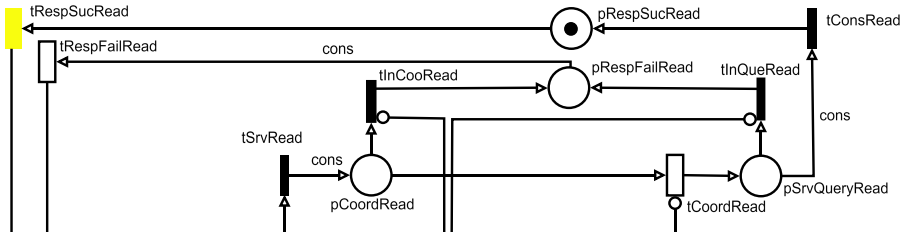


Figura 21 – *tokens* posicionados no lugar *pRespSucRead*

Esse exemplo demonstra a trajetória completa de uma leitura bem-sucedida no modelo, evidenciando a movimentação dos tokens e os critérios de ativação das transições conforme a disponibilidade do sistema e os parâmetros de consistência. O fluxo para uma operação de escrita é semelhante ao de leitura.

5.3.2 Falha em Requisições de Leitura

No caso de falha durante uma operação de leitura, após o disparo de *tSrvRead*, os *tokens* são enviados ao lugar *pCoordRead*. No entanto, se a transição *tFailure* disparar antes que o número de *tokens* em *pCoordRead* atinja o valor de $cons$, a operação de leitura falha. Nesta situação, a transição *tCooRead* é habilitada, enviando um *token* ao lugar *pRespFailRead*, que representa o status de envio da resposta com falha para o cliente. Para a operação de escrita, o fluxo de falha ocorre de forma semelhante.

5.4 CONSIDERAÇÕES

Este capítulo apresentou o modelo concebido para representar o funcionamento de sistemas NoSQL com consistência baseada em quorum. A modelagem foi estruturada em blocos funcionais que representam a chegada de clientes, a disponibilidade das réplicas e os processos de leitura e escrita, com base na técnica de redes de Petri estocásticas. Além disso, o modelo é explicado por meio de exemplos do fluxo de execução, destacando a movimentação dos tokens e o disparo das transições. O modelo proposto é utilizado nos experimentos descritos no próximo capítulo, que avaliam o impacto da consistência eventual no consumo de energia, desempenho e disponibilidade.

6 RESULTADOS EXPERIMENTAIS

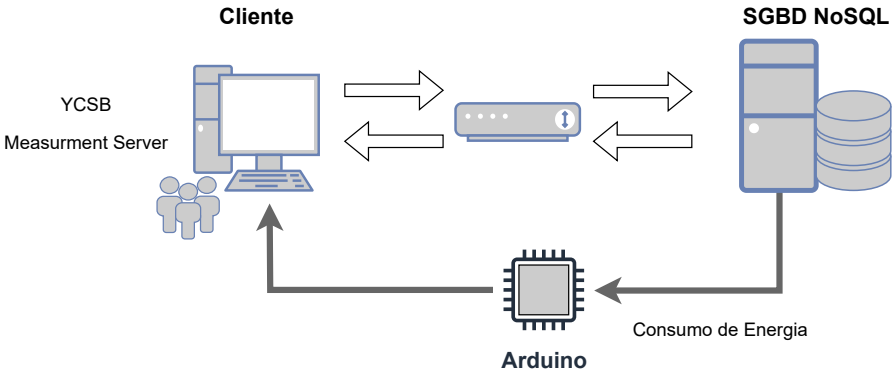
Este capítulo apresenta os experimentos realizados para demonstrar a viabilidade prática do modelo proposto. Primeiramente, descreve-se a validação do modelo. Em seguida, avalia-se quantitativamente a influência da consistência eventual no consumo de energia, na disponibilidade e no desempenho. Por fim, discute-se um estudo de caso que exemplifica a aplicação prática do modelo em um sistema real.

Tabela 3 – Configurações das máquinas

Equipamento	Sistema Operacional	Memória RAM	CPU
Máquina Cliente	Windows 10	16 GB	Intel(R) Core i7
Servidor SGBD	Ubuntu 22.04	8 GB	Intel Xeon(R) (4 nucleos)

A Tabela 3 apresenta a configuração de cada máquina utilizada para validar o modelo. O sistema adotado (Figura 22) considera uma réplica do SGBD Cassandra 4.1 (CASSANDRA, 2024), que executa em uma CPU Intel Xeon(R) (4 núcleos) com 8GB de RAM e Ubuntu 22.04. Uma máquina distinta, utilizada para monitoramento, simula as requisições do cliente, gerando a carga de trabalho e coletando dados sobre o tempo de execução e o consumo de energia.

Figura 22 – Sistema para validação do modelo



Fonte: Próprio autor

Esta pesquisa utiliza a técnica de DoE (ver Subseção 2.3.3) com um desenho fatorial de dois níveis (2^k). Apresenta-se um ranking dos efeitos, que indicam a mudança na resposta devido à alteração no nível do fator. Esses rankings são organizados em ordem decrescente, considerando os valores absolutos, demonstrando como a proposta pode auxiliar o arquiteto

do sistema. O consumo de energia é apresentado em joule (J) e quilowatt-hora (kWh) para melhor visualização.

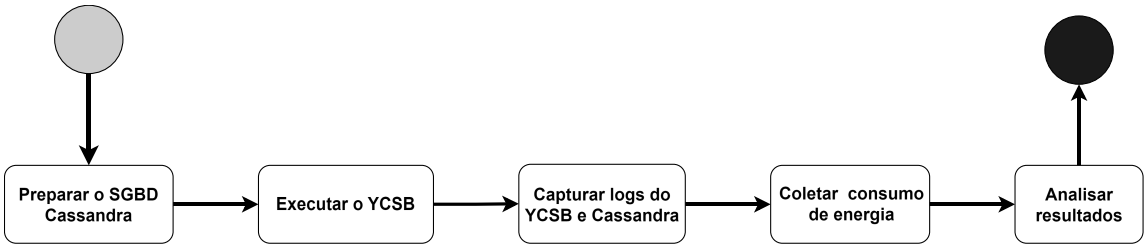
6.1 VALIDAÇÃO DO MODELO

Este experimento valida o modelo proposto ao comparar o consumo de energia estimado com o observado no sistema real. O desempenho é medido pelo tempo de resposta, isto é, o intervalo entre o início e a conclusão de uma operação bem-sucedida. A disponibilidade é definida como a probabilidade de o sistema estar operacional em um dado momento. Este estudo dá continuidade à pesquisa de (ARAÚJO et al., 2024), que validou métricas de desempenho e disponibilidade. A análise estacionária foi utilizada devido às propriedades do modelo que permitem essa abordagem (ver Subseção 2.3.4)(MACIEL, 2023).

Os resultados foram comparados com um intervalo de confiança de 95%, obtido a partir dos dados do sistema. Esse intervalo é calculado com base em dados amostrais e indica a margem de incerteza associada à estimativa de um parâmetro populacional. Nesta pesquisa, a validação do consumo de energia utiliza intervalos de confiança para a média. Para isso, verifica-se se a amostra é aleatória e se a população segue uma distribuição normal ou se o tamanho da amostra (n) é maior ou igual a 30. Nesses casos, o Teorema do Limite Central assegura que as médias amostrais seguem uma distribuição normal, justificando a aplicação dessa abordagem estatística (MONTGOMERY; RUNGER, 2020; AHSANULLAH; KIBRIA; SHAKIL, 2014).

Para a validação, a hipótese nula H_0 afirma que o resultado do modelo (M_r) é semelhante ao resultado do sistema (S_r) dentro de um intervalo de confiança de 95%. Em outras palavras, para H_0 com um nível de significância $\alpha = 0,05$, tem-se ($M_r - S_r = 0$).

Figura 23 – Visão geral do processo de coleta de dados, validação do modelo e análise dos resultados



Fonte: Próprio autor

A Figura 23 apresenta uma visão geral das etapas de modelagem e validação descritas na Seção 4.3, assim como da avaliação dos resultados na Seção 4.4. A preparação do banco de dados envolve a ativação e configuração de logs para monitorar o uso de recursos e registrar o desempenho. O YCSB é sincronizado com o Cassandra para gerar a carga de trabalho. Durante a execução das requisições de leitura e escrita, são coletados logs tanto do YCSB quanto do Cassandra. Por fim, os dados coletados são analisados para validar o modelo, verificando a precisão na estimativa do consumo de energia.

Tabela 4 – Parâmetros do YCSB

Parâmetros	**Valor**
data size	1KB (*default*)
request distribution	*zipfian*
record count	30.000
operation count	1.000
read all fields	*true* (*default*)

A Tabela 4 apresenta os parâmetros definidos na ferramenta YCSB.

- *data size*: Refere-se ao tamanho de cada registro durante as requisições. Optou-se pelo valor padrão da ferramenta, seguindo práticas comuns em pesquisas semelhantes (ARAÚJO et al., 2024; FALCãO et al., 2024);
- *request distribution*: Trata-se da distribuição de probabilidade usada para a seleção das leituras de dados. Foi escolhida a distribuição *zipfian* devido à aleatoriedade de acesso aos dados no SGBD;
- *record count*: Especifica a quantidade de registros já presentes no banco de dados. Selecionou-se um valor que torna provável a leitura de dados no disco do servidor, com a possibilidade de acesso ocasional aos dados;
- *operation count*: Indica o número de operações que serão executadas no banco de dados.
- *read all fields*: É uma variável lógica (*true* ou *false*) que permite a leitura de todos os campos de um registro. Esta opção foi mantida para alinhamento com pesquisas semelhantes (ARAÚJO et al., 2024; FALCãO et al., 2024).

A Tabela 5 apresenta os tempos médios obtidos a partir de 100 amostras com 1000 operações cada. Os tempos $tCoordRead$ e $tCoordWrite$, obtidos dos logs do Cassandra,

Tabela 5 – Tempo das transições para a validação do consumo de energia

Transição	Tempo
tCoordRead	$14,33ms$
tCoordWrite	$15,51ms$
tRespSucRead (*tRespSucWrite*)	$0,49s$
tClient	$12,96ms$

foram de $14,33ms$ para leitura e $15,51ms$ para escrita. O tempo de resposta bem-sucedida ($tRespSucRead$ e $tRespSucWrite$) é de $0,49s$ para ambas as operações. O tempo médio entre requisições ($tClient$) foi de $12,96ms$. Para operações de leitura, foram medidos um consumo médio de $569,94J$ e um tempo médio de execução de $13,57s$, resultando em uma potência média de aproximadamente $42W$ ($42J/s$). Para operações de escrita, o consumo médio foi de $549,58J$ no mesmo intervalo de tempo, o que corresponde a uma potência média de $40W$ ($40J/s$). Esses valores foram obtidos por meio de medições no sistema avaliado, conforme detalhado no Apêndice C.

Utilizando o TimeNet (ver Subseção 4.3.2), o modelo proposto estimou o consumo de energia em $571.8447J$ para requisições de leitura e $550.5650J$ para operações de escrita. Em relação ao sistema real, os intervalos de confiança são $[568.7755, 574.9139J]$ para leitura e $[549.3761, 552.0693J]$ para escrita. Como os valores médios estimados estão dentro desses intervalos, não há evidências estatísticas para rejeitar H_0. Portanto, não há indícios de que o modelo não represente o comportamento do sistema real.

6.2 EXPERIMENTOS

Esta seção apresenta a aplicação do modelo proposto, analisado por meio de um projeto fatorial l^k, utilizando a abordagem DoE (ver Subseção 2.3.3). Os resultados incluem a classificação dos efeitos das interações entre os fatores adotados para cada métrica, bem como os resultados dos tratamentos obtidos no experimento.

As métricas consideradas são o consumo de energia, a disponibilidade e o desempenho do sistema. O desempenho foi medido por meio da latência, enquanto a disponibilidade foi avaliada com base no tempo de inatividade (*downtime*). Essas métricas permitem avaliar a viabilidade de configurações de sistemas NoSQL baseados em quorum. No entanto, é necessária uma análise criteriosa para identificar o equilíbrio mais adequado aos requisitos do sistema.

