Pós-Graduação em Ciência da Computação

Patrícia Dayana de Araújo Souza

Planejamento de infraestrutura computacional para SGBDs NoSQL

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Patrícia Dayana de Araújo Souza

Planejamento de infraestrutura computacional para SGBDs NoSQL

Dissertação de Mestrado apresentada 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: Modelagem e Avaliação Desempenho de Sistemas Comunicantes
Orientador: Prof. Dr. Eduardo Antônio Guimarães Tavares

Recife
2019

Catalogação na fonte
Bibliotecária Mariana de Souza Alves CRB4-2106

S729p Souza, Patrícia Dayana de Araújo
Planejamento de infraestrutura computacional para SGBDs NoSQL – 2019.
91f.: il., fig., tab.

Orientador: Eduardo Antônio Guimarães Tavares.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da computação. Recife, 2019.
Inclui referências.

1. Modelagem e Avaliação Desempenho de Sistemas Comunicantes. 2. NoSQL. 3. Comunicação. 4. Sobrecarga. I. Tavares, Eduardo Antônio Guimarães (orientador). II. Título.

004.029 CDD (22. ed.) UFPE-MEI 2019-154

Patrícia Dayana de Araújo Souza

"**Planejamento de infraestrutura computacional para SGBDs NoSQL**"

Dissertação de Mestrado apresentada 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.

Aprovado em: 30/08/2019.

Orientador: Prof. Dr. Eduardo Antônio Guimarães Tavares

BANCA EXAMINADORA

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

Prof. Dr. Victor Antônio Ribeiro De Lira Cavalcanti
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco / IFPE

Decido este trabalho a minha família (Mãe, pai e irmão), a todos que de alguma forma se inspiram em mim e que me encoraja a enfrentar os desafios (afilhados e primos), e a minha namorada, que foram meu porto seguro perante as dificuldades durante este percurso.

Essa conquista não é só minha, também é de vocês!

AGRADECIMENTOS

Sou imensamente agradecida a Deus por todas as oportunidades de aprendizado intelectual e moral que tive ao longo desta pesquisa. Sou grata por todas as pessoas que estiveram comigo neste caminho. Um caminho recheado de desafios, alguns deles proporcionaram o exercício do autoconhecimento, outros indicaram a necessidade de desenvolver novas habilidades intelectuais. A paciência e a perseverança foram atributos essenciais nesta etapa da minha vida.

A minha família foi meu motor nessa jornada, por isso quero agradecer a todos da minha família, que sempre estiveram comigo. Em especial a minha mãe (Lucimar), ao pai (Nonato), ao meu irmão (Danilo), a minha prima (Fernanda) e a minha companheira de todas as horas (Silvana), os quais amo incondicionalmente.

Agradeço, também, a todos os meus professores que tive até aqui, em especial, agradeço imensamente aos professores: Kerllon (meu orientador da graduação), Aratã (que não tive a oportunidade de ser sua aluna, mas que o conhecimento e ajuda anteriores, nunca serão esquecidos), Vigno (professor da graduação e um grande conselheiro que levarei para vida toda) e ao meu orientador do mestrado, Eduardo Tavares, que me ajudou e teve muita paciência nesse meu percurso.

Não posso deixar de agradecer aos meus colegas de mestrado (Matheus, Thays, Artur, Carlos, Breno, Paulinho, Amanda, Bruna, Alex,Edi, Demis e Guto). Obrigada pela a paciência e apoio nos momentos difíceis que passei enquanto encarava mais esse desafio acadêmico.

Agradeço toda a atenção e a disponibilidade do Sr. Rivaldo (proprietário da casa onde morei durante 2 anos e meio) e de sua família. Também serei eternamente grata a Dona Da Paz, que me recebeu de braços aberto quando cheguei em Recife para vivenciar essa importante experiência, que foi o mestrado.

"Hakuna Matata!"(MINKOFF; ALLERS, 1994).

RESUMO

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs)NoSQL vêm sendo, constantemente, objeto de estudo por serem os mais adequados para trabalhar com grandes volumes de dados e requisições, atendendo, assim, a demandas da Internet das Coisas (IoT) e a aplicações e serviços que integram o *Big Data* (Facebook, Google e Amazon), principalmente por sua capacidade de crescer e se adaptar à carga recebida. Contudo, esses SGBDs são sistemas distribuídos, dessa forma, a comunicação pode ser um gargalo de desempenho. Para garantir disponibilidade, consistência e, em alguns casos, distribuir o desempenho, os SGBDs NoSQL fazem uso, especialmente, do processo de replicação de dados. Embora a replicação seja importante, esse mecanismo pode ocasionar uma sobrecarga da rede, em virtude dos muitos acessos ao nó principal ou por concorrência de outras aplicações na largura de banda disponível, fazendo com que a propagação das réplicas seja interrompida ou que ocorram falhas de consistência na leitura dos nós secundários (como resultado de um *failover* automatizado). Nesse sentido, e tendo em vista a popularização dos SGBDs NoSQL, a ausência de trabalhos que avaliem a comunicação envolvendo esses sistemas e a modelagem focada no desempenho - que é útil para tomada decisões em análises de sistemas e prover uma economia de tempo, dinheiro e/ou trabalho experimental - esta pesquisa propõe modelos baseados em Redes de Petri Estocásticas e uma análise experimental, com o objetivo de avaliar o desempenho de *cluster* NoSQL sob um *link* de rede para contribuir no planejamento da infraestrutura computacional para SGBDs NoSQL, principalmente no tocante à utilização da rede de comunicação. Acreditando ser um diferencial, embora com uma abstração simplista do paradigma SDN, especificamente, a implementação de técnicas de QoS com *OpenFlow* para limitar ou priorizar o tráfego de um *cluster* de banco de dados, ressaltamos que os modelos GSPN gerados neste trabalho, bem como a análise de alguns estudos auxiliam no desempenho da comunicação dos SGBDs NoSQL, além da quantidade de requisições processadas, da taxa de chegada da carga recebida, da replicação dos dados, do tráfego concorrente na largura de banda, do nível de consistência configurada para o banco e da quantidade de nós envolvidos na replicação.

Palavras-chaves: NoSQL. Comunicação. Sobrecarga. Modelagem. Desempenho. Planejamento.

ABSTRACT

The NoSQL Data Base Management Systems (DBMSs) are constantly being studied because they are best suited to work with large volumes of data and requests, thus attending the demands of the Internet of Things (IoT) and applications and services that integrate Big Data (Facebook, Google, and Amazon), primarily for its ability to grow and adapt to the load it receives. However, these DBMSs are distributed systems, so communication can be a performance bottleneck. To ensure availability, consistency and, in some cases, to distribute performance, NoSQL DBMSs makes, especially, use of the data replication process. Although replication is important, this mechanism can cause network overhead due to too many accesses to the head node or competition from other applications over available bandwidth, causing propagation of replicas to stop or consistency failures to read secondary nodes (as a result of an automated failover). In this regarding, and in view of the popularization of NoSQL DBMSs, the lack of work evaluating communication involving these systems and performance-focused modeling - which is useful for decision making in system analysis and provide a saving of time, money and/or experimental work - this research proposes models based on Stochastic Petri Nets and an experimental analysis, aiming to evaluate the performance of NoSQL cluster under a link network to contribute to the computational infrastructure planning for NoSQL DBMSs, especially as to the use of the communication network. Believing to be a differentiator, albeit with a simplistic abstraction of the SDN paradigm, specifically, the implementation of QoS techniques with OpenFlow to limit or prioritize traffic from a database cluster, we emphasize that the GSPN models generated in this work, as well as the analysis of some studies, help in the communication performance of the NoSQL DBMSs, besides the number of requests processed, the rate incoming load, data replication, concurrent bandwidth traffic, the consistency level configured for the bank, and the number of nodes involved in replication.

Keywords: NoSQL. Communication. Overhead. Modeling. Performance. Planning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	Application Programming Interface (Interface de Programação de Aplicações)
BASE	Basicamente Disponível, Estado flexível e Consistência eventual
BJON	Binary JSON
C-DPI	Controller-Data Plane Interface
CAP	Consistência, Disponibilidade e Tolerância a particionamentos
CPS	Cyber-physical Systems (Sistemas Ciberfísicos)
CPU	Central Processing Unit (Unidade Central de Processamento)
CRUD	Criação, Consulta, Atualização e Destruição de dados
DaaS	Database as a Service (Banco de Dados como Serviço)
DBMSs	Data Base Management Systems
DFS	Distributed File System (Sistemas de arquivos distribuídos)
DoE	Design de Experimentos (Planejamento de Experimentos)
ECMP	Equal-cost multi-path routing (Roteamento de vários caminhos de custo igual)
GSPN	Generalized Stochastic Petri Nets (Redes de Petri Estocásticas Generalizadas)
HD	Hard Disk
HDD	Hard Disk Drive
IoT	Internet of Things (Internet das Coisas)
JSON	JavaScript Object Notation
LAN	Local area network (Rede de área local)
MAC	Media Access Control (Controle de Acesso ao Meio)
NoSQL	Not Only SQL (Não apenas SQL)
ONDMs	Object-NoSQL Database Mappers
ONF	Open Networking Foundation
PN	Petri Nets (Redes de Petri)
QoS	Quality of Service (Qualidade de Serviço)
RAM	Random Access Memory (Memória de Acesso Aleatório)

SDN	Software-Defined Network (Rede Definida por Software)
SGBDRs	Sistemas de Gerenciamento de Banco de Dados Relacionais
SGBDs	Sistemas de Gerenciamento de Banco de Dados
SLA	Service Level Agreement (Acordo de Nível de Serviço)
SLO	Service-level Objective
SPN	Stochastic Petri Nets (Redes de Petri Estocásticas)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
SRN	Stochastic Reward Net (Redes de recompensa estocásticas)
SSD	Solid State Drive
TCP/IP	Transmission Control Protocol/Internet Protocol (Protocolo de Controle de Transmissão/Protocolo de Internet)
TLS	Transport Layer Security (Segurança da Camada de Transporte)
TRMG	Transport Modeling Research Group
Vm	Virtual Machine (Máquina Virtual)
WAN	Wide Área Network (Rede de longa distância)
XML	Extensible Markup Language (Linguagem de Marcação Extensível)
YCSB	Yahoo! Cloud Serving Benchmark

LISTA DE SÍMBOLOS

$\in$	Pertence
$\subseteq$	Esta contido
$\cap$	União
∞	Infinito
$\geq$	Maior ou Igual
θ	Teta
μs	Microsegundo
$\prod$	Pi
σ	Sigma

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das aplicações *web*, emergindo junto ao *Big Data* (Facebook, Google, Amazon), e o desenvolvimento da Internet das Coisas (IoT), aliado à nova forma de interação (dinâmica e intuitiva) com o usuário e à necessidade de um alto grau de disponibilidade, além da carência por escalabilidade sob demanda, surgiram os sistemas NoSQL [1]. Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) NoSQL possuem uma expansibilidade dinâmica, isto é, não seguem um esquema rígido, como, por exemplo, o modelo ACID, adotado pelos bancos de dados relacionais, e sim um esquema mais flexível, como o modelo BASE - "*Basically Available*, Soft State, *Eventual consistency*"-, que determina o funcionamento do banco de dados em estar, basicamente, disponível, mas nem sempre sendo, necessariamente, consistente, porém se tornando consistente em algum momento. Esses bancos, por vezes, enfatizam a disponibilidade ou a tolerância a falhas de rede, sacrificando a consistência dos dados, como ocorre, por exemplo, com o banco Cassandra; e, em outros casos, buscam um melhor desempenho e tolerância a falhas de rede, considerando uma consistência eventual dos dados, como é o caso do MongoDB. Assim, para esses sistemas, definir o que é prioridade, é um *trade-off* fundamental que afeta qualquer sistema distribuído, além do fato de que a comunicação pode ser um gargalo de desempenho.

Tal *trade-off* dos sistemas distribuídos é conhecido como teorema CAP ou teorema de Brewer (GILBERT; LYNCH, 2002). De acordo com esse teorema, um sistema distribuído é capaz de fornecer apenas duas entre as três propriedades mais importantes: consistência de dados, disponibilidade do sistema e tolerância a partições de rede. Nessa perspectiva, há uma série de pesquisas em torno desse *trade-off*, no entanto, ainda não foi dada a devida atenção ao meio (rede) pelo qual o sistema funciona ou como o desempenho da comunicação desses sistemas em rede pode interferir na garantia dessas propriedades.

O principal mecanismo dos SGBDs NoSQL para assegurar as propriedades do teorema CAP é a replicação. Esse mecanismo é nativo nos sistemas NoSQL, o que pode significar um aumento no desempenho das operações de leitura ou escrita a partir das réplicas e uma disponibilidade além da capacidade de *failover* do sistema, para que, no caso de um nó original falhar, os dados fiquem salvos e acessíveis em outros nós, como se fossem *backups*. Todavia, o efeito colateral desse processo de replicação é a sobrecarga adicional de tráfego de rede, que vem se tornando, em muitos casos, um gargalo de desempenho, como afirmam Xu et al. (2015), Malik et al. (2016), Corbellini et al. (2017), Morrison e Sprintson (2017) e Chaudhary et al. (2018). Assim, a replicação e, consequentemente, como se dá esse processo, ou seja, como se dá a comunicação dos nós, devem ser explorados para repararem possíveis gargalos de desempenho.