6.2.1 Consumo de Energia

Este projeto de experimento tem o objetivo de avaliar o consumo de energia do sistema em diferentes tratamentos. O experimento considera cinco fatores com dois níveis ($k = 5$): (i) nível de consistência ($cons$) - 1, 3; (ii) fator de replicação (RF) - 3,5; (iii) proporção de operações ($Prop_{RW}$) - 0,6 leitura / 0,4 escrita, 0,8 leitura / 0,2 escrita; (iv) consumo de energia para leitura (P_{read}) - $32J/s$, $40J/s$; e (v) consumo de energia para escrita (P_{write}) - $40J/s$, $48J/s$.

A escolha dos níveis para cada fator foi baseada em estudos representativos (GOMES et al., 2023; ARAÚJO et al., 2024) e em dados obtidos por medição no sistema real. O fator consistência ($cons$) permite avaliar a consistência eventual (1) e forte (3). O valor mínimo de RF $= 3$ permite avaliar a leitura e escrita com consistência forte, enquanto RF $= 5$ representa um ambiente com maior redundância, contribuindo para avaliar o impacto de mais réplicas no consumo. A proporção 0,8 leitura / 0,2 escrita reflete cargas típicas de leitura intensiva em aplicações como *e-commerce* e redes sociais (LEI et al., 2024).

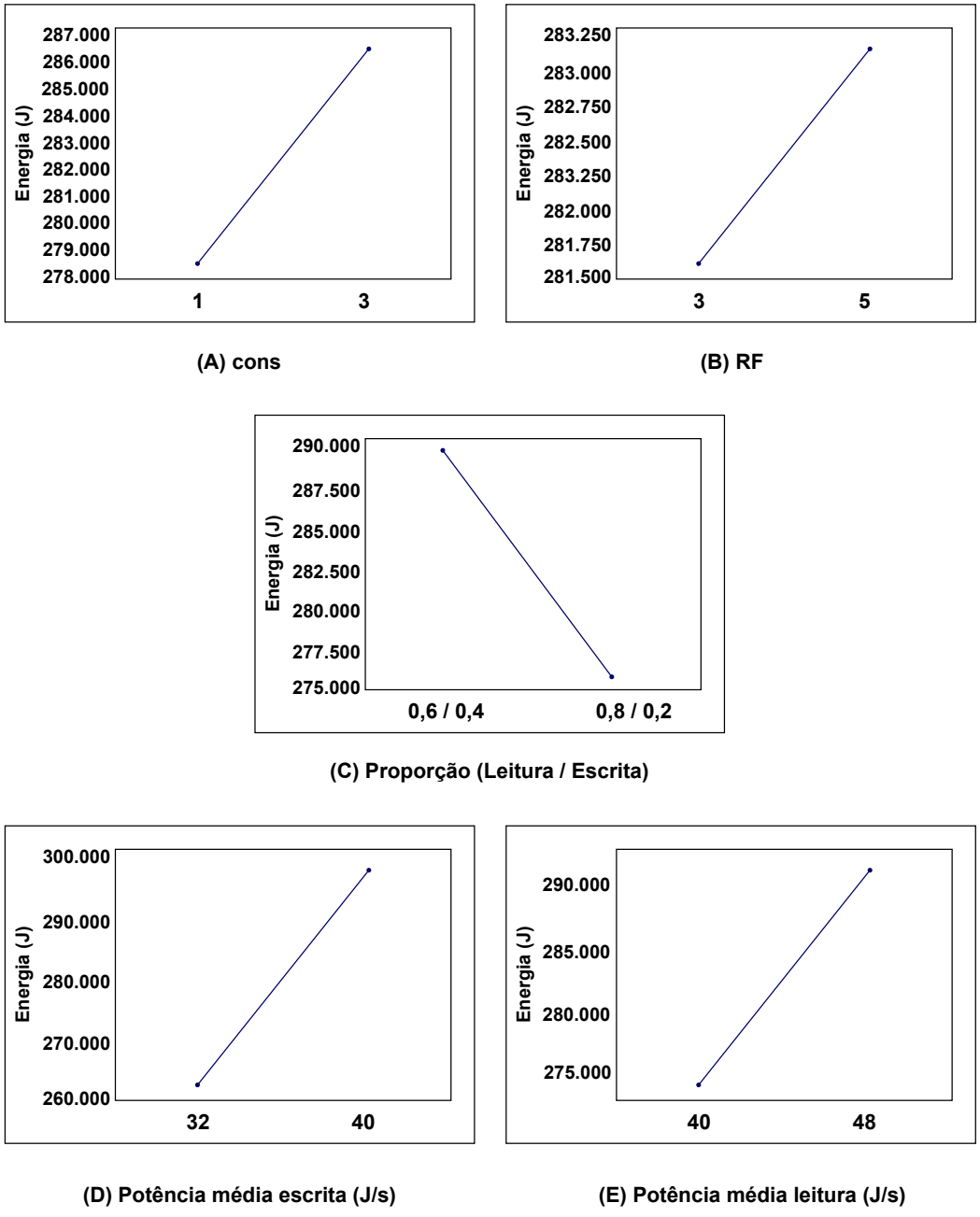
Os valores de potência (P_{read} e P_{write}) foram medidos diretamente no ambiente experimental com apoio do *Measurement Server*, considerando variações esperadas na execução das cargas de trabalho. P_{read} e P_{write} indicam, respectivamente, a demanda de energia devido a variações na carga de trabalho de leitura e escrita em um único servidor. A carga de trabalho pode variar conforme o tamanho dos registros, as configurações específicas do SGBD ou a otimização das queries (CHOU; RAGHAVAN; LAHIRI, 2018; MAHAJAN; BLAKENEY; ZONG, 2019). Parâmetros como cache de memória, índices e políticas de compactação podem afetar o consumo de energia durante a execução da carga de trabalho. A otimização das queries também é relevante, pois consultas bem otimizadas reduzem operações desnecessárias, diminuindo o consumo de energia.

Tabela 6 – Classificação dos efeitos principais - consumo de energia

Fator / Interações	|Efeito| (J)
P_{read}	40.896
P_{write}	17.877
$Prop_{RW}$	-13.479
$cons$	7.642
RF	1.498
$RF \times Prop_{RW}$	-842
$cons \times P_{read}$	641
$cons \times Prop_{RW}$	619
$RF \times P_{write}$	136

Neste experimento, foi considerado apenas um servidor ($N = 1$). A influência da disponibilidade não foi considerada. Portanto, para o servidor físico, os valores para MTTF e MTTR foram mantidos constantes em $2.880h$ e $1h$, respectivamente. Para as réplicas do SGBD, os valores são $1.440h$ para MTTF e $0,25h$ para MTTR. Em relação aos atrasos nas transições temporizadas, utilizam-se os mesmos da validação (Seção 6.1). Além disso, $1h$ foi definida para o período T.

Figura 24 – Gráficos de efeito – consumo de energia



Após executar os experimentos, é possível classificar os efeitos principais e as interações ($cons \times RF$) em relação ao consumo de energia (ver Tabela 6). Um efeito representa a variação na resposta causada por uma alteração no nível de um fator. A classificação é ordenada de maneira decrescente, considerando os valores absolutos dos efeitos (MONTGOMERY, 2017).

A Figura 24 exibe os gráficos de efeito que demonstram como os diferentes níveis dos fatores impactam o consumo de energia. Os fatores P_{read} e P_{write} influenciam significativamente o consumo de energia devido à carga de trabalho. O fator $cons$, tem um impacto maior do que RF, pois réplicas adicionais precisam ser consideradas para confirmar uma operação bem-sucedida. Considerando os níveis adotados $cons$ e RF podem afetar o consumo de energia em até 10%. O fator $Prop_{RW}$ tem um efeito na direção oposta, pois o segundo nível tem uma menor proporção de escrita, exigindo a atualização de menos servidores na consistência eventual.

Tabela 7 – Tratamentos para a variação da carga de trabalho

Nº	$cons$	RF	$Prop_{RW}$	P_{read} (J/s)	P_{write} (J/s)	Energia (J)	Energia (kWh)
1	3	5	0,6 leitura / 0,4 escrita	40	48	323.698	0,0899
2	3	3	0,6 leitura / 0,4 escrita	40	48	321.196	0,0892
3	1	5	0,6 leitura / 0,4 escrita	40	48	315.989	0,0878
4	1	3	0,6 leitura / 0,4 escrita	40	48	313.384	0,0871
5	3	5	0,8 leitura / 0,2 escrita	40	48	309.850	0,0861
6	3	3	0,8 leitura / 0,2 escrita	40	48	309.151	0,0859
7	1	5	0,8 leitura / 0,2 escrita	40	48	300.719	0,0835
8	1	3	0,8 leitura / 0,2 escrita	40	48	299.988	0,0833
9	3	5	0,6 leitura / 0,4 escrita	40	40	299.403	0,0832
10	3	5	0,8 leitura / 0,2 escrita	40	40	297.784	0,0827
11	3	3	0,6 leitura / 0,4 escrita	40	40	297.318	0,0826
12	3	3	0,8 leitura / 0,2 escrita	40	40	297.201	0,0826
13	1	5	0,6 leitura / 0,4 escrita	40	40	292.071	0,0811
14	1	3	0,6 leitura / 0,4 escrita	40	40	289.900	0,0805
15	1	5	0,8 leitura / 0,2 escrita	40	40	288.945	0,0803
16	1	3	0,8 leitura / 0,2 escrita	40	40	288.336	0,0801
17	3	5	0,6 leitura / 0,4 escrita	32	48	288.112	0,0800
18	3	3	0,6 leitura / 0,4 escrita	32	48	285.610	0,0793
19	1	5	0,6 leitura / 0,4 escrita	32	48	281.494	0,0782
20	1	3	0,6 leitura / 0,4 escrita	32	48	278.888	0,0775
21	3	5	0,6 leitura / 0,4 escrita	32	40	263.817	0,0733
22	3	5	0,8 leitura / 0,2 escrita	32	48	262.360	0,0729
23	3	3	0,6 leitura / 0,4 escrita	32	40	261.732	0,0727
24	3	3	0,8 leitura / 0,2 escrita	32	48	261.660	0,0727
25	1	5	0,6 leitura / 0,4 escrita	32	40	257.575	0,0715
26	1	3	0,6 leitura / 0,4 escrita	32	40	255.404	0,0709
27	1	5	0,8 leitura / 0,2 escrita	32	48	254.703	0,0708
28	1	3	0,8 leitura / 0,2 escrita	32	48	253.973	0,0705
29	3	5	0,8 leitura / 0,2 escrita	32	40	250.294	0,0695
30	3	3	0,8 leitura / 0,2 escrita	32	40	249.710	0,0694
31	1	5	0,8 leitura / 0,2 escrita	32	40	242.930	0,0675
32	1	3	0,8 leitura / 0,2 escrita	32	40	242.321	0,0673

A Tabela 7 apresenta o consumo de energia para cada tratamento. Valores mais altos estão relacionados a uma maior carga de trabalho (P_{read} e P_{write}) e a uma maior proporção de operações de escrita. Esses valores também estão associados a $cons = 3$ e $RF = 5$, devido à necessidade de servidores adicionais para operações de leitura e escrita. Os diferentes tratamentos deste experimento, aplicando a técnica DoE, demonstra que a configuração do sistema não deve ser negligenciada, pois pode impactar o consumo de energia.

6.2.2 Disponibilidade

O modelo concebido também pode ser adotado para avaliar o consumo de energia em conjunto com a disponibilidade. Para este experimento foram considerados os seguintes fatores ($k = 5$) e níveis ($l = 2$): (i) nível de consistência ($cons$) - 1 e 3; (ii) fator de replicação (RF) - 3 e 5; (iii) $Prop_{RW}$ - 0.6 *leitura* / 0.4 *escrita*, 0.8 *leitura* / 0.2 *escrita*; (iv) $MTTR$ - 0,25h e 1h; e (v) *failure* - 2.880h e 5.000h.

Os níveis dos fatores $cons$, RF e $Prop_{RW}$ foram mantidos em relação ao experimento anterior de consumo de energia (Seção 6.2.1). Para os fatores $MTTR$ e $MTTF$, os níveis também foram definidos com base nos mesmos estudos. As transições $tFailure$ e $tRestore$ representam, respectivamente, o tempo médio entre falhas ($MTTF$) e o tempo médio de reparo ($MTTR$) das réplicas. Esses valores estão associados ao comportamento das réplicas do SGBD, podendo variar conforme o tipo de *hipervisor* adotado.

Os atrasos para transições temporizadas são os mesmos assumidos na etapa de validação (Seção 6.1). P_{read} e P_{write} são mantidos fixos em $42J/s$ e $40J/s$, respectivamente, representando uma carga de trabalho constante. O período T corresponde a 1 ano (8.760h) para uma melhor avaliação do tempo de inatividade do sistema (DT). Esta métrica fornece uma melhor visualização dos efeitos da disponibilidade (A) e pode ser estimada (em horas) usando a Equação 6.1.

$$DT = (1 - A) \times 8760 \tag{6.1}$$

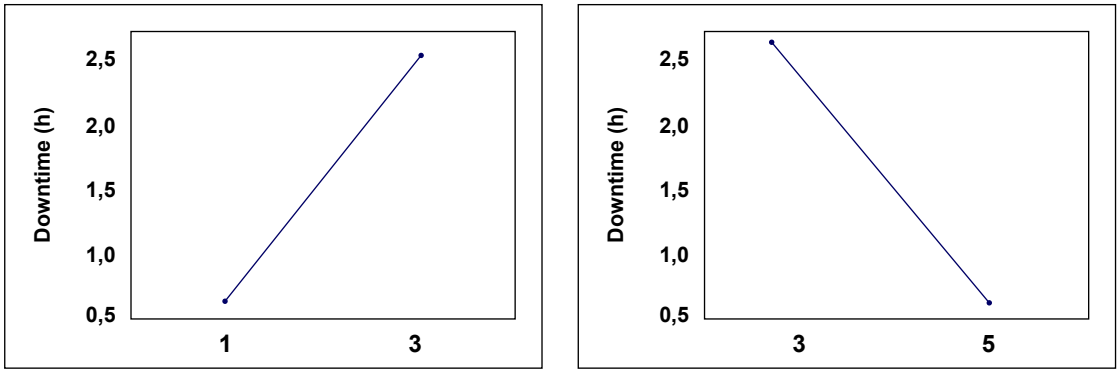
A Tabela 8 apresenta a classificação dos efeitos considerando o tempo de inatividade. Os fatores $cons$, RF e sua interação impactam significativamente o tempo de inatividade, resultando em aproximadamente 2 horas por ano, considerando os níveis adotados. Embora 2 horas pareçam pouco em comparação com 8.760 horas anuais, esse tempo de inatividade pode violar acordos de serviço ou resultar na perda de usuários.

Tabela 8 – Classificação dos efeitos - Tempo de inatividade

Fator / Interação	|Efeito| (h)
$cons$	2,04
RF	-1,95
$cons \times RF$	-1,95
MTTR	1,27
$cons \times MTTR$	1,26
$RF \times MTTR$	-1,18
MTTF	-0,58
$Prop_{RW}$	-0,09

A Figura 25 demonstra como os fatores $cons$ e RF, com seus respectivos níveis, afetam o tempo de inatividade. No nível 1 de $cons$, o tempo de inatividade é de 0,5 horas por ano, aumentando para 2,5 horas no nível 3 (ver Figura 25 (a)). Esse aumento ocorre porque a elevação do nível de consistência requer a sincronização de uma maior quantidade de réplicas antes confirmar uma operação bem-sucedida, o que pode aumentar o tempo de inatividade para assegurar a consistência dos dados. Por outro lado, o fator RF apresenta uma variação inversa a $cons$ para os níveis 3 e 5 (ver Figura 25 (b)). Esse comportamento reflete um aumento na disponibilidade, uma vez que um maior fator de replicação melhora a redundância.

Figura 25 – Gráficos de efeito – Tempo de Inatividade



(A) Nível de Consistência (cons) **(B) Fator de Replicação (RF)**

A Tabela 9 exibe os resultados de cada tratamento. O tempo de inatividade inferior a um minuto é representado como $< 0,01$ horas. Os maiores tempos de inatividade (DT) ocorrem devido à forte consistência ($cons = RF$), pois não há réplicas disponíveis para substituição rápida de um nó com falha. Embora os maiores níveis de $cons$ e RF melhorem a disponibilidade, essa configuração também aumenta o consumo de energia.

Ao comparar os tratamentos Nº 9 e 10 da Tabela 9, ambos com $RF = 5$, a redução do nível de consistência de 3 para 1 diminui o tempo de inatividade (DT) em aproximadamente 87,50% (de 0,72 para 0,09 horas) e reduz o consumo de energia em 2,72%. Esse exemplo demonstra como o modelo pode reduzir simultaneamente o tempo de inatividade e o consumo de energia, permitindo ao projetista identificar configurações que sejam mais adequadas aos requisitos do sistema.

Tabela 9 – Tratamentos para a disponibilidade e o consumo de energia

Nº	$cons$	RF	$Prop_{RW}$	MTTF (h)	MTTR (h)	Energia (J)	Energia (kWh)	DT (h)
1	3	3	0,6 / 0,4	2.880	1,00	2.602.015.965	722,78	8,45
2	3	3	0,8 / 0,2	2.880	1,00	2.601.164.958	722,55	7,83
3	3	3	0,8 / 0,2	5.000	1,00	2.602.041.827	722,79	4,88
4	3	3	0,6 / 0,4	5.000	1,00	2.603.152.683	723,10	1,08
5	3	3	0,6 / 0,4	2.880	0,25	2.603.920.889	723,31	2,05
6	3	3	0,8 / 0,2	2.880	0,25	2.602.935.259	723,04	1,88
7	3	3	0,6 / 0,4	5.000	0,25	2.604.176.676	723,38	1,19
8	3	3	0,8 / 0,2	5.000	0,25	2.603.172.044	723,10	1,08
9	3	5	0,6 / 0,4	2.880	1,00	2.615.548.550	726,54	0,72
10	1	5	0,6 / 0,4	2.880	1,00	2.544.287.285	706,75	0,09
11	3	5	0,8 / 0,2	5.000	0,25	2.605.237.089	723,68	0,04
12	3	5	0,6 / 0,4	5.000	1,00	2.615.764.093	726,60	<0,01
13	3	5	0,6 / 0,4	2.880	0,25	2.615.764.198	726,60	<0,01
14	3	5	0,6 / 0,4	5.000	0,25	2.615.764.198	726,60	<0,01
15	3	5	0,8 / 0,2	2.880	1,00	2.605.248.686	723,68	<0,01
16	3	5	0,8 / 0,2	5.000	1,00	2.605.248.686	723,68	<0,01
17	3	5	0,8 / 0,2	2.880	0,25	2.605.248.756	723,68	<0,01
18	1	5	0,6 / 0,4	5.000	1,00	2.544.314.824	706,75	<0,01
19	1	3	0,6 / 0,4	5.000	1,00	2.532.407.630	703,45	<0,01
20	1	3	0,6 / 0,4	2.880	1,00	2.532.407.629	703,45	<0,01
21	1	5	0,6 / 0,4	2.880	0,25	2.544.314.833	706,75	<0,01
22	1	5	0,6 / 0,4	5.000	0,25	2.544.314.833	706,75	<0,01
23	1	3	0,6 / 0,4	5.000	0,25	2.532.407.645	703,45	<0,01
24	1	3	0,6 / 0,4	2.880	0,25	2.532.407.644	703,45	<0,01
25	1	5	0,8 / 0,2	2.880	1,00	2.524.233.884	701,18	<0,01
26	1	5	0,8 / 0,2	5.000	1,00	2.524.233.884	701,18	<0,01
27	1	3	0,8 / 0,2	5.000	1,00	2.522.363.011	700,66	<0,01
28	1	3	0,8 / 0,2	2.880	1,00	2.522.363.009	700,66	<0,01
29	1	5	0,8 / 0,2	5.000	0,25	2.524.233.890	701,18	<0,01
30	1	5	0,8 / 0,2	2.880	0,25	2.524.233.889	701,18	<0,01
31	1	3	0,8 / 0,2	5.000	0,25	2.522.363.021	700,66	<0,01
32	1	3	0,8 / 0,2	2.880	0,25	2.522.363.019	700,66	<0,01

6.2.3 Desempenho

Este experimento tem como objetivo avaliar o consumo de energia e o desempenho do sistema em diferentes tratamentos. O experimento considera: 5 fatores ($K = 5$) com 2 níveis ($l = 2$): (i) nível de consistência ($cons$) - 1, 3; (ii) tempo resposta bem-sucedida ($resp$) - 0,249, 0,499ms; (iii) latência do coordenador de leitura (cr) - 7,16, 14,33ms; e (iv) latência do coordenador de escrita (cw) - 7,75, 15,51ms; (v) fator de replicação dos dados (RF) - 3, 5.

A influência da disponibilidade não foi considerada neste experimento. Os valores de $MTTF$ e $MTTR$ para o servidor físico foram definidos como 2.880 horas e 1 hora, respectivamente. Para as réplicas do SGBD, esses valores foram estabelecidos como 1.440 horas para $MTTF$ e 0,5 hora para $MTTR$. O fator $Prop_{RW}$, que representa a proporção das operações de leitura e escrita, foi definido como 0,8 para leitura e 0,2 para escrita. A proporção de operações definida reflete aplicações que lidam com grandes volumes de dados, como redes sociais e e-commerce, que comumente são intensivas em operações de leitura (SEHGAL et al., 2020; LIU; ZHOU, 2021).

O intervalo de tempo entre as solicitações dos clientes foi mantido constante em 12,96 ms para priorizar a análise dos componentes do SGBD. Os níveis de cr e cw foram definidos com base em valores medidos durante a validação do modelo, reduzindo em 50% os valores de cada nível seguinte para representar uma melhor capacidade da infraestrutura. Os níveis selecionados para $cons$ e RF são os mesmos utilizados no experimento que avalia o consumo de energia do sistema (ver Subseção 6.2.1). O fator $resp$ (resposta bem-sucedida) é obtido a partir da máquina cliente. Os fatores cr e cw, representam a comunicação do nó coordenador (réplica que recebe a requisição do cliente) de leitura e escrita, respectivamente.

Tabela 10 – Classificação dos efeitos - Desempenho

Fator | Interação	|Efeito| (ms)
$resp$	665,99
$cons$	16,365
cw	9,589
cr	8,873
$cons \times cw$	2,689
$cons \times cr$	2,501
RF	-0,477

Após executar o projeto de experimentos, os efeitos principais e suas interações foram classificados com base na latência bem-sucedida (ver Tabela 10). O tempo de resposta ($resp$)

teve o maior efeito, pois depende da infraestrutura dos equipamentos. $cons$ teve o segundo maior impacto, devido ao número de confirmações das réplicas necessárias para completar uma operação. Em seguida, aparece o efeito da comunicação dos coordenadores (cw e cr).

O fator de replicação (RF) e suas interações não afetaram significativamente o desempenho do sistema. Pois, embora RF represente a replicação dos dados, o número de replicas definidas nos parâmetros do sistema não apresentou impacto significativo quando comparado aos outros fatores.