[1] Nomenclatura que significa *"Not Only SQL"*, ou seja, "não apenas SQL", na tradução para o Português

Um *cluster* de bancos de dados NoSQL pode ser compreendido como um conjunto de vários bancos de dados, logicamente, conectados através de uma rede de computadores, comumente nomeados nós de rede. Desse modo, o roteamento da carga recebida, a replicação, a concorrência por largura de banda disponível, o nível de consistência adotado e a própria configuração do *ReplicaSet* podem impactar no desempenho do *cluster*, sendo necessário analisar e avaliar o desempenho a partir desses fatores. Para tanto, a escolha de como se dará essa avaliação também necessita ser eficaz e deve levar em conta a redução da complexidade do processo de análise e avaliação, e, se possível, economizar eventuais custos operacionais.

Na grande área de avaliação de desempenho e confiabilidade, o uso de formalismos, como as Redes de Petri Estocásticas (SPN), tornou-se comum na modelagem, na análise e na simulação de sistemas. Para Lima et al. (2014), os modelos, bem como as suas extensões, GSPN por exemplo, são a união da teoria das Redes de Petri e o processo de modelagem estocástica. Nessa direção, tal metodologia possibilita a redução de custos operacionais e de execução, e agiliza a análise e a avaliação de sistemas, por isso vem sendo utilizado para avaliar os bancos de dados NoSQL, como consta em Osman e Piazzolla (2014) e Bruneo (2014).

Neste trabalho, apresentamos modelos baseados em Redes de Petri Estocásticas para avaliar um sistema NoSQL, considerando os seguintes fatores: o efeito da replicação de dados pela rede e sua configuração (quantidade de nós envolvidos no processo), a consistência eventual do banco, o tráfego concorrente de nós simultâneos na largura de banda da rede e a taxa de chegada, bem como o tamanho e quantidade das requisições. Assim, o desenvolvimento e a validação de modelos, juntamente com as análises de experimentos, configuram a principal contribuição desse trabalho, visto que essa estratégia contribui para a redução de despesas e de mão-de-obra [2], e pode ser utilizada por usuários com experiência em modelos combinatórios ou modelos baseados em espaço de estados para o planejamento e tomada de decisão, servindo, portanto, de base para o auxílio da modelagem de outras características de sistemas NoSQL ou para diversos sistemas que apresentem componentes semelhantes aos abordados neste trabalho.

1.1 MOTIVAÇÃO

Embora os SGBDs NoSQL tenham se popularizado - principalmente pelo seu desempenho superior a outros SGBDs, no que se refere à manipulação de grandes volumes de dados e de requisições, além de possuírem diversos modelos de dados que apresentam características que favorecem diferentes tipos de aplicações - são sistemas distribuídos, e,

[2] No presente contexto experimental, o desenvolvimento e a validação de modelos possibilitou a realização de apenas algumas configurações em equipamentos de domínio da instituição (Cin/UFPE), reduzindo, assim, as despesas de aquisição de novos aparelhos e o aprimoramento do tempo que seria necessário para configurar novos aparelhos

portanto, a comunicação pode ser um gargalo de desempenho, tendo em vista que: i) a replicação pode exigir uma largura de banda de rede significativa (XU et al., 2015); ii) latência de rede e largura de banda limitam o desempenho do banco de dados (ABADI et al., 2016); iii) a introdução de grandes quantidades de transferência de dados dentro de uma rede de centro de dados acarreta congestionamentos (CUI et al., 2017); iv) a consistência eventual depende da latência da rede, da quantidade de réplicas, da carga do sistema, e de outros fatores (CORBELLINI et al., 2017).

Devido aos diferentes tipos de SGBDs NoSQL, que se diferem na forma de se comunicarem e em suas garantias, há uma ausência de trabalhos que avaliem a comunicação envolvendo os SGBDs NoSQL.

Tendo em vista que a modelagem focada no desempenho é útil para a tomada de decisões em análises de sistemas, como em SGBDs NoSQL, e que esse recurso, especificamente com o uso do formalismo de Rede de Petri Estocástica, já é considerado uma técnica bastante eficiente, uma vez que economiza tempo, dinheiro e trabalho experimental, motivamo-nos a aplicar tal técnica no problema do gargalo de desempenho na comunicação dos SGBDs NoSQL, levando em conta o efeito da replicação, a concorrência entre os nós na largura de banda, o nível de consistência configurada e a quantidade de nós em um *cluster*.

Consta ainda que, até a realização dessa pesquisa, encontramos trabalhos mencionando apenas o efeito da replicação como um gargalo de desempenho para a rede e para o banco, sem considerá-lo para fins experimentais; e trabalhos, nos quais o problema é citado, mas é resolvido com o uso de SDN, na tentativa de monitorar ou limitar a rede para evitar tais gargalos, através do recurso *OpenFlow QoS* (KRISHNA; ADRICHEM; KUIPERS, 2016). Nesse sentido, seguir essa última abordagem implicaria algumas consequências, como: gastos com dispositivos compatíveis para os experimentos, estudo aprofundado para aplicabilidade da abordagem, possíveis falhas de configuração e/ou excessivos testes para funcionamento do ambiente de experimentação. Mesmo assim, essa perspectiva (SDN para evitar sobrecargas/gargalos) pode ser convergida para a técnica de modelagem, a fim de que possa ser implantada no intuito de melhorar o desempenho desses sistemas, conforme, também, é proposto neste trabalho.

Portanto, além de usar uma técnica eficaz para avaliações e análises de desempenho de sistemas, o modelo de avaliação, o qual nos propomos, pode ser útil para a detecção de sobrecargas que afetam o desempenho da comunicação de SGBDs NoSQL e para o planejamento da utilização de largura de banda de rede com esses sistemas. Pode, ainda, incentivar outras pesquisas em torno do desempenho desses sistemas, considerando o aspecto da comunicação e da modelagem.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é avaliar o desempenho de Bancos de Dados NoSQL com o foco na comunicação de rede de um *cluster* NoSQL, utilizando Redes Petri Estocásticas Generalizadas para facilitar a administradores ou projetistas de rede o planejamento da estrutura computacional (rede de comunicação) para esses SGBDs.

1.2.2 Objetivos específicos

1. Criar e apresentar modelos GSPN que representem o comportamento do sistema de armazenamento na rede;
2. Apresentar a validação dos modelos criados para demonstrar sua viabilidade na representação de sistemas reais;
3. Demonstrar a aplicabilidade prática do modelo, para tal, elaborar e analisar o projeto de experimentos (DoE);
4. Apresentar o ranqueamento dos fatores principais representados pelo modelo;
5. Mostrar, através de gráficos, os efeitos dos fatores e interações dos resultados experimentais.

1.3 ESTRUTURA DA DISSERTAÇÃO

Este presente trabalho está organizado em seis capítulos, sendo o primeiro a introdução, que discute, entre outras aspectos, os objetivos e motivação do trabalho. Os demais capítulos atendem à disposição seguinte:

1. Com o objetivo de entender a proposta e a importância desta pesquisa, discutimos, no Capítulo 2, alguns trabalhos que: i) evidenciam as diferenças entre bancos de dados relacionais e bancos de dados NoSQL, que suprem melhor as necessidades do *BigData* (seção 2.1); ii) qualificam e norteiam a escolha entre os diferentes tipos de SGBDs NoSQL (seção 2.2); iii) enfatizam a importância da replicação e as desvantagens desse mecanismo (seção 2.3); iv) destacam o desempenho, a tolerância à partição e a disponibilidade dos sistemas NoSQL, considerando os aspectos da consistência (seção 2.4); v) desenvolvem estudos analisando os sistemas distribuídos em rede e possíveis sobrecargas oriundas da comunicação desses sistemas ou de aplicações concorrentes à largura de banda disponível (seção 2.5); vi) apresentam alternativas de configuração com SDN para sanar alguns problemas de sistemas distribuídos (seção 2.6); e vii) discutem sobre os diferentes métodos de avaliação

aplicados a sistemas distribuídos (seção 2.7). Ao término, as considerações finais deste capítulo (seção 2.8).

2. No Capítulo 3, discutimos o referencial teórico utilizado como base para a execução desta pesquisa. Os tópicos a seguir estão de acordo com a metodologia utilizada neste estudo. Para tanto, dividimos este capítulo em 3 partes: i) seção 3.1 e subseções tratam da comunicação de sistemas NoSQL – desde como se dá o protocolo de comunicação nos principais bancos desse gênero a custos de como manter a comunicação em certos casos e propostas que podem auxiliá-los; ii) seção 3.2 e subseções descrevem as principais características do SGBDs NoSQL, especialmente do MongoDB; iii) seção 3.3 e subseções referem-se à avaliação de desempenho – planejamento de experimentos, técnicas de avaliação, em especial, a modelagem baseada em Redes de Petri Estocástica (SPN).

3. A metodologia utilizada é discutido no Capítulo 4, em que são descritos os procedimentos metodológicos adotados nesta pesquisa, os quais estão organizados na Visão geral da metodologia (seção 4.1) e os Modelos GSPN propostos (seção 4.2).

4. Os resultados deste trabalho são esclarecidos no Capítulo 5 de acordo com os seguintes pontos: sobre a validação do modelo (seção 5.1), o projeto de experimentos (seção 5.2) e os resultados experimentais (seção 5.3).

5. Por último, discutimos a conclusão do trabalho no Capítulo 6, destacando as limitações do trabalho na seção 6.1 e os trabalhos futuros na seção 6.2.

2 TRABALHOS RELACIONADOS

Com o objetivo de entender a proposta e a importância desta pesquisa, discutimos, no Capítulo 2, alguns trabalhos que: i) evidenciam as diferenças entre bancos de dados relacionais e bancos de dados NoSQL, que suprem melhor as necessidades do *BigData* (seção 2.1); ii) qualificam e norteiam a escolha entre os diferentes tipos de SGBDs NoSQL (seção 2.2); iii) enfatizam a importância da replicação e as desvantagens desse mecanismo (seção 2.3); iv) destacam o desempenho, a tolerância à partição e a disponibilidade dos sistemas NoSQL, considerando os aspectos da consistência (seção 2.4); v) desenvolvem estudos analisando os sistemas distribuídos em rede e possíveis sobrecargas oriundas da comunicação desses sistemas ou de aplicações concorrentes à largura de banda disponível (seção 2.5); vi) apresentam alternativas de configuração com SDN para sanar alguns problemas de sistemas distribuídos (seção 2.6); e vii) discutem sobre os diferentes métodos de avaliação aplicados a sistemas distribuídos (seção 2.7). Ao término, as considerações finais deste capítulo (seção 2.8)

2.1 BANCOS DE DADOS RELACIONAIS VS BANCOS DE DADOS NOSQL

Em Jung et al. (2015), os autores selecionaram o PostgreSQL e MongoDB para representar os SGBDRs e bancos NoSQL, respectivamente, a fim de realizarem uma análise comparativa de desempenho das operações de inserção, seleção, atualização e exclusão de dados nesses sistemas. Em geral, esse trabalho deduz que as velocidades de operação do MongoDB são mais rápidas que as do PostgreSQL. Assim, essa pesquisa visa, acima de tudo, contribuir para o avanço da tecnologia de banco de dados em ambiente de *big data*, salientando que os SGBDRs, que estão sendo utilizados, apresentam problemas na estruturação de dados não estruturados e problemas de desempenho/custo no processamento de dados massivos. Todavia, com a função de mapeamento de memória, o NoSQL realiza leitura/gravação rápida - o que torna o NoSQL mais adequado para processar grandes volumes de dados - e ainda pode manipular dados não estruturados com mais facilidade.

Com o objetivo de responder se NoSQL tem melhor desempenho do que o SQL em todos os cenários para *Internet of Things* (IoT), em Rautmare e Bhalerao (2016), os autores comparam o desempenho dos bancos de dados SQL (MySQL) e NoSQL (MongoDB), quanto aos dados gerados por uma aplicação IoT criada para esse estudo. Assim, observou-se o tempo gasto para executar consultas *Select* e *Insert* em relação ao número variável de registros e *threads*. Os resultados mostram que, em alguns cenários, o MongoDB requeria menos tempo de resposta comparado ao MySQL, no entanto, as respostas do MySQL foram estáveis em comparação com o MongoDB. O artigo concluiu que escolher um banco de dados melhor para a (IoT) depende de qual consulta é mais usada e de

são quais os requisitos da aplicação.

Em Kunda e Phiri (2017), os autores discutem que ambos os bancos de dados continuarão a existir lado a lado sem que nenhum seja melhor que o outro, e a escolha do banco de dados a ser utilizado dependerá da natureza do aplicativo que está sendo desenvolvido, contudo, é possível destacar os desafios e pontos fortes de cada tipo de banco. Para tanto, o artigo se baseou em uma revisão da literatura. Assim, a respeito dos SGBDRs, o trabalho observou que a consistência, a segurança e a vantagem de uma linguagem de consulta padrão são melhores nesses sistemas, ainda que possuam pouca escalabilidade, fraco desempenho, custam mais, enfrentam desafios de disponibilidade ao oferecer suporte a um grande número de usuários e lidam com um volume limitado de dados. Já sobre os bancos de dados NoSQL, é relatado que esses sistemas enfatizam maior escalabilidade, fornecem esquema flexível, oferecem melhor desempenho, possuem código fonte aberto, mas não possuem linguagem de consulta padrão e não fornecem mecanismos de segurança adequados.

Nos trabalhos citados nesta seção, observamos que os bancos de dados NoSQL não são plataformas que irão substituir os bancos de dados tradicionais, mas que são mais adequados para aplicações que trabalham com um grande volume de dados, como Facebook, Google, Amazon e Internet das Coisas (IoT), por isso se tornam objeto de estudo em diversas pesquisas.

2.2 TIPOS DE BANCOS DE DADOS NOSQL

Tang e Fan (2016) avaliaram o desempenho de cinco *clusters* NoSQL (Redis, MongoDB, Couchbase, Cassandra, HBase), usando uma ferramenta de medição - Yahoo! Cloud Serving Benchmark (YCSB) - em operações de inserção, leitura e atualização. Esse trabalho demostra que o Redis, particularmente, carrega e executa bem as cargas de trabalho, embora tenha limitação quanto a dados extremamente grandes. Constatam, ainda, que os bancos de dados de documentos, seguidos pelos bancos de dados de família de colunas, têm um bom desempenho e possuem eficiência e escalabilidade. Além disso, destacam que a estrutura de banco de dados no modo mestre-mestre (ou seja, quando um cliente tem permissão para escrever em qualquer réplica) tem mais vantagens sobre as arquiteturas mestre-escravo (um nó "mestre"recebe todas as operações de gravação do cliente e replica no nó "escravo").

Ventura e Antunes (2016) apresentam uma avaliação da confiabilidade do MongoDB, do Cassandra e do Redis, com base no uso de injeção de falhas. Nessa pesquisa foram observados os tempos de recuperação (disponibilidade) e o *throughput* (desempenho). Os resultados indicaram que os sistemas avaliados são muito limitados para garantir a integridade dos dados, e que mais da metade dos testes executados atingiram um estado inconsistente se comparado ao que foi confirmado para o cliente. Além disso, o impacto das falhas no *throughput* não foi muito claro, pois, apenas em alguns casos, o *throughput*

foi significativamente diferente em comparação com a injeção anterior. Por outro lado, os tempos de recuperação foram, em muitos casos, bastante significativos (em falhas baseadas em reinicializações forçadas). Assim, os autores advertem que é necessário selecionar cuidadosamente o banco de dados que melhor se adapta aos objetivos do sistema em estudo.

Em Gupta et al. (2017), os autores selecionaram os bancos: MongoDB (bancos chave/-valor), Cassandra (sistemas orientados a documentos), Redis (armazenamento de famílias de coluna) e Neo4 (modelos de grafos), os quais foram comparados quanto aos seus modelos de dados, às características do teorema CAP, às propriedades distributivas e a outros fatores (recursos não funcionais). Nessa pesquisa, constatou-se que: os bancos orientados a documentos são recomendados para sistemas que necessitam de um alto desempenho, flexibilidade e escalabilidade, e se os dados têm formato JSON; o armazenamento de colunas pode ser usado para dados semiestruturados, o que requer alto desempenho e escalabilidade; os bancos de dados orientados a grafos podem ser usados quando se tratam de dados altamente interconectados e modelos de dados em constante evolução; as operações de gravação e exclusão são rápidas para os bancos de dados MongoDB, Redis e Cassandra, enquanto a operação de leitura é comparativamente lenta no Cassandra;o Redis é um armazenamento de memória que executa, excepcionalmente, rápido operações de um único *shard*; e o Neo4j tem desempenho semelhante ao MongoDB, quanto ao desempenho do *REST*.

Kuzochkina, Shirokopetleva e Dudar (2018) descrevem e comparam cada banco de dados NoSQL (Cassandra, CouchDB, Aerospike e MongoDB) de acordo com as seguintes características: categoria, posicionamento no contexto do teorema CAP, garantias de consistência e configurabilidade, garantias de durabilidade e configurabilidade, possibilidades e mecanismos de consulta, mecanismos de controle de concorrência, esquemas de particionamento, existência de particionamento nativos e replicação. Assim, os autores recomendam que: os bancos valor/chave podem ser usados para armazenamento de dados rápido e frequente para uso futuro; os SGBDs de armazenamento coluna são melhores para armazenar grandes quantidades de dados por um longo tempo; e SGBDs orientado a documentos são preferíveis para dados que devem ser armazenados, não em linhas ou colunas, mas na forma de documentos semelhantes ao JSON, e que há integridade apenas no nível de registros separados (não no nível das comunicações entre eles).

Ao estudar os trabalhos relatados nesta seção, observamos que os pesquisadores buscaram nortear qual banco de dados é melhor para determinado serviço ou aplicação em termos, principalmente, de desempenho, de confiabilidade ou de disponibilidade. No entanto, o impacto do desempenho de comunicação desses serviços na rede não recebeu a devida importância, embora haja algumas pesquisas em torno do SDN para monitorar ou limitar a rede para evitar sobrecargas, especialmente quando o sistema de banco de dados usa a replicação (técnica que fornece redundância e aumenta a disponibilidade de dados

na perda de um nó de armazenamento), conforme veremos na Seção 2.5 **(SDN aliado a sistemas distribuídos)**.

2.3 REPLICAÇÃO DE DADOS

A replicação é um mecanismo essencial para prover disponibilidade, *failover* e, em alguns casos, um melhor desempenho em *clusters* de banco de dados distribuído. No entanto, a sobrecarga provocada por esse processo tem sido constantemente abordado como um fator subjacente nos estudos de avaliação de desempenho de bancos de dados. A seguir, discutiremos alguns trabalhos que discorrem sobre a replicação em diferentes cenários.

Tendo em vista que provedores de banco de dados, como serviço (DaaS), usam a replicação para atender às garantias de desempenho e disponibilidade exigidas por seus clientes (conhecidas como SLO), em Floratou e Patel (2015), os autores estudam sobre o problema de colocar as réplicas do banco de dados de um locatário de maneira ideal, ao mesmo tempo que se atende às restrições de replicação e aos SLO de desempenho, buscando, ainda, maximizar a utilização de recursos das máquinas. Nesse trabalho é apresentado uma definição para variantes do problema de posicionamento de réplica e alguns critérios de avaliações das soluções propostas. Além disso, projetaram e avaliaram algoritmos de posicionamento de réplica para ambientes DaaS.

Yang et al. (2016) propuseram um gerenciador de réplica de dados projetado para sistemas distribuídos de cache e processamento de dados, com o uso de sistemas de armazenamento de camada SSD-HDD, uma vez que foi constatada uma sobrecarga de tráfego extra na rede, reduzindo o desempenho geral de E/S do *cluster*, causado pela replicação de sistemas de armazenamento de *data center* para sistemas Ciberfísicos em larga escala (CPS). Desse modo, o intuito principal foi realizar um balanceamento de carga entre nós e reduzir a sobrecarga de rede para configurar automaticamente a estrutura de réplica de nós cruzados, considerando o tráfego de rede, a velocidade de E/S e os SLAs (Acordos de Nível de Serviço).

Em Gessert et al. (2017), os autores fornecem uma visão geral e classificação dos bancos de dados NoSQL em face aos atributos qualitativos, requisitos funcionais e não funcionais. Além disso, os bancos são comparados de acordo com os mecanismos de gerenciamento de armazenamento, processamento de consulta, particionamento (*sharding*) e replicação. Com relação à replicação, é destacado que sistemas NoSQL, normalmente, dependem de replicação lenta, seja em combinação com o mestre-escravo ou com a abordagem de atualização em qualquer lugar. Desse modo, muitos sistemas NoSQL deixam a escolha entre latência e consistência para o cliente. Outro fator interessante levantado nessa pesquisa é o aspecto da distância entre as réplicas, em que é apontada a vantagem de se colocar réplicas próximas (baixa latência). No entanto, essa proximidade pode reduzir os efeitos positivos na disponibilidade (réplicas em um único local traz o perigo da perda total em

um cenário de desastre). Por outro lado, a replicação com dimensões geográficas pode proteger o sistema contra perda de dados completa e melhorar a latência de leitura para acesso distribuído de clientes.

2.4 CONSISTÊNCIA EM SISTEMAS NOSQL

Estimulados pela não existência de modelos matemáticos claros e pelas conclusões genéricas sobre consistência, Burdakov et al. (2016) propõem uma modelagem probabilística para avaliar a influência de réplicas de banco de dados nas operações de leitura e escrita sobre a consistência eventual de Sistemas de Gerenciamento de Banco de Dados (SGBDs) NoSQL. Dessa forma, desenvolveram um *software* para aplicação desses modelos, os quais foram usado para a avaliação da consistência de dados em um *cluster* com 50 nós. Os resultados dessa pesquisa permitem estimativas de tempo de espera aleatória da solicitação de leitura para a conclusão da atualização do registro no banco.

Visto que repositórios de dados replicados modernos têm como objetivo fornecer alta disponibilidade, respondendo imediatamente a solicitações de clientes, e que geralmente implementam objetos que não podem ser especificados sequencialmente, como, por exemplo, os *registradores de múltiplos valores*, em Attiya, Ellen e Morrison (2017), são apresentados alguns modelos matemáticos para especificar e avaliar a consistência eventual. Desse modo, comprova que um banco de dados eventualmente consistente, que implementa esses registradores, não podem satisfazer a um modelo de consistência estritamente mais forte do que a *consistência causal observável (OCC)*. Além disso, constataram limites inferiores no tamanho da mensagem e no espaço para armazenamentos de dados. Esse trabalho ressalta, ainda, que a *OCC* captura execuções nas quais as observações do cliente podem usar a causalidade para inferir a simultaneidade de operações, o que implica duas suposições sobre o armazenamento de dados: as operações de leitura não modificam o estado do banco, e as mensagens são geradas somente após uma operação.

Em Tarasyuk et al. (2015), os autores discutem a experiência da pesquisa e dos resultados teóricos da investigação do impacto da consistência na latência de sistemas distribuídos tolerantes a falhas, construídos na Internet. Esse trabalho apresenta resultados experimentais de medição do tempo de resposta para sistemas orientados a serviços replicados que fornecem diferentes níveis de consistência, mostrando, dessa forma, que melhorias na consistência aumentam a latência do sistema. Assim, modelos analíticos foram propostos para permitir a previsão de tempo de resposta (quantificado), dependendo do nível de consistência fornecido por um sistema replicado. A aplicação prática desse trabalho auxilia pesquisadores na configuração ideal de tempo limite de resposta e na compreensão do *trade-off* entre a consistência do sistema e latência.

2.5 REDE (COMUNICAÇÃO) E SISTEMAS DISTRIBUÍDOS

De acordo com o Relatório Beckman sobre Pesquisa de Banco de Dados[3], os dados podem ser proibitivamente caros de se mover, pois o armazenamento anexado à rede facilita a expansão de um mecanismo de banco de dados, no entanto, latência de rede e largura de banda limitam o desempenho do banco de dados. Desse modo, notamos que é importante considerar o fator da rede e, consequentemente, a comunicação de sistemas de armazenamento distribuídos. Assim, relacionamos, nesta seção, alguns trabalhos que sintetizam e contribuem para esta discussão.

Xu et al. (2015) definem que a largura de banda de rede necessária para manter as réplicas sincronizadas é cara e costuma ser um gargalo de desempenho em *Sistemas de banco de dados orientados a documentos (document-oriented databases)*. Nesse sentido, o trabalho apresenta um sistema de deduplicação, chamado sDedup, para reduzir a quantidade de dados transferidos pela rede para *document-oriented databases* de documentos replicados. A avaliação experimental da metodologia proposta mostra uma redução de 38× nos dados enviados pela rede, superando, significativamente, as técnicas tradicionais de deduplicação baseada em fragmentos e incorrendo em sobrecarga insignificante de desempenho. Essa pesquisa elucida, principalmente, sobre a redução de largura de banda de replicação em sistemas distribuídos (incluindo uma integração a *framework* de replicação do MongoDB), apresentando o uso geral do sDedup e as opções e otimizações de design que são importantes para o uso da deduplicação em linha nos bancos de dados de documentos.