Tabela 11 – Tratamentos do consumo de energia e desempenho

Nº	$cons$	RF	$resp$	MTTF (h)	MTTR (h)	Energia (J)	Energia (kWh)	Latencia (s)
1	3	3	0,499	0,01433	0,01551	304.531	0,08459	1,38
2	3	5	0,499	0,01433	0,01551	330.256	0,09174	1,38
3	3	3	0,499	0,00716	0,01551	302.062	0,08391	1,37
4	3	3	0,499	0,01433	0,00753	302.188	0,08395	1,37
5	3	5	0,499	0,00716	0,01551	328.112	0,09114	1,37
6	3	5	0,499	0,01433	0,00753	315.362	0,08760	1,37
7	1	3	0,499	0,01433	0,01551	303.594	0,08433	1,36
8	1	5	0,499	0,01433	0,01551	312.275	0,08674	1,36
9	3	3	0,499	0,00716	0,00753	299.652	0,08324	1,36
10	3	5	0,499	0,00716	0,00753	313.000	0,08694	1,36
11	1	3	0,499	0,00716	0,01551	303.584	0,08433	1,35
12	1	5	0,499	0,00716	0,01551	312.325	0,08676	1,35
13	1	3	0,499	0,01433	0,00753	299.321	0,08314	1,35
14	1	5	0,499	0,01433	0,00753	303.694	0,08436	1,35
15	1	3	0,499	0,00716	0,00753	299.281	0,08313	1,34
16	1	5	0,499	0,00716	0,00753	303.685	0,08435	1,34
17	3	3	0,249	0,01433	0,01551	313.026	0,08695	0,71
18	3	5	0,249	0,01433	0,01551	361.683	0,10047	0,71
19	3	3	0,249	0,00716	0,01551	308.364	0,08566	0,70
20	3	5	0,249	0,00716	0,01551	358.174	0,09949	0,70
21	3	3	0,249	0,01433	0,00753	308.567	0,08572	0,70
22	3	5	0,249	0,01433	0,00753	333.671	0,09269	0,70
23	1	3	0,249	0,01433	0,01551	310.976	0,08638	0,69
24	1	5	0,249	0,01433	0,01551	327.834	0,09106	0,69
25	3	3	0,249	0,00716	0,00753	303.703	0,08436	0,69
26	3	5	0,249	0,00716	0,00753	329.385	0,09149	0,69
27	1	5	0,249	0,00716	0,01551	328.102	0,09114	0,69
28	1	3	0,249	0,00716	0,01551	311.016	0,08639	0,69
29	1	3	0,249	0,01433	0,00753	302.687	0,08408	0,69
30	1	5	0,249	0,01433	0,00753	311.239	0,08645	0,69
31	1	3	0,249	0,00716	0,00753	302.615	0,08406	0,68
32	1	5	0,249	0,00716	0,00753	311.284	0,08647	0,68

Tabela 11 apresenta os resultados considerando a latência em ordem decrescente. O maior valor de latência foi de 1,38 s, associado ao nível mais alto de consistência ($cons = 3$), devido à necessidade de mais confirmações de operações entre réplicas. Os resultados indicam que a redução do nível de consistência de 3 (tratamento Nº 18) para 1 (tratamento Nº 24) resulta em uma diminuição no consumo de energia de aproximadamente 9%, sem comprometer signi-

ficativamente o desempenho do sistema. Este experimento demonstra que a escolha adequada dos níveis de consistência permite equilibrar a latência e o consumo de energia.

6.2.4 Estudo de Caso

Este estudo de caso demonstra a viabilidade do modelo proposto no projeto de um banco de dados como serviço (DBaaS) utilizando um SGBD NoSQL. Esses serviços comumente exigem baixa latência e alta disponibilidade (KHAN et al., 2023). O tamanho do registro foi definido como 1KB para operações de leitura e escrita, representando o modo de capacidade provisionada do DynamoDB (AWS..., 2024). Os níveis para o fator de replicação (RF) são 3 e 5, e, para a consistência ($cons$), os valores adotados são 1 e 3. A Tabela 12 apresenta a proporção de operações de leitura e escrita, assim como os valores de consumo de energia. A Tabela 13 descreve os atrasos de transição. Todos os valores são baseados em (GOMES et al., 2023), e o período de avaliação (T) é definido como $1hora$.

Tabela 12 – Valores de proporção e potência

Fator	Valor
Proporção de leitura ($tSrvRead$)	0,6
Proporção de escrita ($tSrvWrite$)	0,4
Potência - leitura (P_{read})	$32W$
Potência - escrita (P_{write})	$40W$

Tabela 13 – Transições do modelo

Transições	Valor
$tClient$	$3,3ms$
$tCoordRead$	$3,4ms$
$tCoordWrite$	$2,6ms$
$tRespSucRead$ ($tRespFailRead$)	$0,8ms$
$tRespSucWrite$ ($tRespFailWrite$)	$0,8ms$
$tFailure$	$2.880h$
$tRestore$	$0,25h$
$tInFail$	$5.000h$
$tInRes$	$2h$

A Tabela 14 apresenta os resultados, confirmando o comportamento obtido em experimentos anteriores em relação a *cons* e *RF*. No entanto, uma observação importante é o impacto de *cons* na latência devido ao aumento da comunicação entre réplicas. Em relação ao consumo de energia, os resultados também reforçam a influência de *cons*. Por exemplo, com $RF = 3$, o consumo de energia pode aumentar em mais de 40% em um único *host*.

Tabela 14 – Resultados do estudo de caso

RF	*cons*	Latência (ms)	Disponibilidade	Energia (kWh)
3	1	16	$>$ 0,999999	0,0584
3	3	17	0,999979	0,0843
5	1	16	$>$ 0,999999	0,0807
5	3	17	$>$ 0,999999	0,1447

O nível de consistência 1 é uma alternativa melhor para um sistema de rede social, já que a disponibilidade é alta com menos réplicas. No entanto, pode haver inconsistência temporária de dados. A consistência de dados é mais relevante para um site de notícias, pois essa aplicação requer uma reputação melhor. Nesse caso, recomenda-se um nível de consistência mais alto, o que pode resultar em menor disponibilidade. O custo da infraestrutura pode justificar um serviço mais estável com maior disponibilidade, mas o custo de manutenção precisa ser levado em consideração. Os resultados indicam a importância de avaliar o consumo de energia, que é uma preocupação global (MALMODIN, 2023). O modelo proposto é uma ferramenta adicional que permite às organizações avaliar o impacto da consistência eventual no consumo de energia, desempenho e disponibilidade.

6.3 CONSIDERAÇÕES

A consistência eventual é uma característica dos SGBDs NoSQL baseados em quorum que pode influenciar diretamente o consumo de energia, a disponibilidade e o desempenho. A escolha de um nível de consistência apropriado requer a análise cuidadosa dos requisitos do sistema e dos custos de implementação.

Os resultados experimentais apresentados neste capítulo validaram o modelo proposto, demonstrando sua capacidade de estimar os impactos dos níveis de consistência e do fator de replicação no consumo de energia, desempenho e disponibilidade de sistemas NoSQL baseados em quorum. Além disso, os experimentos destacaram que ajustes no quorum de leitura e escrita podem otimizar os recursos disponíveis no sistema, alinhando-os aos requisitos desejados. O estudo de caso também evidenciou que o modelo é aplicável em cenários reais e pode auxiliar projetistas e organizações a encontrar um equilíbrio eficiente entre qualidade do serviço e custo operacional.

7 CONCLUSÃO

Nas últimas décadas, as mudanças na sociedade foram influenciadas pelo avanço tecnológico e pela comunicação mais rápida e acessível. A crescente utilização de redes sociais, dispositivos móveis e Internet das Coisas tem transformado o gerenciamento de dados em todo o mundo. O SGBD NoSQL foi desenvolvido para atender a demandas que os bancos de dados relacionais têm dificuldade em solucionar, como a necessidade de maior disponibilidade, flexibilidade e escalabilidade. Atualmente, os SGBDs NoSQL baseados em quorum são amplamente utilizados por organizações devido à sua eficiência no processamento de grandes volumes de dados.

Apesar de sua relevância, os aspectos de qualidade que dependem da configuração dos SGBDs NoSQL, especialmente em relação ao nível de consistência das operações, ainda são pouco explorados em comparação com outros fatores. A consistência garante que todas as réplicas redundantes mantenham a versão mais atual dos dados, assegurando acesso a informações confiáveis e atualizadas. Essa característica é crucial para proporcionar vantagens competitivas e aumentar a confiança no sistema. Contudo, é necessário considerar os impactos da consistência no consumo de energia, desempenho e disponibilidade. No meio acadêmico e nas organizações, pesquisas destacam a medição do consumo de energia e alguns aspectos relacionados à consistência desses sistemas, enquanto outros estudos enfatizam o desempenho e a disponibilidade associados a esses SGBDs (ARAÚJO et al., 2024; KHAN et al., 2023; FALCãO et al., 2024).

Para melhorar o desempenho e a disponibilidade, os SGBDs NoSQL adotam comumente a consistência eventual, que permite uma inconsistência temporária enquanto os dados são atualizados nas réplicas. Com o tempo, todas as réplicas alcançam a consistência. Nesse contexto, este trabalho propôs um modelo para avaliar o consumo de energia, o desempenho e a disponibilidade de SGBDs NoSQL baseados em consistência quorum. Para validar o modelo, foi gerada uma carga de trabalho com o YCSB, e o consumo de energia foi medido utilizando o *framework Measurement Server* e uma plataforma Arduino. Foram realizadas 1.000 operações de leitura e 1.000 de escrita, com 100 amostras para cada operação, com o objetivo de obter valores médios de consumo de energia e tempo de execução para validação do modelo.

Os resultados evidenciam que os níveis de consistência podem influenciar significativamente o consumo de energia, o desempenho e a disponibilidade. A elevação do nível de consistência

melhora a disponibilidade, mas aumenta o consumo de energia e pode reduzir o desempenho do sistema. O método proposto permite avaliar essas métricas simultaneamente, identificando a configuração mais adequada para equilibrar essas variáveis, conforme os requisitos específicos de cada sistema.

O modelo proposto foi desenvolvido com base em redes de Petri estocásticas (SPN) e validado por meio de experimentos reais e técnicas estatísticas detalhadas ao longo desta dissertação. A próxima seção apresenta as principais contribuições deste trabalho, bem como suas limitações. Por fim, são sugeridas possibilidades para estudos futuros derivados desta pesquisa.

7.1 CONTRIBUIÇÕES

Esta pesquisa apresenta diversas contribuições para o avanço do conhecimento sobre SGBDs NoSQL baseados em quorum, além de destacar a aplicabilidade dessas técnicas em organizações. As principais contribuições são descritas a seguir:

- Método proposto: Foi apresentado um método para criar um modelo que estima o consumo de energia, desempenho e disponibilidade. O trabalho utilizou a técnica de DoE para organizar as comparações e um *benchmark* para gerar a carga de trabalho. O consumo de energia foi medido com o *Measurement Server*, e o modelo foi validado por meio de análise estatística com base nos valores coletados;

- Modelo: Foi proposto um modelo para estimar o consumo de energia das operações de SGBDs NoSQL baseados em quorum, considerando seu perfil de utilização. O modelo permite avaliar o consumo de energia de forma isolada ou em conjunto com o desempenho e a disponibilidade, levando em consideração a capacidade do SGBD e a demanda do cliente;

- Experimentos: Diversos tratamentos foram realizados para avaliar o impacto dos fatores e suas interações em cada métrica do modelo. Esses experimentos destacam a relevância da escolha do nível de consistência e demonstram como diferentes configurações podem se relacionar. Por exemplo, uma maior disponibilidade pode implicar em um maior consumo de energia, devido ao número adicional de réplicas necessárias para confirmar uma operação bem-sucedida. Além disso, um alto nível de consistência pode impactar ne-

gativamente o desempenho, resultando em maior latência e aumento do consumo de energia do sistema.

O método e o modelo propostos nesta dissertação possibilitam a otimização de sistemas de armazenamento NoSQL em larga escala, promovendo melhorias significativas na aquisição e manutenção de recursos. Além disso, esta pesquisa resultou na submissão de um artigo ao *The Journal of Supercomputing*, que atualmente se encontra em processo de avaliação.

7.2 LIMITAÇÕES

Apresentam-se a seguir algumas limitações identificadas durante o desenvolvimento desta pesquisa:

- Consumo de energia: Para aplicar o modelo de consumo de energia, é necessário conhecer a infraestrutura ou medir o consumo médio de uma é replica do SGBD utilizado. Além disso, o modelo se restringe à medição do consumo total de energia da máquina física, sem detalhar o consumo por componente de hardware, devido às limitações do equipamento de metrologia utilizado;
- Escopo: Esta pesquisa propõe um modelo para avaliar o consumo de energia, o desempenho e a disponibilidade em SGBDs NoSQL que adotam consistência baseada em quorum, sem contemplar outras abordagens de consistência;
- Técnica e modelo proposto: Para utilizar o modelo proposto, o arquiteto do sistema deve possuir um conhecimento detalhado do SGBD avaliado e da infraestrutura onde o sistema está implementado. Em cenários como DBaaS, é imprescindível ter acesso às informações sobre a infraestrutura para realizar uma avaliação precisa.