Partindo do principio que a replicação de dados é necessária para fornecer alta disponibilidade e eficiência, González-Aparicio et al. (2016) testaram os bancos de dados chave/valor do NoSQL, considerando as operações CRUD transacionais e não transacionais. Assim, propuseram um modelo sensível ao contexto, que leva em conta as informações contextuais do cliente e do sistema. Isso é importante para aplicações de *big data*, nas quais as operações CRUD devem gerenciar recursos adicionais, como a replicação de dados em diferentes nós e solicitações simultâneas de vários clientes, fornecendo, assim, a disponibilidade de dados. Dessa forma, esse trabalho traz à baila a discussão sobre *trade-off* entre consistência de dados, disponibilidade e eficiência.

Em Cui et al. (2017), os autores propuseram a criação conjunta de aplicativos de rede e de nuvem para otimizar o desempenho na presença de congestionamentos na rede de um *data center*, em que foi construído uma estrutura de banco de dados distribuída com reconhecimento de rede que usa informações de estado de rede para melhorar o desempenho. Aqui, é apresentada duas técnicas para reduzir o tempo de conclusão de transação para aplicativos de banco de dados em nuvem: primeiramente, aplica-se um algoritmo de

[3] Relatório da reunião de um grupo de pesquisadores de banco de dados que se reúnem, periodicamente, para discutir o estado do campo e suas principais direções. Essa reunião se deu no Centro Beckman, no campus da Universidade da Califórnia-Irvine em 2016 (ABADI et al., 2016).

balanceamento de carga na camada de rede para suportar a seleção inteligente de rotas de rede, após isso, é introduzido um algoritmo de cache com reconhecimento de rede para recuperar réplicas de dados recentes de *link*s na rede com baixo congestionamento. Os resultados experimentais dessa pesquisa mostram que as técnicas aplicadas podem reduzir significativamente os tempos médios de conclusão de transação em comparação com uma linha de base fundamentada em ECMP, ou seja, roteamento de vários caminhos de custo igual.

Tendo em vista que na medida em que os sistemas se tornam mais complexos, são necessários métodos mais novos e mais eficientes para gerenciar e, possivelmente, diminuir a sobrecarga causada por entidades concorrentes. Nesse direção, Bérces, Imre e Prakfalvi (2018) fizeram proveito dos avanços do campo da computação quântica para modificar o MAC com *slot-ALOHA* para obter um protocolo de resolução de competição moderno e eficiente. Assim, esse estudo faz uma análise do desempenho de um mecanismo de controle de acesso a recursos de sistemas distribuídos através de um algoritmo baseado em emaranhamento quântico que fornece acesso justo a todas as entidades participantes. Para tanto, um ambiente de simulação foi criado com o software MATLAB para modelar a implantação e o funcionamento da rede, possibilitando, assim, a outros trabalhos a inclusão da extensão a outros modelos de movimento, bem como a simulação de dados de tráfego real.

2.6 SDN ALIADO A SISTEMAS DISTRIBUÍDOS

Um dos primeiros trabalhos a analisar e mostrar as oportunidades das Redes definidas por software (SDN) para a otimização de consultas distribuídas são os estudos de Xiong, Hacigumus e Naughton (2014). Os autores apresentam um método que observa o *status* da rede e reage adaptando o plano de execução de consulta a um que produz melhor desempenho. Os métodos permitiram que um processador de consulta distribuída oferecesse um serviço de consulta diferenciado aos usuários com diferentes prioridades: um permite a priorização do tráfego de rede e o outro fornece a capacidade de reserva de determinada quantidade de largura de banda para consultas específicas e uso da largura de banda garantida durante a otimização da consulta. O estudo ocorreu com a implementação das técnicas em um protótipo de site com armazenamento de dados relacional (PostgreSQL), executando em uma SDN com switches comerciais *OpenFlow*. Desse modo, esse trabalho demostrou que os métodos propostos exibem um potencial significativo para o gerenciamento de desempenho de consultas analíticas em armazenamentos de dados distribuídos.

É sabido que os sistemas de arquivos distribuídos (aqui, nomeados genericamente de DFS) permitem o armazenamento de exabytes de informações em servidores distribuídos por uma rede. Assim, esses sistemas alcançam confiabilidade e desempenho armazenando várias cópias de dados em diferentes locais da rede. No entanto, o gerenciamento dessas cópias de dados é comumente feito por servidores intermediários que rastreiam e coorde-

nam o posicionamento de dados na rede, acarretando, dessa forma, possíveis gargalos na rede, já que várias transferências rápidas para nós de armazenamento podem saturar o *link*. Todavia, com o advento dos sistemas abertos, como o *OpenVSwitch* e a programabilidade de SDN, surgiu uma oportunidade para aliviar esse gargalo. Tendo isso em vista, em Morrison e Sprintson (2017), os autores propõem uma arquitetura de processamento de pacotes dentro da rede para DFS, na qual o experimento consistia em descarregar várias ações dos DFS em um componente do sistema, cujo é executado em dispositivos de rede, como roteadores e switches. Desta forma, a arquitetura proposta resultou em redução do congestionamento da rede e em uma diminuição na latência da solicitação proporcional ao número de nós de armazenamento envolvido nas requisições.

Chaudhary et al. (2018) discutem sobre o grande número de dados e *data centers* distribuídos geograficamente. Para processar essa enorme quantidade de dados e para que os usuários obtenham uma resposta rápida, o Hadoop e o SPARK são amplamente utilizados por esses provedores de serviços. No entanto, menos ênfase foi dada na Infraestrutura subjacente (a rede pela qual os dados fluem). Os autores argumentam, então, que devido ao intenso tráfego de rede em relação às migrações de dados em diferentes *data centers*, a infraestrutura de rede subjacente pode não conseguir transferir pacotes de dados da origem para o destino, resultando na degradação do desempenho. Portanto, propuseram uma abordagem de gerenciamento de *big data* baseada em SDN(especificamente, um controlador *OpenFlow*, para gerenciar o fluxo de tráfego entre diferentes centros de dados em nuvem) com relação ao consumo otimizado de recursos de rede, como largura de banda de rede e unidades de armazenamento de dados.

2.7 AVALIAÇÃO DE SISTEMAS NOSQL

Diversas pesquisas foram conduzidas com o objetivo de comparar o desempenho de sistemas de armazenamento de dados (sejam bancos relacionais ou NoSQL), entretanto, poucos modelos estocásticos ou matemáticos foram concebidos para avaliação ou análise desses sistemas, tendo em vista que tal metodologia poupa maiores esforços e menos gastos. Na maioria dos casos, os modelos concebidos não levam em conta a infraestrutura subjacente, a qual se faz necessário para a utilização desses sistemas (comunicação ou configuração da rede). Já em outros casos, a avaliação e outras análises ocorrem com base em relatórios de saídas de *benchmark* – que, por sua vez, nem sempre trazem resultados que satisfazem a realidade, principalmente se não houver um planejamento correto sobre quais dados devem ser observados -, em outros casos, ocorre a não existência de mecanismos já desenvolvidos para extração de determinadas métricas. Todavia, tais trabalhos apresentam questionamentos, soluções, ferramentas extras ou modelos prontos que sevem como base para outras pesquisas e, portanto, são necessários para a discussão desta seção.

Em Bruneo (2014), o autor aplica um modelo analítico, baseado em Redes de recompensa estocásticas (SRN), para avaliar o gargalo de sistemas de bancos de dados em um

ambiente de computação em nuvem, considerando parâmetros que quantificam o QoS experimentado por usuários e do ponto de vista do provedor (utilização, disponibilidade, tempo de espera e de resposta). Além disso, o autor analisa a resiliência para levar em conta as rajadas de carga. Também é apresentada uma abordagem geral, que, partindo do conceito de capacidade do sistema, pode ajudar os gerentes de sistemas a definir, oportunamente, os parâmetros do *data center* sob diferentes condições de trabalho.

Osman e Piazzolla (2014) definem que as características de desempenho de sistemas NoSQL são determinadas pelo grau de replicação de dados e pelas garantias de consistência exigidas pela aplicação. Desse modo, o trabalho faz uso de Redes de Petri para avaliar o desempenho dos SGBDs NoSQL (com base no Cassandra implantado na plataforma de nuvem do Amazon EC2), focando na relação entre o tamanho do *cluster* e sua da consistência. O modelo construído não representa o efeito do atraso da rede, mas mostra que o modelo fornece valores próximos ao sistema real a que se propõe avaliar (como em tempos de resposta e taxa de transferência). Dessa forma, a pesquisa contribui para identificação do efeito que a capacidade e a configuração do nó têm sobre o desempenho geral do *cluster*, qualificando esses aspectos como fatores importantes a serem levados em consideração em pesquisas que propõem uma abordagem parecida com a que é proposta neste trabalho.

Em Chandra (2015), o autor faz um estudo analítico das propriedades BASE de alguns bancos de dados NoSQL. Também discute as diferentes técnicas que são usadas para alcançar consistência e faz, ainda, recomendações baseadas em observações sobre a seleção de um NoSQL para fins específicos. O trabalho destaca o Yahoo! Cloud Serving Benchmark (YCSB) como uma ferramenta de estrutura extensível popular, projetada para comparar diferentes armazenamentos de tabelas em cargas de trabalho sintéticas idênticas, mas que pode ser usado para medir o desempenho básico de bancos NoSQL, tais como o HBase, Voldemort, Cassandra e MongoDB. Além disso, a pesquisa ressalta que o YCSB tem sido usado em mais de 70 comparações de *benchmark* publicadas, e representa uma maneira padronizada de comparar o desempenho entre sistemas. Porém, o trabalho também menciona o SandStorm (ferramenta de teste de desempenho corporativo para aplicativos *web*, móveis e *Big data*), que fornece uma estrutura para o *benchmarking* dos sistemas de fila de mensagem e Hadoop, e oferece suporte para as seguintes tecnologias NoSQL: MongoDB, Cassandra, HBase e Oracle NoSQL.

Swaminathan e Elmasri (2016), González-Aparicio et al. (2017), Reniers et al. (2017) e Martins, Abbasi e Sá (2019) fazem uso dos recursos dos *benchmarks* para produzir os trabalhos de avaliação de desempenho sobre algum aspecto dos SGBDs NoSQL. Em comum, usam o YCSB para tal fim. No primeiro estudo, para a análise das variações no desempenho da taxa de transferência (operações por segundo), em relação à contagem de operações, foi feito o cálculo da média dos *throughputs*, com base nos resultados do *benchmark* sobre os sistemas MongoDB, Cassandra e HBase. No segundo trabalho, são

investigados os efeitos das transações na consistência e eficiência dos dados nos aplicativos do usuário. Assim, o MongoDB e o Riak são avaliados usando o *benchmark* YCSB +T, que é baseado no YCSB. Na terceira pesquisa, é apresentado um estudo do *benchmark* para quantificar e comparar a sobrecarga de desempenho introduzida pelo ONDMs, que é uma opção de interface de abstração uniforme para diferentes tecnologias NoSQL para operações de criação, leitura, atualização e busca, na qual é envolvido cinco dos ONDMs mais promissores e prontos para o setor: Impetus Kundera, Apache Gora, Eclipse Link, DataNucleus e Hibernate OGM. Já na última proposta, são avaliados cinco dos mais populares bancos de dados NoSQL: MongoDB, Cassandra, HBase, OrientDB, Voldemort, Memcached, e Redis, com base nos aspectos relacionados ao desempenho da consulta (leituras e atualizações), a fim de comparação em diferentes cargas de trabalho que o YCSB permite executar.

2.8 CONSIDERAÇÕES FINAIS

Este capítulo descreveu os principais trabalhos relacionados nesta dissertação. Ainda assim, levando em conta os pontos que consideramos os mais importantes, que torna viável a comparação da nossa pesquisa com trabalhos citados anteriormente, formulamos um quadro (Quadro 1) que resume e compara as pesquisas. Para tal, consideramos os seguintes pontos:

- Qual ou quais o(s) SGBDs estudado(s), em que é possível distinguir qual sistema é foco da pesquisa (Coluna Banco (s) estudado (s));
- Resumimos, na Coluna parâmetro(s) e/ou métrica(s) considerada(s), qual ou quais parâmetro(s) e/ou métrica(s) foram utilizadas nos trabalhos;
- Qual ou quais o(s) Método(s) de avaliação utilizados, por exemplo, se houve uso de carga sintética para realização da pesquisa, estão descritos na Coluna Método(s) de avaliação utilizado (s);
- A fim de diferenciar nossa contribuição com as demais, descrevemos, resumidamente, a principal contribuição das pesquisas mencionadas na Coluna Contribuição;
- Destacamos se as pesquisas auxiliam ou não no planejamento de infraestrutura computacional com foco principal no planejamento de largura de banda para SGBDs, cuja é a nossa principal contribuição;
- Acentuamos quais das pesquisas propuseram modelos matemáticos ou baseado em SPN, tal como a metodologia de nosso trabalho;
- Por fim, apontamos quais as pesquisas realizam testes experimentais para avaliação de desempenho, por exemplo com uso de DoE.

Quadro 1 – Trabalhos relacionados

Pesquisas	**Banco(s) estudado(s)**	**Parâmetro(s) / Métrica(s) considerada(s)**	**Método(s) de avaliação utilizado**	**Contribuição**	**Auxilia no planejamento de largura de banda para SGBDs?**	**Propõe modelos matemáticos ou baseado em SPN?**	**Realiza testes experimentais para avaliação de desempenho?**
Planejamento de infraestrutura computacional para SGBDs NoSQL (Esta pesquisa)	NoSQL (MongoDB)	vazão	Uso *Benchmark* (carga sintética) para validação dos modelos e planejamento experimental para análise dos resultados	Propõe modelos que auxilia no planejamento de largura de banda para bancos de dados NoSQL	sim	sim	sim
Jung et al. (2015)	SGBDR (PostgreSQL) e SGBD NoSQL (MongoDB)	Desempenho das operações de inserção, seleção, atualização e exclusão de dados	Execução de cargas sintéticas nos sistemas de armazenamento	Distingue as principais diferenças e compara o desempenho entre os sistemas de armazenamento com base em *workloads* reais	não	não	sim
Rautmare e Bhalerao (2016)	SGBDR (MySQL) e NoSQL (MongoDB)	Tempo gasto para executar consultas Select e Insert em relação ao número variável de registros e threads	Carga oriunda de uma aplicação IoT real e estudo comparativo dos sistemas de armazenamento	Observa o desempenho dos bancos para cenários para Internet of Things (IoT)	não	não	sim

Kunda e Phiri (2017)	Considera as características mais comuns e frequentes em ambos os tipos de SGBDRs e SGBDs NoSQL	Tipo da plataforma(*open source* ou fechada), Escalabilidade, Custo, Capacidade de dados, Disponibilidade, Desempenho, etc.	Revisão da literatura	Oferece um guia fácil para nortear a escolha de um banco para aplicações em que se exige determinadas características explicitadas pelo trabalho	não	não	não
Tang e Fan (2016)	SGBDs NoSQL (Redis, MongoDB, Couchbase, Cassandra e Hbase)	Operações de inserção, leitura e atualização	Utiliza a ferramenta YCSB para executar e analisar as operações sobe uma carga de trabalho sintética	Compara o desempenho de cinco dos principais *clusters* NoSQL	não	não	sim
Ventura e Antunes (2016)	SGBDs NoSQL (MongoDB, Cassandra e Redis)	Número de elementos confirmados ou atualizados corretamente no banco, tempos de recuperação e *throughput*	Avalia a confiabilidade dos bancos com base no uso de injeção de falhas	Oferece resultados úteis sobre a integridade, disponibilidade e rendimento dos bancos estudados	não	sim	sim
Gupta et al. (2017)	SGBDs NoSQL (MongoDB, Cassandra, Redis e Neo4j)	Modelos de dados, propriedades distributivas e recursos funcionais e não funcionais	Revisão da literatura	Comparação detalhada dos principais bancos NoSQL com base em requisitos funcionais e não-funcionais, além de outras propriedades	não	não	não

Kuzochkina, Shirokopetleva e Dudar (2018)	SGBDs NoSQL (Cassandra, CouchDB, Aerospike e MongoDB)	Categoria, posicionamento no contexto do teorema CAP, garantias de consistência, replicação, etc.	A comparação é obtida com base em critérios submetidos a um modelo matemático	Um modelo matemático para ponderação dos critérios comparativos	não	sim	sim
Floratou e Patel (2015)	(DaaS) para bancos similares ao Microsoft SQL Server	Número total de máquinas disponíveis, fator de replicação, distribuição da classe de inquilino p_i, probabilidade de um inquilino pertencente à classe de inquilino c_i, etc.	Avalia alguns algoritmos de posicionamento de réplica em configurações DaaS com vários inquilinos	Estuda o problema de colocar as réplicas do banco de dados de um locatário de maneira ideal	não	não	sim
Yang et al. (2016)	Sistemas distribuídos de armazenamento de camada SSD-HDD	Atraso de rede, velocidade de acesso de E/S, taxa de utilização de espaço / *throughput*, etc.	Virtualização dos nós de armazenamento para experimentação	Um gerenciador de réplica para o balanceamento da carga entre os nós e p/ reduzir a sobrecarga de rede em sistemas de armazenamento em larga escala (Ciberfísicos)	sim	não	sim
Gessert et al. (2017)	Bancos de dados NoSQL	Atributos qualitativos, requisitos funcionais e não funcionais, mecanismos de gerenciamentos e de processamento, *sharding* e replicação	Revisão da literatura	Oferece um guia detalhado para facilitar a escolha do banco para obter um melhor desempenho dependendo dos requisitos das aplicações	não	não	não

Burdakov et al. (2016)	Bancos de dados NoSQL (Riak)	Atraso e probabilidade de leitura de dados obsoletos	Modelagem probabilística para avaliar a influência de réplicas nas operações de leitura e escrita sobre a consistência eventual em um ambiente virtualizado	Uma ferramenta de software que foi desenvolvida para aplicação prática dos modelos	não	sim	sim
Attiya, Ellen e Morrison (2017)	Banco de dados eventualmente consistente que implementa registradores de múltiplos valores e que estão relacionados com o *teorema CAP*	Propagação de gravação, tamanho da mensagem e espaço para armazenamentos de dados	Apresenta modelos matemáticos para especificar e avaliar a consistência eventual	Modelos matemáticos que contribuem para a discursão sobre os modelo de consistência existentes	não	sim	sim
Tarasyuk et al. (2015)	Bancos de dados NoSQL (Cassandra)	Tempo de resposta	Modelos analíticos para previsão de tempo de resposta	Endossam o *trade-off* sobre consistência e latência dos SGBDs	não	sim	sim
Xu et al. (2015)	Banco de dados orientados a documentos (MongoDB)	Taxa de Compressão, sobrecarga da memória de indexação, recuperação de falha, *throughput*, sharding,	Análise dos resultados de um módulo de desduplicação que foi desenvolvido e integrado na replicação do SGBD adotado no experimento	Uma metodologia que reduz os dados enviados pela rede, incorrendo em uma sobrecarga insignificante de desempenho	sim	não	sim
González-Aparicio et al. (2016)	Bancos de dados chave/valor do NoSQL	Desempenho das de operações CRUD que leva em conta informações contextuais de usuários	Experimento conduzido em ambiente um real	O modelo proposto testa operações CRUD em um modo transacional ou não transacional	não	não	sim

Cui et al. (2017)	Bancos de dados baseado no modelo chave/valor	Tamanho dos dados, tamanho do fluxo de máximo de dados, tempo médio entre chegadas, tempo de chegada, número de pares de valores-chave no banco de dados, etc.	Emulação da topologia do *data center* usando o Mininet e análise dos algoritmo de balanceamento de carga de rede	Minimiza a agregação de tráfego da camada de enlace de redistribuir a carga de trabalho	sim	não	sim
Bérces, Imre e Prakfalvi (2018)	Modelos de sistemas de dados distribuídos	Tamanho da população, tamanho da área, probabilidade de entrar e sair da rede e propriedades do modelo de movimento	Ambiente de simulação feito no MATLAB p/ modelagem da rede e análise do desempenho de através de um algoritmo baseado em emaranhamento quântico	Computação quântica para criação algoritmos que inclui a extensão de modelos de movimento como no caso de rede como sistema distribuído	sim	não	sim
Xiong, Hacigumus e Naughton (2014)	SGBDR (PostgreSQL)	tráfego de entrada, tempo de execução de consulta e *throughput*	Utilização de cargas de trabalho sintetica	Otimiza o gerenciamento de desempenho de consultas distribuída com SDN	sim	não	sim
Morrison e Sprintson (2017)	DFS	Tempo de conclusão da gravação, Tempo total de conclusão da solicitação para gravação multiusuário	Utilização de *benchmarks* e a construção de uma versão em pequena escala de um sistema de arquivos	A proposta reduziu o congestionamento da rede e diminui na latência da solicitações proporcional ao número de nós de armazenamento envolvidos na solicitação	sim	não	sim

Chaudhary et al. (2018)	Centros de dados em nuvem	Operações de inserção	Criou-se um algoritmo de balanceamento de carga no sistema	apresentada uma arquitetura para um controlador SDN que gerencia o fluxo de tráfego entre diferentes centros de dados em nuvem	sim	não	sim
Bruneo (2014)	Centros de dados em nuvem	Utilização, disponibilidade, tempo de espera e de serviço	Desenvolvimento de um modelo analítico	A solução do modelo analítico, que quantifica os efeitos de diferentes estratégias baseadas em nuvem sobre o desempenho do sistema	não	sim	sim
Osman e Piazzolla (2014)	SGBD NoSQL (Cassandra)	Tempo de resposta e *throughput*	Utilizou-se a ferramenta Cassandra-stress	O modelo que prediz com precisão os tempos de resposta e rendimento para diferentes configurações para *clusters* Cassandra	não	sim	sim
Chandra (2015)	SGBDs (Dynamo, GFS (Sistema de Arquivos do Google), Bigtable e Hadoop)	Integração com a *web*, uso de SSD, replicação, compensações entre a consistência, a disponibilidade e a tolerância a partição, etc.	Revisão da literatura	As recomendações para a seleção de um sistema NoSQL p/ determinados fins, com base em um estudo analítico das propriedades BASE, com foco nos SGBDs de grande escala	sim	não	não

3 REFERENCIAL TEÓRICO

Neste capítulo, discutimos o referencial teórico utilizado como base para a execução desta pesquisa. Os tópicos a seguir estão de acordo com a metodologia utilizada neste estudo. Para tanto, dividimos este capítulo em 3 partes: i) seção 3.1 e subseções tratam da comunicação de sistemas NoSQL – desde como se dá o protocolo de comunicação nos principais bancos desse gênero a custos de como manter a comunicação em certos casos e propostas que podem auxiliá-los; ii) seção 3.2 e subseções descrevem as principais características do SGBDs NoSQL, especialmente do MongoDB; iii) seção 3.3 e subseções referem-se à avaliação de desempenho – planejamento de experimentos, técnicas de avaliação, em especial, a modelagem baseada em Redes de Petri Estocástica (SPN).

3.1 COMUNICAÇÃO, CONFIGURAÇÃO E MÉTRICAS DE REDE PARA SISTEMAS DISTRIBUÍDOS

Um banco de dados distribuído pode ser entendido como um conjunto de várias bases de dados logicamente inter-relacionadas, distribuídas por uma rede de computadores (comumente chamados de nós). Semelhante a esse conceito, temos o conceito de *cluster*, que também pode ser entendido como máquinas interligadas por uma rede local, mas que se comportam como se fossem uma só máquina. O ponto interessante e importante entre esses conceitos, que nada mais é que o "modus operandi" desses bancos, pois é necessário se pensar na configuração da rede (seja ela localmente ou geograficamente) de forma a prover melhor tal serviço e demais demandas.

A previsão de redes de computadores cada vez mais rápidas, e que hoje já atingem taxas de transferência de gigabytes por segundo, prenuncia desafiadores problemas no que tangue à QoS, que é, simplificadamente, a capacidade de melhorar os serviços trafegados na rede. Aliado a essa situação, aplicações *web* emergentes (Facebook, Google, Amazon) e o desenvolvimento da Internet das coisas (IoT) trazem consigo os SGBDs NoSQL – sistema de dados mais adequado para tais demandas –, que também merecem atenção no tocante à comunicação e a possíveis sobrecargas oriundas de seus mecanismos nativos, como a replicação (cópia de registros de dados em outros servidores). Embora seja essencial para atender as garantias de desempenho e disponibilidade, de acordo com alguns autores, como Yang et al. (2016), a replicação tem como efeito colateral a sobrecarga de tráfego adicional de rede.

O tráfego trocado entre comutadores na rede cresceu rapidamente, e cada vez mais dados são transmitidos em redes de *data centers*. Isso significa que pode haver problemas de congestionamento, causando longos atrasos e baixa vazão (taxa de transferência) (LAN; WANG; HSU, 2016).

Um índice comumente adotado do desempenho percebido por uma comunicação de rede é a taxa de transferência máxima alcançável (vazão), que depende da capacidade restante ao longo do caminho, isto é, a largura de banda disponível (PERSICO et al., 2015). No contexto de redes de dados, a largura de banda quantifica a taxa de dados, na qual um link ou caminho de rede pode transferir (PRASAD et al., 2003). Para muitos aplicativos com uso intenso de dados, como os SGBDs NoSQL, a largura de banda disponível para o SGBD afeta diretamente o desempenho desse sistema.

A vazão pode ser entendida como a velocidade da viagem de uma determinada quantidade de dados de um lugar para outro. Broach et al. (1988) definem que a vazão são dados transferidos ao longo de um período de tempo expresso em kilobits por segundo (kbps), ou a razão entre os pacotes de dados enviados para os pacotes de dados recebidos. Em geral, quanto maior a largura de banda de um determinado caminho, maior a taxa de transferência de dados. Tendo isso em vista, pode-se dizer que para garantir o QoS, principalmente no que diz respeito à comunicação de servidores de dados distribuídos, é necessário, por vezes, superar problemas de comunicação referentes à latência, ao atraso na entrega, à perda de pacotes e aos congestionamentos. Dessa forma, é imprescindível estabelecer métricas para medição de tráfego de rede a fim de medir a quantidade e o tipo de tráfego em uma determinada rede, cuja é especialmente importante no que diz respeito ao gerenciamento efetivo de largura de banda. Analisando, por exemplo, a vazão e ajustando-a como medida preventiva, o sistema pode tornar-se mais eficiente e mais preparado para lidar com restrições extras de largura de banda em momentos de uso intenso.