7.3 TRABALHOS FUTUROS

Os resultados deste trabalho incentivam a realização de pesquisas adicionais voltadas à eficiência no consumo de energia. Os itens a seguir são sugeridos para consideração em estudos futuros

- Fatores adicionais: Avaliar outros fatores, como o volume de dados e a taxa de atualização. Esses parâmetros podem fornecer novos *insights* sobre o comportamento dos SGBDs NoSQL em diferentes cargas de trabalho e configurações;
- Extensão para outros tipos de consistência: O modelo proposto generaliza a consistência baseada em quorum, possibilitando sua aplicação a diferentes abordagens de consistência. Outras formas de consistência podem ser estudadas e modeladas com métodos semelhantes aos utilizados nesta pesquisa;
- Novas soluções: Desenvolver sistemas autônomos que ajustem dinamicamente as configurações dos SGBDs com base na qualidade de serviço requerida. Por exemplo, criar um sistema que ajuste automaticamente o nível de consistência para equilibrar latência e disponibilidade conforme a carga de trabalho varia.
- Explorar a aplicação do modelo proposto em outros cenários: Ampliar o escopo de análise considerando diferentes arquiteturas, mecanismos de replicação e cenários de falha, com o objetivo de avaliar sua aplicabilidade em contextos mais variados.

REFERÊNCIAS

AHMED, J.; KARPENKO, A.; TARASYUK, O.; GORBENKO, A.; SHEIKH-AKBARI, A. Consistency issue and related trade-offs in distributed replicated systems and databases: a review. *Radioelectronic and Computer Systems*, n. 2, p. 171–179, 2023.

AHSANULLAH, M.; KIBRIA, B. G.; SHAKIL, M. *Normal and student's t distributions and their applications*. [S.l.]: Springer, 2014. v. 4.

ARAÚJO, C.; JR, M. O.; NOGUEIRA, B.; MACIEL, P.; TAVARES, E. Performability evaluation of nosql-based storage systems. *Journal of Systems and Software*, Elsevier, v. 208, p. 111885, 2024.

ARAÚJO, C. G. Uma abordagem para análise do impacto da consistência de dados no desempenho e disponibilidade em sgbds nosql. Universidade Federal de Pernambuco, 2022.

ARAUJO, C. J. M. *Avaliação e modelagem de desempenho para planejamento de capacidade de sistema de transferência eletrônica de fundos utilizando tráfego em rajada*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2009.

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

AWS, Service, account, and table quotas in Amazon DynamoDB. 2024. Disponível em: <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/>. Acesso em: 21/09/2024.

BANSAL, N.; SACHDEVA, S.; AWASTHI, L. K. Are nosql databases affected by schema? *IETE Journal of Research*, Taylor & Francis, p. 1–22, 2023.

BARON, C. A. et al. Nosql key-value dbs riak and redis. *Database Systems Journal*, Academy of Economic Studies-Bucharest, Romania, v. 6, n. 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.

BORAL, H.; DEWITT, D. J. A methodology for database system performance evaluation. *ACM SIGMOD Record*, ACM New York, NY, USA, v. 14, n. 2, p. 176–185, 1984.

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. *Computer*, IEEE, v. 45, n. 2, p. 23–29, 2012.

BUKH, P. N. D. *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: IEEE. *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. [S.l.], 2016. p. 35–42.

CALLOU, G.; MACIEL, P.; TAVARES, E.; ANDRADE, E.; NOGUEIRA, B.; ARAUJO, C.; CUNHA, P. Energy consumption and execution time estimation of embedded system applications. *Microprocessors and Microsystems*, Elsevier, v. 35, n. 4, p. 426–440, 2011.

CARVALHO, G. H. S. de; COSTA, J. C. W. A. *Modelagem e análise de desempenho de esquemas de alocaçao de recursos em redes móveis celulares*. Tese (Doutorado) — Tese de Doutorado, Instituto de Tecnologia-Programa de Pós-Graduaçao em . . . , 2005.

CASSANDRA. 2024. Disponível em: <https://cassandra.apache.org/doc/latest/>.

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. In: SPRINGER. *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2*. [S.l.], 2019. p. 473–484.

CHOU, Y.-H.; RAGHAVAN, A.; LAHIRI, T. Oracle timesten scaleout: a new scale-out in-memory database architecture for extreme oltp. In: *Proceedings of the international workshop on real-time business intelligence and analytics*. [S.l.: s.n.], 2018. p. 1–4.

CIFERRI, R. R. *Um benchmark voltado a análise de desempenho de sistemas de informações geográficas*. Tese (Doutorado) — [sn], 1995.

COOPER, B. F. *Issues brianfrankcooper/YCSB*. 2024. Disponível em: <https://github.com/brianfrankcooper/YCSB/issues>.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: *Proceedings of the 1st ACM symposium on Cloud computing*. [S.l.: s.n.], 2010. p. 143–154.

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.

DEEPIKA, T.; DHANYA, N. Multi-objective prediction-based optimization of power consumption for cloud data centers. *Arabian Journal for Science and Engineering*, Springer, v. 48, n. 2, p. 1173–1191, 2023.

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.

ERYUREK, E.; GILAD, U.; LAKSHMANAN, V.; KIBUNGUCHY-GRANT, A.; ASHDOWN, J. *Data governance: The definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2021.

EVANGELISTA, C. Performance modelling of nosql dbms. In: UNIVERSITÄT ULM. *Proceedings of the 2020 OMI Seminars (PROMIS 2020)*. [S.l.], 2021. v. 1, p. 6–1.

FALCãO, F.; MOURA, J.; SILVA, G.; ARAUJO, C.; SOUSA, E.; TAVARES, E. Energy consumption and performance evaluation of multi-model nosql dbmss. *Revista de Informática Teórica e Aplicada*, v. 30, n. 2, p. 132–140, May 2024.

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: *Proceedings of the 22nd International Conference on Distributed Computing and Networking*. [S.l.: s.n.], 2021. p. 191–195.

GAUR, N.; JOSHI, P.; SRIVASTAVA, R. Modelling database server sizing for concurrent users using coloured petri-nets. In: IEEE. *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*. [S.l.], 2017. p. 90–94.

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

GERMAN, R.; KELLING, C.; ZIMMERMANN, A.; HOMMEL, G. Timenet: a toolkit for evaluating non-markovian stochastic petri nets. *Performance Evaluation*, Elsevier, v. 24, n. 1-2, p. 69–87, 1995.

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

GOMES, C.; JUNIOR, M. N. de O.; NOGUEIRA, B.; MACIEL, P.; TAVARES, E. Nosql-based storage systems: influence of consistency on performance, availability and energy consumption. *The Journal of Supercomputing*, Springer, v. 79, n. 18, p. 21424–21448, 2023.

GORBENKO, A.; ROMANOVSKY, A.; TARASYUK, O. Interplaying cassandra nosql consistency and performance: A benchmarking approach. In: SPRINGER. *Dependable Computing-EDCC 2020 Workshops: AI4RAILS, DREAMS, DSOGRI, SERENE 2020, Munich, Germany, September 7, 2020, Proceedings 16*. [S.l.], 2020. p. 168–184.

GUO, B.; WU, J.; PU, Y.; ZHANG, J.; YU, J. Energy consumption estimation and profiling for queries in distributed database systems based on a bottom-up comprehensive energy model. *Future Generation Computer Systems*, Elsevier, 2024.

HSU, T.-Y.; KSHEMKALYANI, A. D. Convergent causal consistency for social media posts. In: *Proceedings of the 8th Workshop on Principles and Practice of Consistency for Distributed Data*. [S.l.: s.n.], 2021. p. 1–8.

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. *Application and Theory of Petri Nets and Concurrency: 38th International Conference, PETRI NETS 2017, Zaragoza, Spain, June 25–30, 2017, Proceedings 38*. [S.l.], 2017. p. 78–98.

IIMURA, N.; NISHIKAWA, N.; NAKANO, M.; OGUCHI, M. A proposal of storage power control method with data placement in an environment using many hdds. In: *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*. [S.l.: s.n.], 2015. p. 1–8.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley New York, 1991. v. 1.

KALID, S.; SYED, A.; MOHAMMAD, A.; HALGAMUGE, M. N. 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.

KENITAR, S. B.; ARIOUA, M.; YAHYAOUI, M. A novel approach of latency and energy efficiency analysis of iiot with sql and nosql databases communication. *IEEE Access*, IEEE, v. 11, p. 129247–129257, 2023.

KHAN, W.; KUMAR, T.; ZHANG, C.; RAJ, K.; ROY, A. M.; LUO, B. Sql and nosql database software architecture performance analysis and assessments—a systematic literature review. *Big Data and Cognitive Computing*, MDPI, v. 7, n. 2, p. 97, 2023.

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.

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

KUMAR, M. S. et al. Comparison of nosql database and traditional database-an emphatic analysis. *JOIV: International Journal on Informatics Visualization*, v. 2, n. 2, p. 51–55, 2018.

LEI, H.; LI, C.; ZHOU, K.; ZHU, J.; YAN, K.; XIAO, F.; XIE, M.; WANG, J.; DI, S. X-stor: A cloud-native nosql database service with multi-model support. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 17, n. 12, p. 4025–4037, 2024.

LI, X.-Y.; LIU, Y.; LIN, Y.-H.; XIAO, L.-H.; ZIO, E.; KANG, R. A generalized petri net-based modeling framework for service reliability evaluation and management of cloud data centers. *Reliability Engineering & System Safety*, Elsevier, v. 207, p. 107381, 2021.

LILJA, D. J. *Measuring computer performance: a practitioner's guide*. [S.l.]: Cambridge university press, 2005.

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. R. M. *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. [S.l.]: Chapman and Hall/CRC, 2023.

MAHAJAN, D.; BLAKENEY, C.; ZONG, Z. Improving the energy efficiency of relational and nosql databases via query optimizations. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 22, p. 120–133, 2019.

MALMODIN, J. Just measure it!-electricity consumption measurements of electronic devices and estimates of datacenter and network services for one household. In: IEEE. *2023 International Conference on ICT for Sustainability (ICT4S)*. [S.l.], 2023. p. 35–45.

MARSAN, M. A.; BALBO, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998.

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. *Application and Theory of Petri Nets and Concurrency: 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings 37*. [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.

MEGA2560. 2023. Disponível em: <https://docs.arduino.cc/hardware/mega-2560/>. Acesso em: 2023-11-01.

MEIER, A.; KAUFMANN, M.; MEIER, A.; KAUFMANN, M. Nosql databases. *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*, Springer, p. 201–218, 2019.

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

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

MONTGOMERY, D. C.; RUNGER, G. C. *Applied statistics and probability for engineers*. [S.l.]: John wiley & sons, 2020.

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

NILSSON, J. W.; RIEDEL, S. A. *Electric circuits*. [S.l.]: Pearson Upper Saddle River, NJ, 2015.

NOUDOUST, N. N. S.; ADABI, S.; REZAEE, A. A quorum-based data consistency approach for non-relational database. *Cluster Computing*, Springer, v. 25, n. 2, p. 1515–1540, 2022.

PALANISAMY, S.; SUVITHAVANI, P. A survey on rdbms and nosql databases mysql vs mongodb. In: IEEE. *2020 International Conference on Computer Communication and Informatics (ICCCI)*. [S.l.], 2020. p. 1–7.

RAVAT, F.; SONG, J.; TESTE, O.; TROJAHN, C. Efficient querying of multidimensional rdf data with aggregates: Comparing nosql, rdf and relational data stores. *International Journal of Information Management*, Elsevier, v. 54, p. 102089, 2020.

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: IEEE. *2019 IEEE International Systems Conference (SysCon)*. [S.l.], 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.

SEHGAL, N. K.; BHATT, P. C. P.; ACKEN, J. M.; SEHGAL, N. K.; BHATT, P. C. P.; ACKEN, J. M. Cloud workload characterization. *Cloud Computing with Security: Concepts and Practices*, Springer, p. 75–97, 2020.

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.