Congestionamentos podem ocorrer quando as demandas de recursos dos usuários ou aplicativos excedem a capacidade disponível da rede, geralmente ocorrendo em pontos de estrangulamento na rede, em que o tráfego total de entrada para um nó excede a largura de banda de saída (SILVA et al., 2015). As métricas de desempenho que podem ser usadas para avaliar os protocolos de controle de congestionamento da Internet incluem vazão, *delay*, *Packet loss rates*, dentre outros. Algumas dessas métricas podem ter diferentes interpretações, dependendo da sua referência à rede, a um fluxo ou a um usuário. Por exemplo, a vazão pode ser medida como uma métrica baseada em taxa de transferência de *link* agregado, como uma métrica baseada em fluxo de tempos de transferência por conexão, ou como métrica de utilitário baseada em usuário, de acordo com o Transport Modeling Research Group (TRMG) em Floyd (2008).

Lan, Wang e Hsu (2016) propuseram um algoritmo *DLPO (DynamicLoad-balanced Path Optimization)* baseado em SDN, que pode: ser adequado para diferentes topologias de rede de centros de dados; alterar caminhos de fluxos durante a transmissão de fluxo; alcançar balanceamento de carga entre diferentes *link*s em uma rede de *data center*; e resolver, eficientemente, o problema de congestionamento. O trabalho é impulsionado pelo fato que a arquitetura SDN desacopla o plano de dados e o plano de controle, o que

permite aos administradores de rede programar o comportamento de suas redes com um controlador SDN, sendo possível alterar diretamente o comportamento de encaminhamento de pacotes ou fluxos pela rede. Mais detalhes sobre esse tipo de configuração de rede em benefício a sistemas de bancos de dados distribuídos, como os SGBDs NoSQL, serão descrito na Subseção 3.1.3.

3.1.1 Protocolo de comunicação dos SGBDs NoSQL

Os Protocolos de Rede são conjuntos de regras que permitem a comunicação entre computadores conectados na internet. Uma das funções dos protocolos é pegar os dados que serão transmitidos pela rede, dividir em pequenos pedaços (pacotes), onde dentro de cada pacote há informações de endereçamento que informam a origem e o destino do pacote. Dessa forma, cada serviço ou aplicação de rede estabelecem diferentes protocolos de comunicação. Da mesma forma, os diferentes tipos de SGBDs NoSQL se diferem na maneira de se comunicarem, como podemos ver a seguir nos 3 mais populares sistemas de gerenciamento de banco de dados: NoSQL (Mongo, Redis e Cassandra) de acordo com o ranque da DB-Engines (DB-ENGINES, 2019).

Hendawi et al. (2018) descrevem o Cassandra como um sistema distribuído, em que cada nó age como mestre e escravo. O sistema executa todas as funções críticas de maneira descentralizada. Os nós se comunicam usando um protocolo de comunicação *peer-to-peer*, trocando periodicamente informações de estado sobre si mesmo e sobre os outros nós que conhecem. Esse protocolo é conhecido como *protocolo Gossip*.

O RedisDB é um armazenamento de chave/valor avançado de código aberto, e é geralmente chamado de servidor de estrutura de dados, pois as chaves podem conter *Strings, Hashes, List, Sets* e *Sorted-sets*. O protocolo Redis consiste em uma camada de rede na qual os clientes conectam-se à porta 6379. Todos os nós estão conectados entre si e as funcionalidades de cada nó são equivalentes. Como em qualquer outro protocolo, o protocolo Redis também tem um cabeçalho (*Meta information*) e uma parte do corpo (*request data*). A parte de dados da solicitação consiste em informações, como dados de comando e o próprio comando. Em resposta, ele conterá partes como *Meta information* (se a solicitação foi um sucesso ou falha) e *payload* de dados de resposta real (THANTRIWATTE; KEPPETIYAGAMA, 2011; DAS, 2015).

O MongoDB opera de acordo com uma arquitetura cliente-servidor. Essa interação é fundamentada no protocolo *Wire* do MongoDB, que é um protocolo baseado em soquete TCP/IP. Os drivers do MongoDB e o *shell* do MongoDB convertem consultas e comandos, possivelmente definidos com diferentes linguagens de programação, para mensagens do MongoDB *Wire*. Da mesma forma, as respostas às solicitações do cliente são emitidas pelo servidor na forma de mensagens do protocolo *Wire*. O *front-end* do servidor MongoDB decide estabelecer uma conexão e, nesse caso, permite que os clientes enviem comandos e consultas de administração. O *back-end* do servidor pode ser composto de um único nó ou

vários nós distribuídos em um *cluster*, que permite lidar com grandes volumes de dados, garantindo, ao mesmo tempo, bom desempenho e disponibilidade. O cliente interage com o servidor de maneira independente da arquitetura do lado do servidor adotada (COLOMBO; FERRARI, 2015).

3.1.2 Replicação e Consistência vs custo adicional

Bancos de dados orientados a documentos estão se tornando mais populares devido à prevalência de dados semiestruturados e ao fato de permitirem que entidades sejam representadas de maneira não esquematizada, usando uma hierarquia de propriedades. Como esses Sistemas de Gerenciamento de Banco de Dados (SGBDs) são normalmente usados com aplicativos voltados para o usuário, costumam estar sempre *online* e disponíveis. Para garantir essa disponibilidade, tais sistemas replicam dados entre nós com algum nível de diversidade, por exemplo, um SGBD pode ser configurado para manter réplicas em *data centers* (nós em *racks* diferentes, *clusters* diferentes) ou em *data centers* em regiões geograficamente separadas (XU et al., 2015).

A replicação pode exigir uma largura de banda de rede significativa, tornando essa cada vez mais escassa e cara quanto mais longe as réplicas estiverem localizadas de seus nós de primários(XU et al., 2015). Assim, ela não apenas impõe custo adicional para a manutenção de réplicas, como também pode se tornar o gargalo para o desempenho do SGBD se o aplicativo não puder tolerar divergência significativa entre as réplicas. Esse problema é, especialmente, inoportuno em cenários de replicação geográfica, em que a largura de banda da *WAN* é cara e a capacidade cresce de maneira relativamente lenta nas atualizações de infraestrutura ao longo do tempo.

O desempenho de um banco de dados NoSQL distribuído pode ser afetado por vários fatores: complexidade da solicitação, já que cada solicitação única tem um padrão de acesso diferente e requer uma quantidade diferente de operações de disco; concorrência, visto que a execução simultânea de duas classes distintas de carga de trabalho pode gerar interferências causadas pelo compartilhamento de recursos; capacidade do sistema, tendo em vista que os bancos de dados, como SGBDs NoSQL, empregam escalas horizontais para melhorar o desempenho do sistema, tomando como estratégia fundamental a replicação total ou parcial para implementação dessas escalas, mas incorrendo em uma sobrecarga para manter a consistência entre as cópias. Além disso, as estratégias de consistência podem usar bloqueios distribuídos que causam esperas e *deadlocks*, e, consequentemente, o desempenho é degradado de forma não linear (FARIAS et al., 2018).

As propriedades conhecidas como ACID (atomicidade, consistência, isolamento e durabilidade) aumentam a complexidade do projeto de sistemas de banco de dados e ainda mais dos bancos de dados distribuídos, que distribuem dados em múltiplas partições ao longo de uma rede de computadores. Esse recurso, no entanto, simplifica o trabalho do desenvolvedor, garantindo que todas as operações deixem o banco de dados em um estado

consistente. Nesse contexto, as operações são suscetíveis a falhas e atrasos da própria rede, sugerindo, assim, que precauções adicionais podem ser tomadas para garantir o sucesso de uma transação. Normalmente, nos SGBDRs (Sistemas de Gerenciamento de Banco de Dados Relacionais), que seguem as propriedades ACID, todas as réplicas são persistentes e qualquer leitura retornará a dados atualizados, garantindo uma consistência forte desses sistemas. A consistência fraca ocorre quando as leituras podem obter dados desatualizados. Dessa forma, a consistência eventual é um caso especial de consistência fraca, na qual há garantias de que se um dado for gravado no sistema, ele eventualmente atingirá todas as réplicas, mas isso dependerá da latência da rede, da quantidade de réplicas, da carga do sistema, dentre outros fatores que devem ser levados em conta na escolha de um SGBD NoSQL que assimila esse modelo (CORBELLINI et al., 2017).

Embora seja comum a ocorrência de falhas em redes de centros de dados, quando se aplica a replicação e outras técnicas tolerantes a falhas nos sistemas, baseados na Internet e na nuvem, é preciso entender, sobretudo, as sobrecargas de tempo e se preocupar com atrasos e incertezas. Um sistema replicado, tolerante a falhas, torna-se particionado quando uma de suas partes não responde, devido à perda arbitrária de mensagens, atraso ou falha de réplica, resultando em um tempo limite. A falha em receber respostas de algumas das réplicas dentro do tempo limite especificado causa o particionamento do sistema replicado. Assim, o particionamento pode ser considerado como um limite no tempo de resposta da réplica. Uma conexão de rede lenta, uma réplica de resposta lenta ou as configurações incorretas de tempo limite podem levar a uma decisão errônea de que o sistema foi particionado. Quando o sistema detecta uma partição, tem que decidir se é para retornar uma resposta possivelmente inconsistente a um cliente ou para enviar uma mensagem de exceção em resposta, o que compromete a disponibilidade do sistema (TARASYUK et al., 2015).

Em um tipo de replicação *eager* (ansiosa ou síncrona), mesmo que a consistência seja garantida, a latência de atualização é um problema. Em um tipo de replicação *lazy* (preguiçosa ou assíncrona), a necessidade de um *refresh*, baseado em tempo de atualizações, cria uma sobrecarga e a consistência depende do intervalo de *refreshs* (HAILE et al., 2017).

Kamal e Kumar (2017) descrevem que um *replica set* é um conjunto de servidores que trabalham juntos para fornecer a alta disponibilidade, empregando a replicação no MongoDB. Assim, se um nó primário estiver inativo, um dos nós secundários se tornará o novo primário. O nó secundário inicia uma eleição entre os outros nós secundários se não puder alcançar o nó primário. As réplicas podem ser usadas para descarregar leituras e gravações. No descarregamento de leitura, os nós secundários executarão a operação de leitura. Como a replicação é assíncrona e sempre há um intervalo de tempo entre uma solicitação de gravação em execução no nó primário e a solicitação de leitura em execução no servidor secundário, os dados podem ficar inconsistentes. O atraso de replicação é o atraso de tempo entre a última transação confirmada no primário e a última transação

aplicada no secundário. Em um conjunto de réplicas em execução, todo o secundário segue de perto as alterações no primário, buscando cada grupo de operações de seu *oplog* e reproduzindo-as aproximadamente na mesma velocidade em que ocorrem. As leituras de qualquer nó são razoavelmente consistentes e se o primário atual ficar indisponível, o secundário que assumir a função *primary* poderá servir aos clientes um conjunto de dados quase idêntico ao original. Quando o atraso é muito grande, portanto, se o aplicativo estiver configurado para rotear a consulta de leitura para o secundário, ele não obterá os dados mais recentes por causa do atraso. O nó secundário deve ter largura de banda de rede suficiente para poder recuperar o *oplog* do primário na taxa razoável, e também a vazão de armazenamento suficiente para aplicar o *oplog*.

3.1.3 SDN

Na seção 2.6 dos trabalhos relacionados (SDNaliado a sistemas distribuídos), são discutidos alguns trabalhos que utilizam os recursos do *OpenFlow* (e seu suporte limitado à QoS) para fornecer um melhor serviço aos Sistemas de Gerenciamento de Banco de Dados (SGBDs). Todavia, é necessário abordar com mais detalhes o que é esse paradigma e como ele tem sido empregado na solução de problemas envolvendo os bancos de dados.

O SDN (Software Defined Networking) descreve uma estrutura de rede com alta flexibilidade (o plano de dados e o plano de controle são separados), que prover a programação de comutadores habilitados para SDN, o que reduz custos operacionais da rede, otimizando, assim, o uso de recursos por meio de algoritmos ou dados centralizados, e simplificando as atualizações de software de rede (DAS et al., 2015).

A Open Networking Foundation, ou simplesmente ONF (FOUNDATION, 2014), define a arquitetura SDN, dividindo-a em três planos principais:

- Plano de dados: é o plano inferior e consiste em dispositivos de rede, como roteadores, switches físicos/virtuais, ponto de acesso, etc. Esses dispositivos são acessíveis e gerenciados por meio de C-DPI (Controller-Data Plane Interface), pelo (s) controlador (s) SDN. Os elementos de rede e os controladores podem se comunicar através de conexões seguras, como a conexão TLS*(Transport Layer Security)*. O protocolo *OpenFlow*(HALEPLIDIS et al., 2015) é considerado o C-DPI padrão mais prevalente usado para a comunicação entre os controladores e dispositivos de plano de dados, ou seja, o *OpenFlow* é uma interface aberta para o desenvolvimento de controladores SDN;

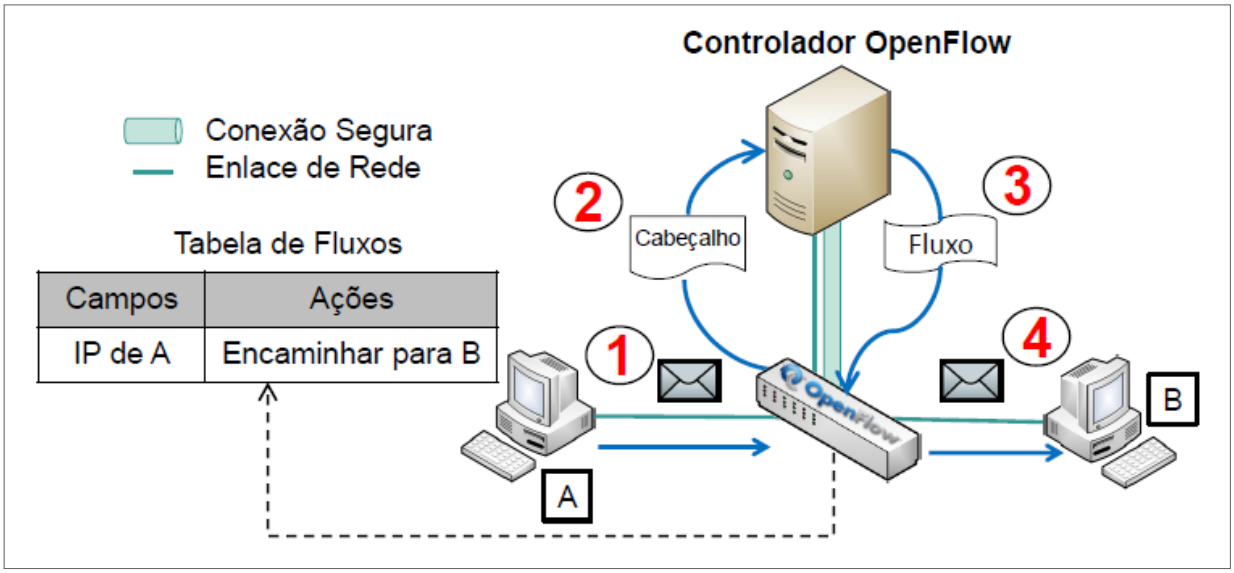
- Plano do controlador: é composto por um ou mais controladores SDN baseados em software para fornecer funcionalidade de controle supervisionando de comportamento e encaminhamento de rede por meio de um C-DPI (*OpenFlow*). Um controlador consiste em dois componentes principais: componentes funcionais e lógica de controle. Assim, os controladores podem incluir mais de um componente funcional

como, por exemplo, coordenador, virtualizador etc., para gerenciar o comportamento do controlador. Além disso, a lógica de controle mapeia requisitos de aplicações de rede em instruções para recursos de elementos de rede;

- Plano de aplicação: consiste em uma ou mais aplicações de rede, por exemplo, segurança, visualização, etc., que interagem com o (s) controlador (es) para utilizar a visão abstrata da rede para seus processos internos de tomada de decisão. Essas aplicações se comunicam com o (s) controlador (es) através de um A-C-DPI aberto, por exemplo, a *API REST*.

A Figura 1 exemplifica como se dá, basicamente, o funcionamento do paradigma SDN, em que é demonstrada a chegada de pacotes que o switch não reconhece e como o controlador SDN trata tal pacote.

Figura 1 – Funcionamento do paradigma SDN

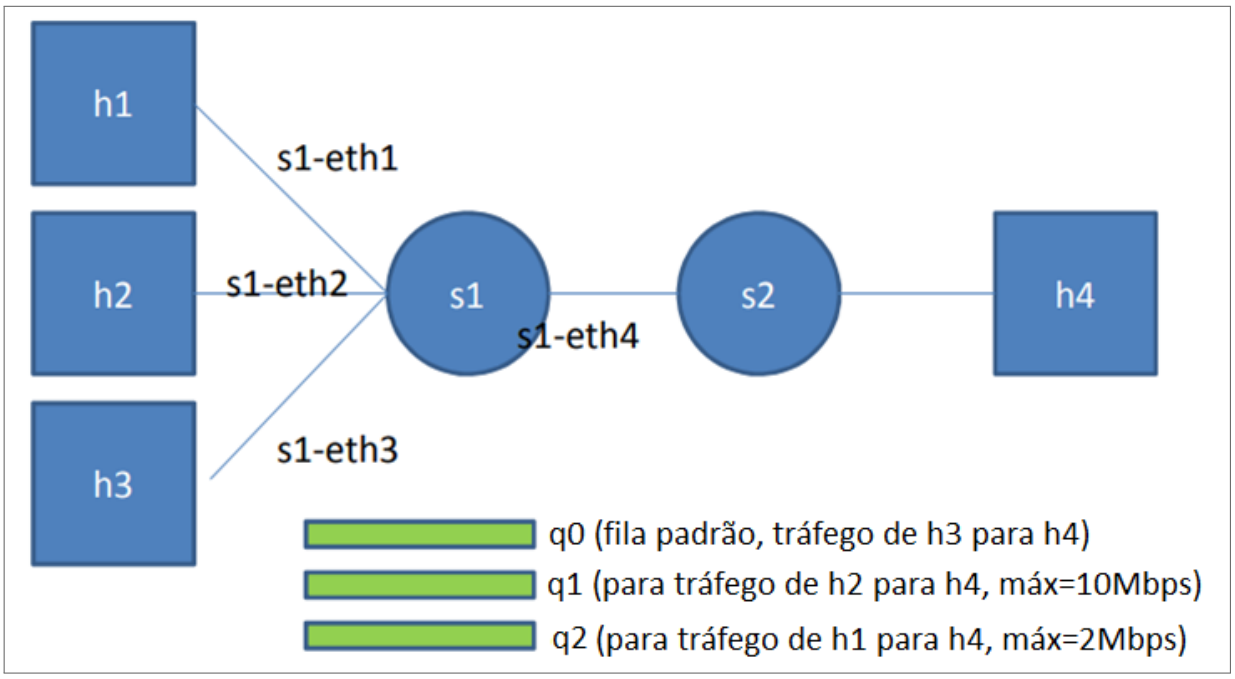


Pacotes sem fluxo definido (1) têm seu cabeçalho enviado para o controlador (2), que configura o novo fluxo na Tabela de Fluxos do comutador (3). Pacotes seguintes são encaminhados normalmente (4).**Fonte:** adaptado de Costa et al. (2014).

Os controladores *OpenFlow*, tais como Ryu (RYU, 2017), Opendaylight (OPENDAYLIGHT, 2018), floodlight(OPENDAYLIGHT, 2019), ONOS(ONOS, 2019), dentre outros, podem, ao usar o *OpenFlow* - que é patrocinado pelo Google, Microsoft e Facebook - inserir entradas de fluxo em switches habilitados para *OpenFlow* para controlar as regras de encaminhamento de todos os switches em uma rede de *data center*. Para tal, os administradores de rede podem instalar algoritmos de otimização de caminho, como o protocolo *OpenFlow*, a fim de controlar os caminhos dos fluxos novos e alterar os caminhos dos fluxos durante suas transmissões, ou, em outros casos, fornecer um suporte para QoS por meio de um simples mecanismo de enfileiramento, isto é, uma ou mais filas podem ser anexadas a uma porta e serem usadas para mapear fluxos nela. Desse modo, os fluxos

mapeados para uma fila específica serão tratados de acordo com a configuração QoS dessa fila, por exemplo, a taxa mínima (NARAYAN; BAILEY; DAGA, 2012). Assim, tais recursos provenientes do paradigma SDN podem ser úteis para resolver problemas de congestionamentos possivelmente oriundos da replicação de dados de SGBDs NoSQL somado às aplicações concorrentes à largura de banda disponível.

Figura 2 – Exemplo de QoS com Openflow (mecanismo de enfileiramento)



Fonte: própria autoria.

A Figura 2 representa o suporte para QoS a partir do mecanismo de enfileiramento com *OpenFlow*, em que se exemplifica os fluxos mapeados para diferentes filas com diferentes configurações QoS (vazão padrão e máxima entre determinados *hosts*) para essas filas.

3.2 SGBD NOSQL

Para suportar aplicações *big data*, nas quais são grandes os fluxos de dados e as necessidades de escalabilidades dos servidores, surgiram os bancos de dados não relacionais, chamados NoSQL*(Not Only SQL)*, que se apresentam como uma solução otimizada, em que os bancos relacionais têm grande deficiência.

Os sistemas tradicionais de gerenciamento de banco de dados (SGBDRs) provaram ser confiáveis para uma ampla variedade de aplicativos baseados em transações em dados estruturados. Em Hendawi et al. (2018), consta um estudo que analisa a participação de mercado dos SGBDs, que foi avaliada em US $ 35,9 bilhões, em 2015, mostrando que os principais fornecedores de mecanismos de banco de dados perderam entre 1,5% e 5,6% de suas ações nos últimos cinco anos. Por outro lado, há um aumento na participação de mercado do sistema de banco de dados não relacional (armazenamentos de dados NoSQL)

durante o mesmo período de tempo. O mercado NoSQL alcançou mais de meio bilhão de dólares em 2015 e se espera que ultrapasse US $ 3,5 bilhões em 2019.

Os bancos NoSQL seguem a propriedade chamada BASE (*Basically Available*, *Soft-State*, *Eventual consistency*), que é o oposto do ACID. Enquanto o ACID define que, ao final de cada operação, o banco deve manter sua consistência, o BASE aceita que a consistência do banco de dados esteja em estado de fluxo, ou seja, tolera inconsistências temporárias para priorizar a disponibilidade (SOUZA; SANTOS, 2015). A utilização do modelo BASE possibilita, aos SGBDs NoSQL, uma flexibilidade que atende às demandas dos ambientes distribuídos, pois a *Basically Available* (disponibilidade básica) indica que o sistema deve estar disponível na maior parte do tempo, mas que subsistemas podem estar temporariamente inoperantes. O *Soft-State* (estado leve e consistente) estabelece que os desenvolvedores devem criar mecanismos para consistências de dados e a *Eventual consistency* (consistência eventual) garante que, se determinado registro não sofrer atualizações, eventualmente, estará consistente e todas as leituras retornarão à última atualização do valor.

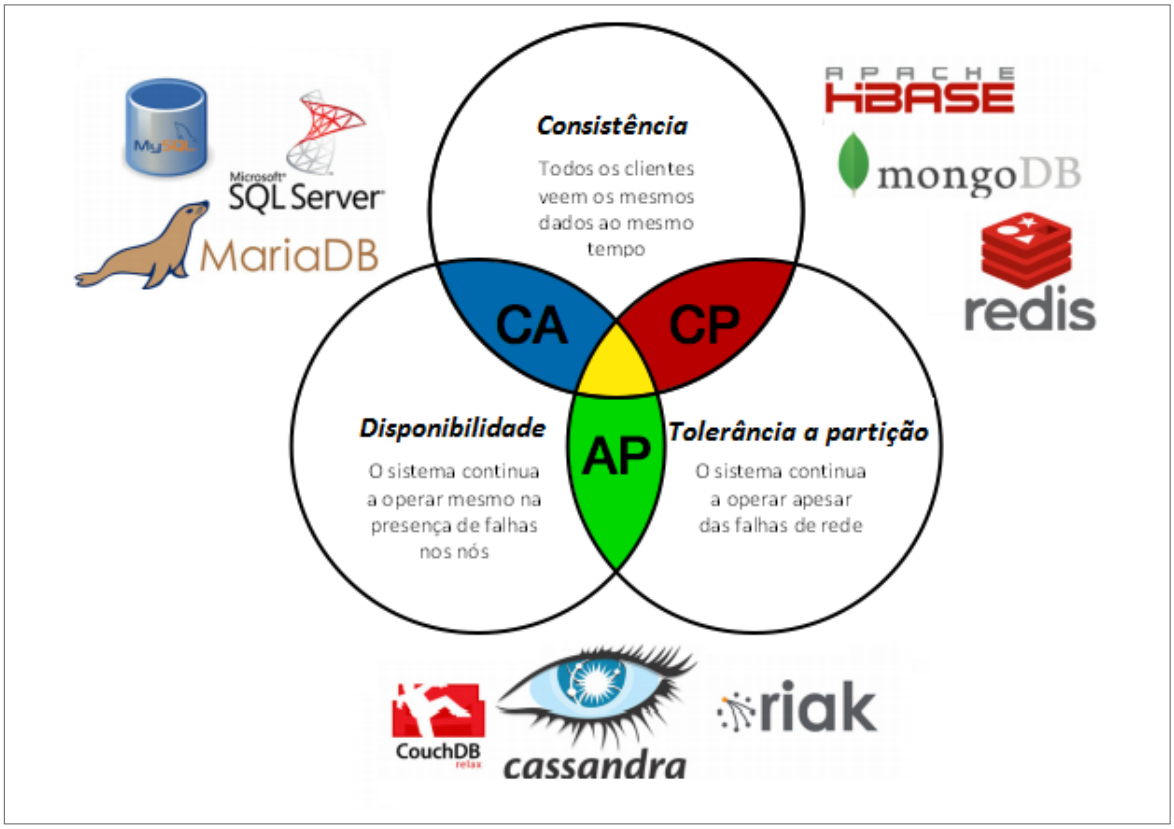
Outra característica importante dos bancos NoSQL é o teorema CAP (GILBERT; LYNCH, 2002), que é a base da escalabilidade desses sistemas. CAP é o acrônimo para *Concistency* (Consistência), *Availability* (Disponibilidade) e *Tolerance of Network Partition* (tolerância ao particionamento de rede). Esse teorema descreve que uma aplicação distribuída não pode ter mais que duas propriedades simultaneamente. No entanto, há pesquisas, como a de Souza e Santos (2015), que salientam sobre as mudanças no cenário atual, como os sistemas em que o particionamento está presente e os projetistas devem fazer escolhas entre disponibilidade e consistência, sendo que umas das propriedades não precisa ser necessariamente negligenciada, mas, sim, apresentar uma combinação entre elas.