SHI, X.; PRUETT, S.; DOHERTY, K.; HAN, J.; PETROV, D.; CARRIG, J.; HUGG, J.; BRONSON, N. {FlightTracker}: Consistency across {Read-Optimized} online stores at facebook. In: *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. [S.l.: s.n.], 2020. p. 407–423.

SOBOL, V. Enhanced wars model proposal for advancing reasoning consistency based on probabilistically bounded staleness. In: *ICTERI*. [S.l.: s.n.], 2021. p. 276–285.

STOREY, V. C.; SONG, I.-Y. Big data technologies and management: What conceptual modeling can do. *Data & Knowledge Engineering*, Elsevier, v. 108, p. 50–67, 2017.

TAVARES, P. E. *Measurement server v0.01*. 2023. Disponível em: <https://sites.google.com/site/eagtav/measurement-server>. Acesso em: 01/11/2024.

WAHID, A.; KASHYAP, K. Cassandra—a distributed database system: An overview. *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 1*, Springer, p. 519–526, 2019.

Worldwide, IDC. *Global DataSphere Forecast, 2024–2028: AI Everywhere, But Upsurge in Data Will Take Time*. 2024.

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.

ZIMMERMANN, A. Modelling and performance evaluation with timenet 4.4. In: SPRINGER. *Quantitative Evaluation of Systems: 14th International Conference, QEST 2017, Berlin, Germany, September 5-7, 2017, Proceedings 14*. [S.l.], 2017. p. 300–303.

APÊNDICE A – LOG DE EXECUÇÃO MEASUREMENT SERVER

```
1  Evaluation technique: bootstrap
   Sample size: 10
3  Significance level: 0.05
   Batch or bootstrap sample size: 40
5  Device: arduino
   ----------
7  Server initiated using port 12345
   Client connected: 10.0.0.163
9  Obtaining sample 1 - Energy(J): 19.09967    Time(s):    3.95499
   Obtaining sample 2 - Energy(J): 20.34978    Time(s):    4.18999
11 Obtaining sample 3 - Energy(J): 20.07942    Time(s):    4.04499
   Obtaining sample 4 - Energy(J): 20.09634    Time(s):    4.06399
13 Obtaining sample 5 - Energy(J): 20.38299    Time(s):    4.16599
   Obtaining sample 6 - Energy(J): 20.42420    Time(s):    4.02699
15 Obtaining sample 7 - Energy(J): 19.91024    Time(s):    4.05999
   Obtaining sample 8 - Energy(J): 20.06814    Time(s):    4.09999
17 Obtaining sample 9 - Energy(J): 19.93460        Time(s):    4.09299
   Obtaining sample 10 - Energy(J):    20.57604    Time(s):    4.19899
19 ----------
   Result energy (J):  20.06786    [19.80069,20.33502]
21 Result time (s):    4.08905 [4.03969,4.13839]
   Server Closed
```

Listing A.1 – Exemplo do log de execução da carga de trabalho no Cliente.

APÊNDICE B – CÓDIGO BENCHMARK PYTHON

O código Python sincroniza o YCSB com o *Measurement Server* e executa o *benchmark* conforme a quantidade de amostras definidas no *framework* de medição.

```
1
import socket
3 import time
import subprocess
5
HOST = "127.0.0.1"  # The server's hostname or IP address
7 PORT = 12345  # The port used by the server

9 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
11     x = s.makefile("rb")
    delayStartStop = 10
13     resp = x.readline()
    #print(s.recv(1024).decode('ascii'))
15     print(resp.decode('ascii'))
    count = 0
17     while True:
        s.sendall("#start\n".encode('ascii'))
19
        comand = "/YCSB/bin/ycsb.bat run cassandra-cql -s -P workloads/workloada -p
            cassandra.hosts=10.0.0.149 -p cassandra.port=9042 -p operationcount=1000 -
            p cassandra.protocol=HTTP_JSON > cassandradbRead_1000_"
21
        count = count + 1
23         countPlus = str(count) + ".txt"
        comand = comand + countPlus
25         process = subprocess.Popen(comand, shell=True, stdout=subprocess.PIPE)
        process.wait()
27         print("Process return code " + str(process.returncode))
        s.sendall("#stop\n".encode('ascii'))
29         #resp = s.recv(1024)
        resp = x.readline()
31         print(resp.decode('ascii'), end ="")
        resp = x.readline()
33         print(resp.decode('ascii'))

35         if resp.decode('ascii').find("#finished") > -1:
            break
37     resp = x.readline()
    print(resp.decode('ascii'))
39     s.close()
```

Listing B.1 – Exemplo do *script* de execução da carga de trabalho.

APÊNDICE C – APRESENTAÇÃO DAS AMOSTRAS OBTIDAS

Apresentam-se 100 amostras de consumo de energia do sistema, resultantes da execução de cada operação (leitura e escrita).

Tabela C.1 – Valores médios das operações de leitura

Métricas	Média	Intervalo
Energia (J)	571,8447	[568,7755; 574,9139]
Tempo (s)	13,5791	[13,5718; 13,5864]

Potência média de leitura = 42,1121 J/s

Tabela C.2 – **Amostras Coletadas: 1.000 Operações de Leitura**

Nº da amostra	Energia (J)	Tempo (s)
1	592,53732	13,62699
2	590,76157	13,61599
3	580,72165	13,52999
4	572,85462	13,56099
5	560,71577	13,54499
6	598,34725	13,60299
7	618,43279	13,60699
8	579,23579	13,61799
9	665,50527	13,58599
10	595,62273	13,64999
11	596,98810	13,63299
12	599,38689	13,52199
13	594,68121	13,58099
14	579,42848	13,59899
15	576,51557	13,54299
16	579,02737	13,59699

17	567,82533	13,51499
18	569,94238	13,58899
19	565,20196	13,56399
20	572,22857	13,57999
21	572,38684	13,59399
22	568,94570	13,58799
23	567,93062	13,58599
24	565,87383	13,62899
25	567,63523	13,55099
26	567,90262	13,63299
27	573,96259	13,60199
28	573,55489	13,53299
29	570,25197	13,57799
30	567,58386	13,54899
31	574,59280	13,64299
32	565,72457	13,54499
33	571,58331	13,53999
34	565,10489	13,54299
35	571,55844	13,63499
36	568,15049	13,53199
37	566,30693	13,58699
38	567,27299	13,58599
39	572,29355	13,58599
40	566,26789	13,56699
41	567,91068	13,54599
42	564,09068	13,55499
43	563,58099	13,55999

44	561,20076	13,56799
45	563,96361	13,53599
46	567,36162	13,56599
47	559,98297	13,49399
48	569,84785	13,60299
49	564,10206	13,62899
50	565,53979	13,58299
51	567,87190	13,57199
52	568,26060	13,56099
53	568,72529	13,63199
54	563,43045	13,55899
55	564,97570	13,56699
56	563,83513	13,57899
57	568,99714	13,56899
58	563,97992	13,53899
59	566,88831	13,54299
60	566,88882	13,57799
61	566,94791	13,59599
62	564,38571	13,57099
63	567,88130	13,59699
64	573,98508	13,61299
65	568,72371	13,53299
66	567,78773	13,59499
67	566,20534	13,60199
68	563,51897	13,58699
69	564,16033	13,55199
70	570,54907	13,62299

71	562,67739	13,50699
72	564,76743	13,59399
73	566,93636	13,57999
74	568,61489	13,63599
75	561,49620	13,59099
76	564,79720	13,55699
77	567,70071	13,57299
78	572,00988	13,62999
79	570,20462	13,61199
80	575,72389	13,57699
81	563,80497	13,54599
82	564,39211	13,58499
83	571,17208	13,57099
84	565,49071	13,48299
85	564,13984	13,59799
86	565,67611	13,57699
87	573,04591	13,60499
88	561,60890	13,56299
89	567,07405	13,64799
90	571,45758	13,63699
91	566,01854	13,53399
92	563,23606	13,47899
93	565,33629	13,61299
94	573,45469	13,63199
95	563,70427	13,54099
96	566,59971	13,54499
97	565,12760	13,57299

98	566,34006	13,54899
99	575,40714	13,59399
100	568,35566	13,49899

Tabela C.3 – Valores médios das operações de escrita

Métricas	Média	Intervalo
Energia (J)	550,7227	[549,3761; 552,0693]
Tempo (s)	13,5665	[13,5557; 13,5773]

Potência média de escrita = 40,5943 J/s

Tabela C.4 – **Amostras Coletadas: 1.000 Operações de Escrita**

Nº da amostra	Energia (J)	Tempo (s)
1	567,68183	13,59199
2	554,75481	13,59299
3	564,48485	13,53199
4	554,85180	13,54299
5	571,61213	13,62299
6	574,84457	13,60699
7	558,90302	13,61799
8	554,57682	13,59399
9	545,24832	13,53799
10	553,79138	13,59199
11	559,55579	13,52899
12	558,47610	13,58699
13	545,42135	13,50499
14	557,01598	13,56999
15	551,54530	13,60399
16	555,70634	13,53699

17	549,91516	13,60899
18	553,96009	13,56699
19	555,46932	13,54399
20	552,18570	13,53099
21	551,69756	13,54099
22	551,14802	13,56899
23	559,50273	13,54199
24	545,44777	13,48399
25	545,58196	13,54499
26	550,98763	13,60899
27	544,18472	13,51099
28	550,79420	13,56699
29	549,74489	13,57399
30	549,44243	13,50799
31	548,72629	13,54699
32	548,57939	13,56699
33	547,65350	13,55899
34	552,87124	13,49599
35	553,31825	13,58099
36	543,92379	13,44899
37	553,60189	13,61199
38	554,92452	13,54799
39	549,88817	13,57099
40	550,36293	13,57399
41	550,25684	13,57699
42	545,63494	13,54699
43	545,25358	13,47899

44	547,65580	13,56499
45	548,27907	13,62399
46	548,46454	13,55599
47	553,46127	13,59199
48	546,35612	13,54799
49	548,91666	13,57999
50	548,22408	13,55999
51	547,56333	13,55899
52	550,21996	13,56199
53	550,10432	13,53699
54	545,29879	13,51799
55	547,59425	13,55599
56	551,12297	13,54199
57	547,95344	13,61799
58	557,30581	13,59699
59	552,10044	13,60099
60	550,32345	13,55499
61	546,92965	13,55699
62	545,91093	13,51199
63	552,39670	13,50499
64	541,03326	13,53899
65	546,60865	13,56799
66	547,89010	13,49299
67	550,16293	13,61199
68	552,96577	13,62999
69	550,49010	13,56299
70	549,02199	13,63099

71	545,61482	13,53799
72	540,98919	13,57899
73	549,28819	13,52199
74	544,91578	13,55299
75	554,50354	13,56799
76	548,00256	13,52099
77	546,75649	13,51199
78	548,23660	13,52199
79	548,30619	13,55399
80	550,50724	13,59399
81	545,17045	13,59799
82	546,61535	13,52399
83	539,98092	13,57699
84	547,86193	13,49899
85	555,93593	13,69299
86	548,80367	13,59799
87	547,16728	13,58599
88	548,10401	13,58799
89	546,48610	13,51499
90	549,49529	13,57299
91	544,87983	13,53099
92	548,91694	13,58799
93	544,58859	13,48599
94	547,35407	13,60999
95	549,95938	13,56499
96	543,58116	13,51599
97	538,11027	13,49499

98	550,84203	13,57499
99	554,96449	13,62899
100	546,90741	13,51399

APÊNDICE D – EXEMPLO DE FLUXO OPERAÇÃO DE LEITURA BEM-SUCEDIDA

A seguir, apresenta-se um exemplo de execução do modelo proposto neste estudo. Neste exemplo, é considerada uma requisição bem-sucedida para a operação de leitura.

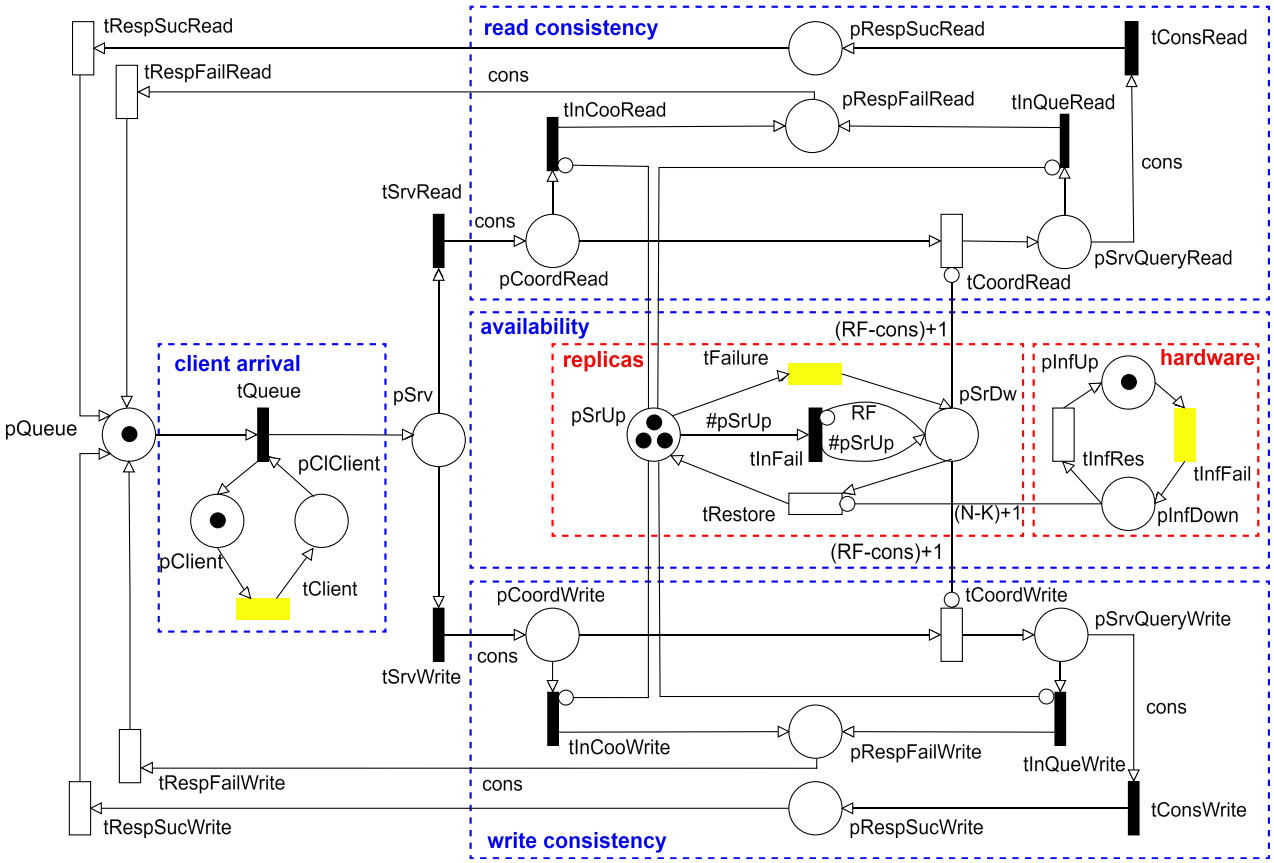


Figura 26 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 01.

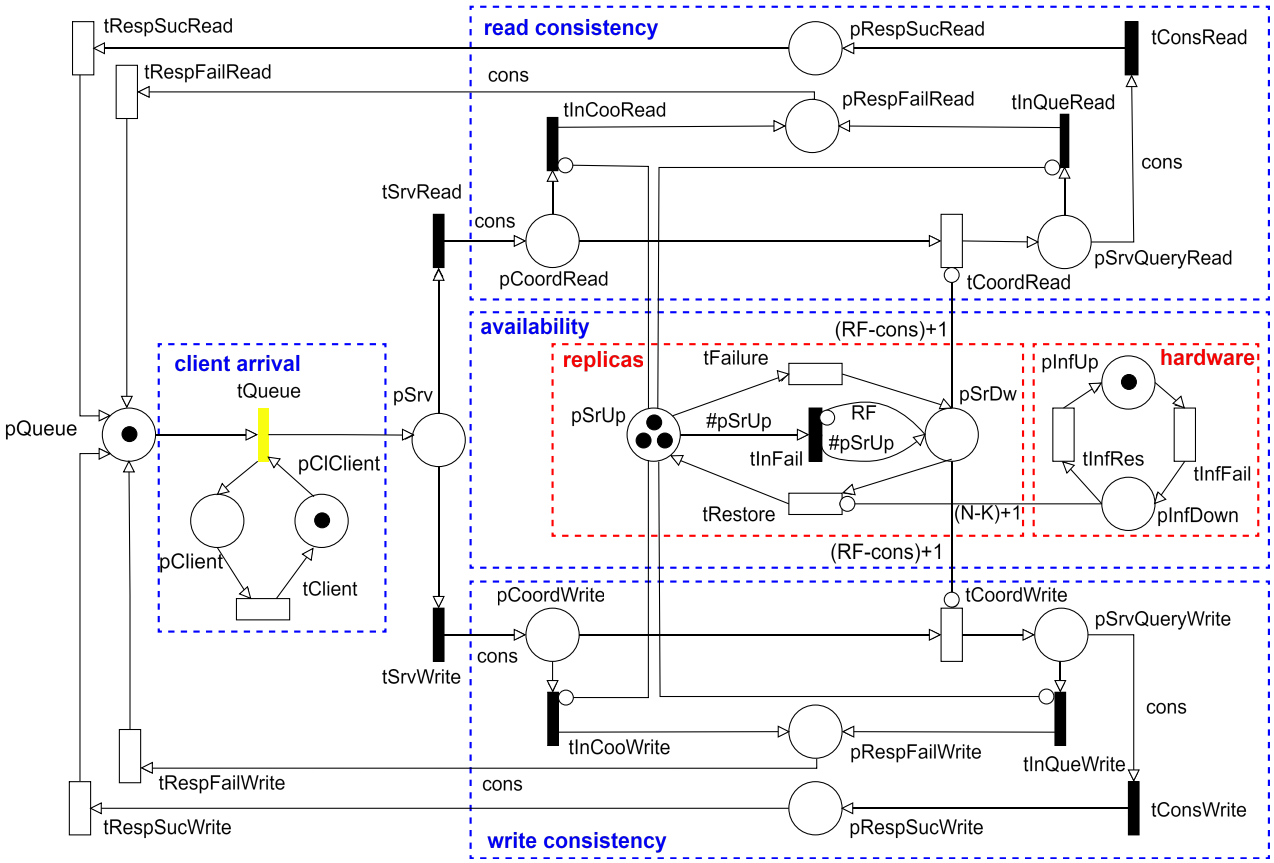


Figura 27 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 02.

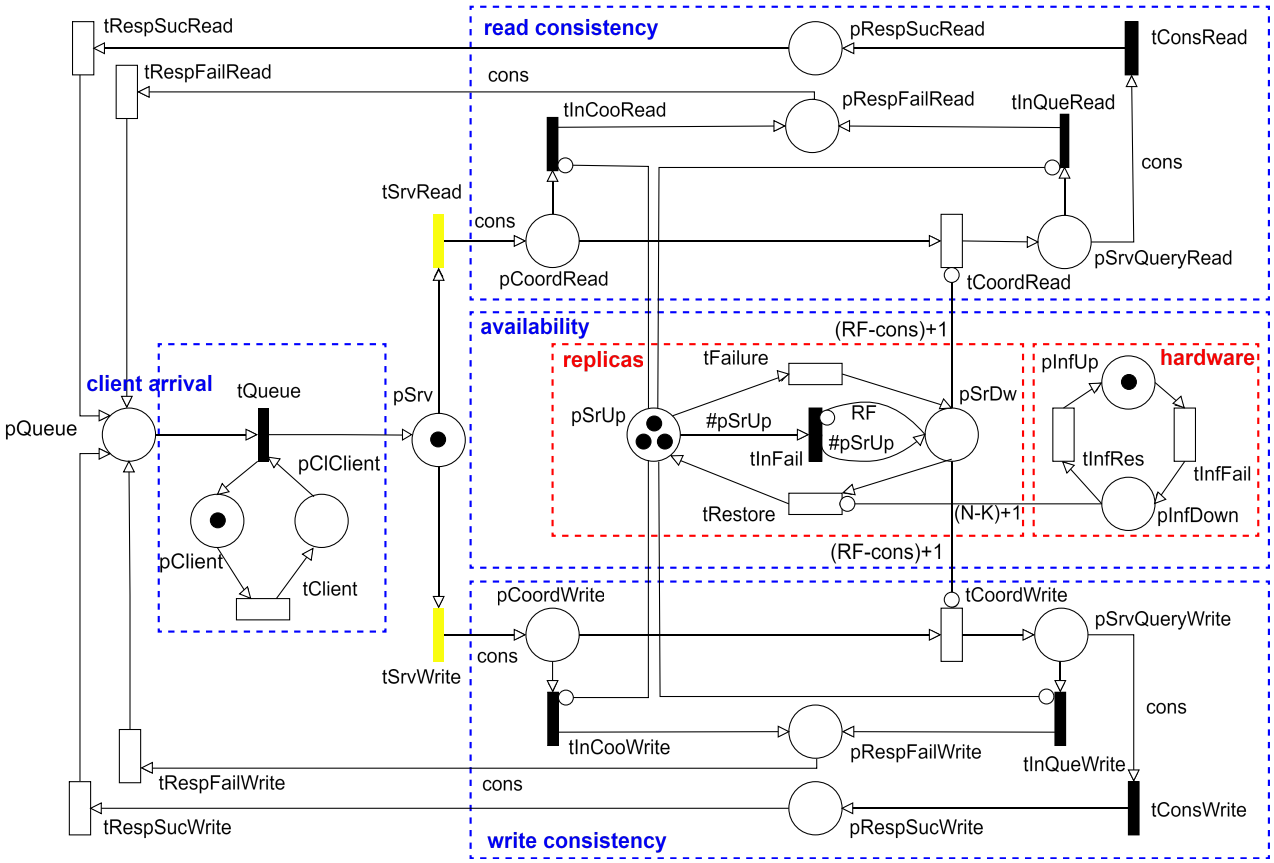


Figura 28 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 03.

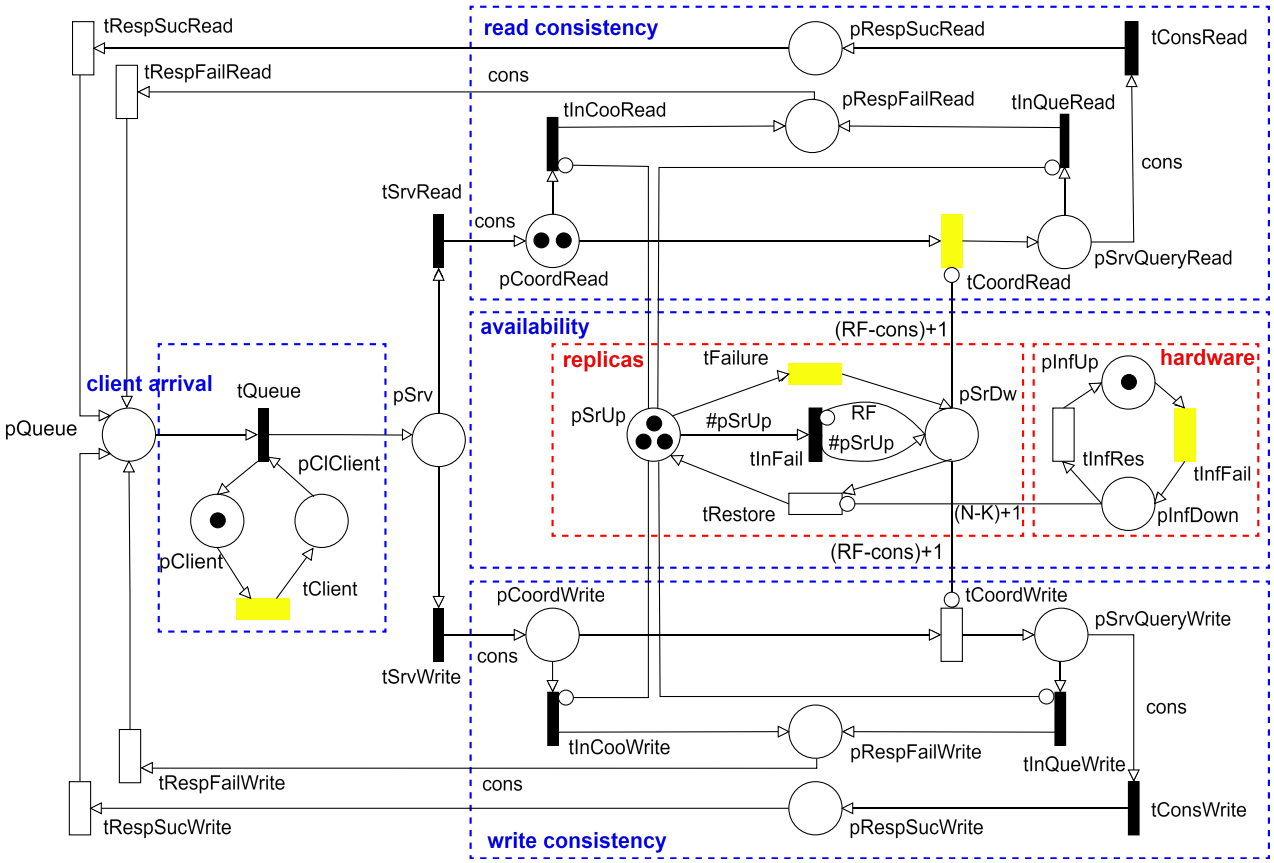


Figura 29 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 04.

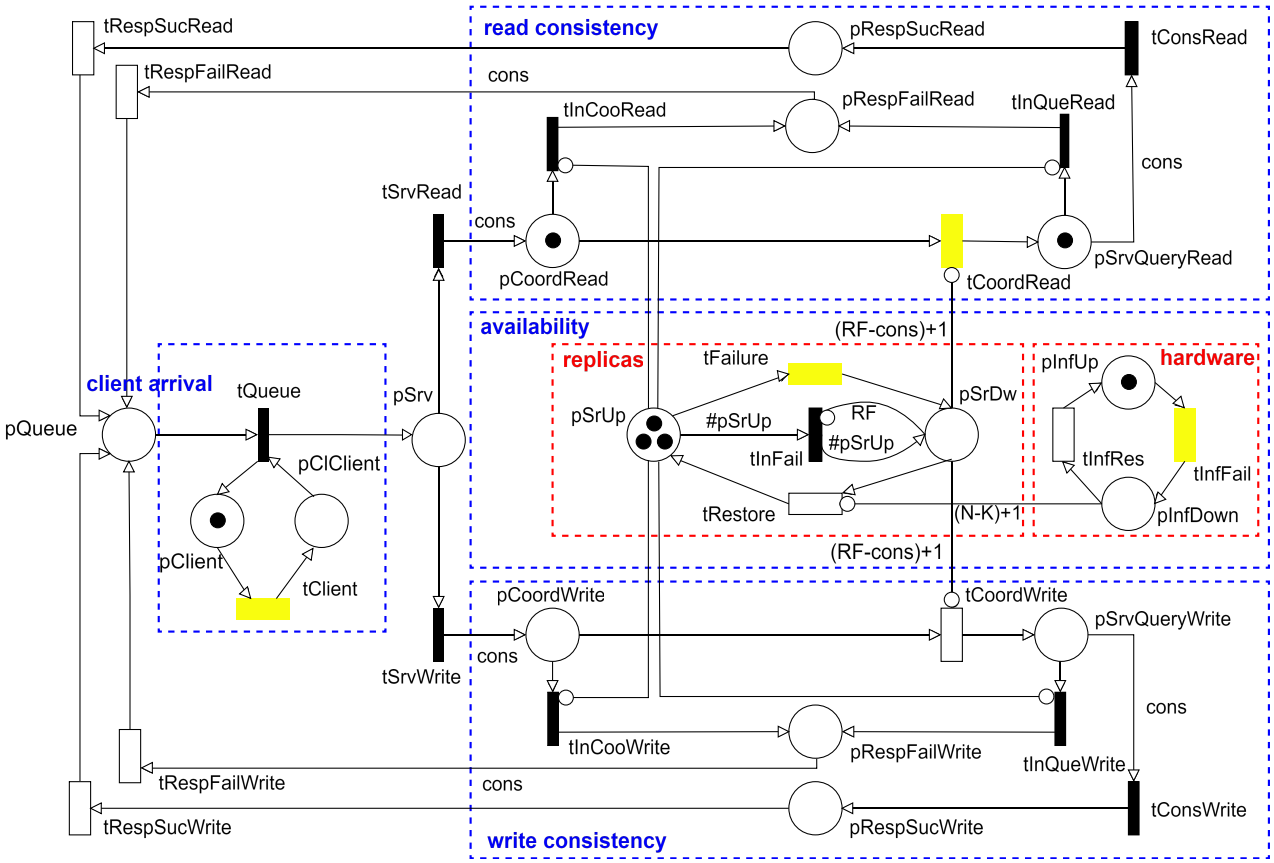


Figura 30 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 05.

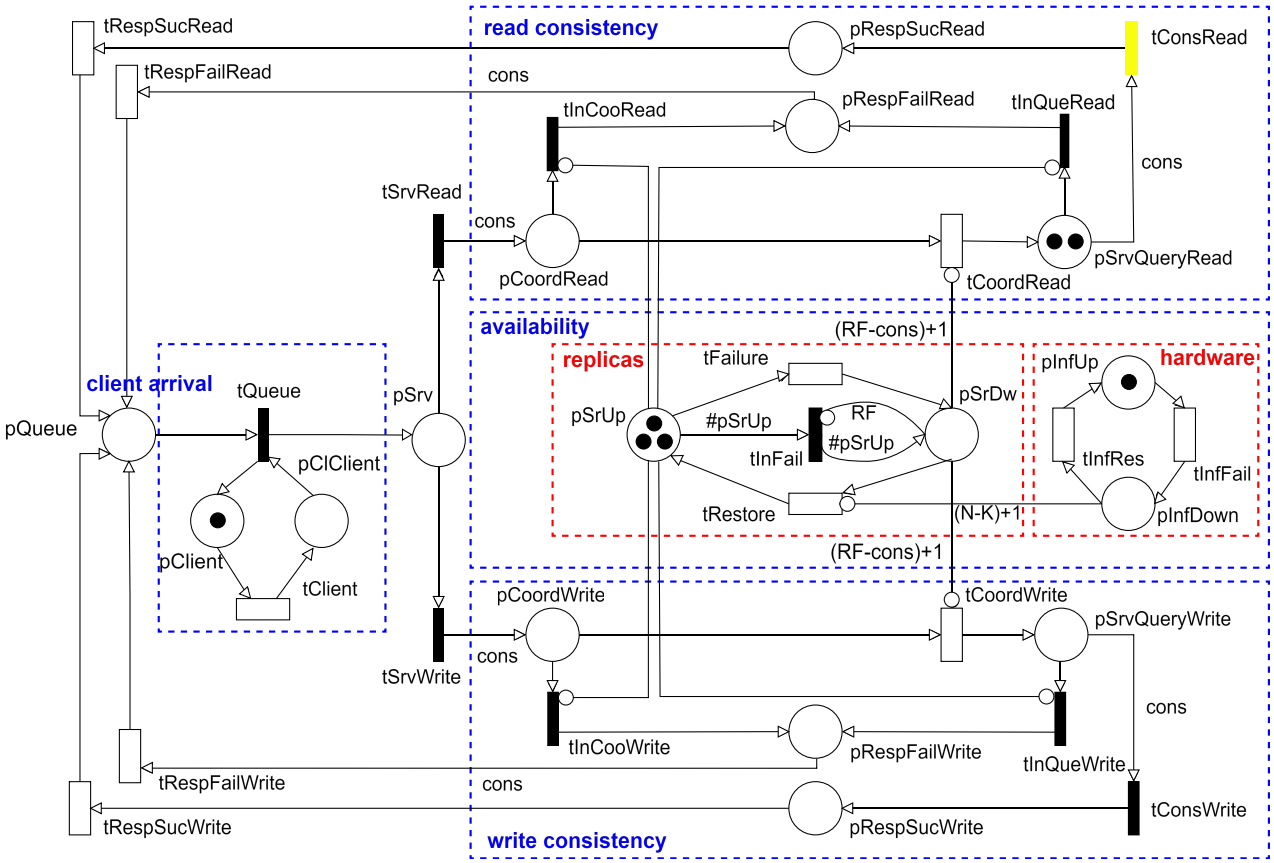


Figura 31 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 06.

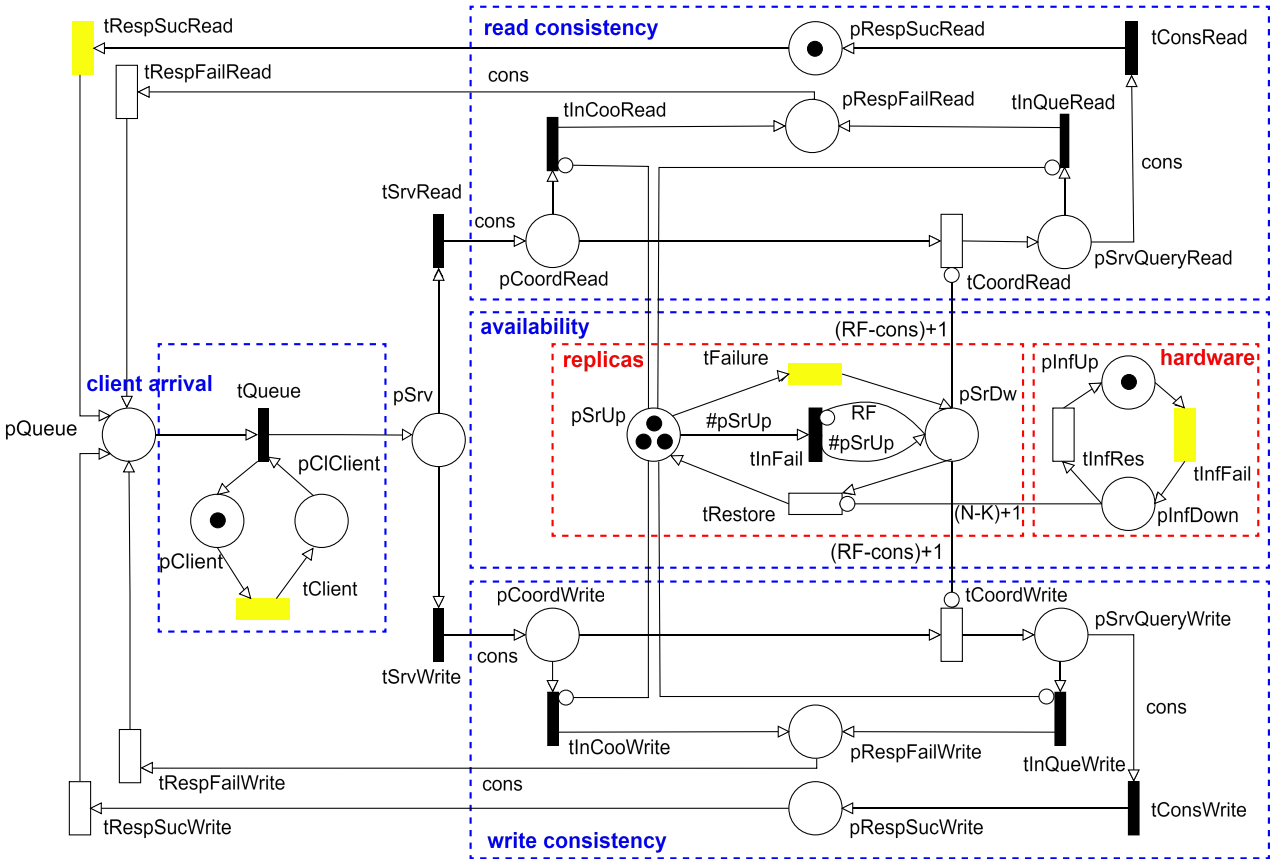


Figura 32 – (Apêndice D) Fluxo de operação de leitura bem-sucedida do Modelo - 07.