Diferente do modelo relacional, os bancos de dados NoSQL possuem diversos modelos de dados que apresentam características que favorecem diferentes tipos de aplicações, que podem ser categorizadas em chave/valor, orientado a coluna, orientado a documento e orientado a grafos.

Na perspectiva do API, os bancos chave/valor são os modelos de armazenamento de dados NoSQL mais simples de se usar, o DynamoDb e o Redis são exemplos desses modelos. Os Sistemas orientados a documentos armazenam e recuperam documentos, como o XML, o JSON, o BJON, dentre outros, e fornecem uma rica linguagem de consulta – dessa categoria, os bancos mais populares são o CouchDB e o MongoDB. Com o armazenamento de coluna, os dados são armazenados como linhas em que possuem muitas colunas associadas com uma chave de linha – nesse modelo de armazenamento, destacam-se o BigTable e Cassanda. Nos modelos de grafos são permitidos armazenar entidades e relacionamentos entre essas entidades – os bancos Neo4j e Titan são grandes bancos que empregam esse tipo de modelo.

Sistemas relacionais distribuídos geralmente executam a replicação mestre-escravo para manter uma consistência forte. A atualização antecipada em qualquer lugar da replicação - como por exemplo, na Megastore do Google - sofre de uma sobrecarga de comunicação pesada gerada pela sincronização e pode causar *deadlocks* distribuídos, que são caros de detectar. Os sistemas de banco de dados NoSQL normalmente dependem de replicação lenta, seja em combinação com o mestre-escravo (sistemas CP, por exemplo, HBase e MongoDB) ou na abordagem de atualização em qualquer lugar (sistemas AP, por exemplo, Dynamo e Cassandra). Muitos sistemas NoSQL deixam a escolha entre latência e consistência para o cliente, ou seja, para cada solicitação, o cliente decide se espera por uma resposta de qualquer réplica para obter latência mínima ou por uma resposta certamente consistente (pela maioria das réplicas ou mestre) para evitar dados obsoletos (GESSERT et al., 2017). A figura a seguir (Figura 3) trás uma boa noção da divisão e principais características dos mais importantes sistemas de armazenamento com base no teorema CAP.

Figura 3 – Divisão e principais características dos sistemas de armazenamento de acordo com teorema CAP



Fonte: adaptado de Lourenço et al. (2015).

Os bancos de dados NoSQL, especialmente MongoDB, ocupam cada vez mais espaço no mercado (ASAY, 2019). Além disso, o MongoDB, Cassandra e o Redis estão, constantemente, presentes no Top 10 dos bancos mais populares, segundo o DB-Engines (2019). Levando tais fatores em consideração, o MongoDB foi o banco escolhido para realiza-

ção de experimentos aqui realizados. Dessa forma, a seguir, nas subseções 3.2.1 e 3.2.2, descrevemos algumas características especificas do MongoDB.

3.2.1 MongoDB

MongoDB é um sistema de armazenamento orientado a documento NoSQL que não tem esquemas tão rigorosos quanto aos SGBDRs, o que permite crescer horizontalmente, sem degradação do desempenho, Além disso, o MongoDB traz possibilidades para diferentes cenários de armazenamento e permite que os programadores usem o banco de dados como um armazenamento que se adapta as suas necessidades e não o contrário. Ele foi desenvolvido pela 10gen em 2007 e é escrito na linguagem de programação C ++(TABET; MOKADEM; LAOUAR, 2018).

O MongoDB é um banco de dados de documentos, de código aberto, que fornece alto desempenho, alta disponibilidade e dimensionamento automático. A organização de seus dados segue a seguinte hierarquia: banco de dados, coleção e documento, sendo que um banco de dados é um conjunto de coleções e uma coleção é um conjunto de documentos. Coleções são análogas às tabelas em bancos de dados relacionais, porém, diferentemente de uma tabela, uma coleção não exige que seus documentos tenham o mesmo esquema. Um registro no MongoDB é um documento, que é uma estrutura de dados composta de pares de campos e valores (MAHAJAN; BLAKENEY; ZONG, 2019).

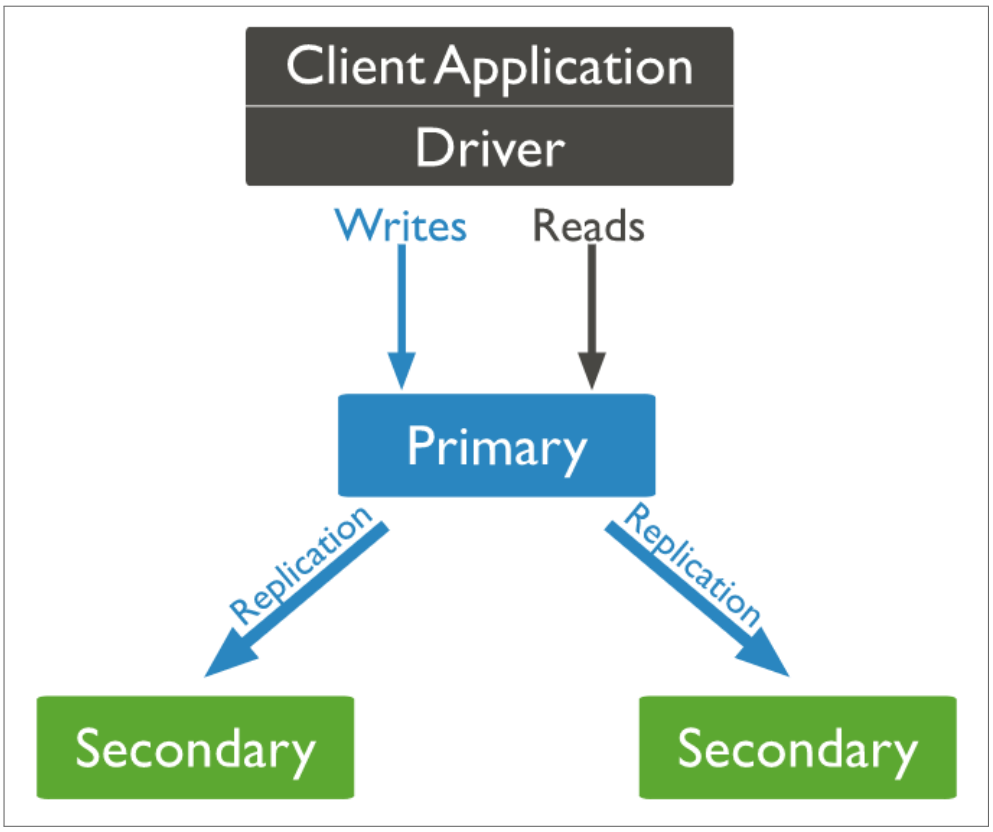
Uma implantação do MongoDB consiste em três tipos de servidores: os *mongod* – que são os servidores que armazenam os próprios pedaços de dados, e, normalmente, são agrupados em um *replica set* (conjuntos de réplicas), em que cada conjunto de réplicas contêm o mesmo número de servidores (geralmente, 3 servidores), com réplicas exatas uns dos outros: um dos servidores marcado como primário (mestre) e os outros dois atuando como secundários (escravos); *config servers* (servidores de configuração) – em que os parâmetros de configuração do banco de dados são armazenados e os clientes enviam consultas CRUD (Criar, Ler, Atualizar, Excluir) para um servidor *front-end*, chamado *mongos*, o qual representa, exatamente, o terceiro dos tipos de servidores – que também armazena algumas das informações de configuração dos *config servers*, como, por exemplo, o roteamento das consultas (GHOSH et al., 2015).

3.2.2 Replicação de dados e Consistência no MongoDB

Lidando com a replicação de dados no MongoDB, dois modelos de replicação são possíveis: o primeiro é o modelo Master/Slave (Mestre/Escravo) e o segundo é definido como modelo *replica set* (conjunto de réplicas). A principal diferença entre os dois modelos de replicação é que um *replica set* tem a capacidade de realizar *failover* automaticamente quando o nó primário não está disponível, elegendo um novo nó primário dos nós secundários existentes com base no voto dos nós situados no mesmo *cluster*; enquanto que, no modelo de replicação mestre-escravo, todas as réplicas são apenas de leitura, sendo

atualizadas somente a partir do nó mestre após as alterações, o que pode causar problemas, pois quando o nó mestre falha, não há possibilidade de gravar novos dados. Assim, o modelo de replicação mais adequado é o *replica set* (TABET; MOKADEM; LAOUAR, 2018). Abaixo, a Figura 4 demonstra como se dá replicação no modelo *replica set*.

Figura 4 – Replicação de dados do MongoDB.



Fonte: Mongo (2019).

Um *Replica set* consiste de várias instâncias (*mongod*), incluindo um nó árbitro (que não replica dados e só existe para quebrar vínculos ao eleger um novo nó primário quando necessário), um nó mestre atuando como primário e vários escravos atuando como secundários que mantêm o mesmo conjunto de dados. Se o nó mestre travar, o nó do árbitro selecionará um novo mestre a partir do conjunto de escravos remanescentes. Todas as operações de gravação devem ser direcionadas para uma única instância primária. Por padrão, os clientes enviam todas as solicitações de leitura para o mestre, no entanto, uma preferência de leitura é configurável no nível do cliente, o que torna possível enviar solicitações de leitura para os nós escravos. As preferências de leituras variadas oferecem diferentes níveis de garantias de consistência (HAUGHIAN; OSMAN; KNOTTENBELT, 2016).

A replicação do MongoDB opera por meio de um *oplog*, no qual o nó mestre registra todas as alterações em seus conjuntos de dados. Os nós escravos, então, replicam o *oplog* do mestre, aplicando essas operações a seus conjuntos de dados. Esse processo de replicação é assíncrono, portanto, nós escravos nem sempre refletem os dados mais atualizados.

Write Concern (referência ou preocupação de gravação) pode ser definidas por operação de gravação para determinar o número de nós que devem processar uma operação de gravação antes de retornar ao cliente com êxito. Isso permite configurações de consistência ajustáveis e refinadas, incluindo *Quorum* e gravações totalmente consistentes. Além disso, permite leituras simultâneas em uma coleção, mas impõe um único mecanismo de bloqueio encadeado em todas as operações de gravação para garantir a atomicidade. Contudo, todas as operações de gravação precisam ser anexadas ao *oplog* (log de operações) em disco, o que envolve maior sobrecarga, mas, por outro lado, independentemente da preocupação de leitura solicitada, nenhuma verificação de consistência adicional é executada entre as réplicas nas operações de leitura (MONGO, 2019).

O parâmetro *write concern (w)* indica a quantidade de servidores para onde a operação (inclusão, exclusão, alteração) deve ser propagada, para que, assim, o MongoDB considere a operação bem sucedida. Por padrão, esse parâmetro tem o valor 1 no MongoDB, o que indica que os dados são considerados persistidos após a inclusão no servidor *master*. O MongoDB também permite que seja usada a opção *majority*, que indica que a operação será bem sucedida caso tenha sido efetuada na maioria dos servidores (deve-se levar em conta que quanto maior o *write concern* menor poderá ser o desempenho do banco de dados). Outros parâmetros podem ser combinados a *write concern*: *Wtimeout*, tempo (em milissegundos), que pode ser estipulado para operação, se w for diferente de zero; e J *(Journaling)*, um booleano, que, quando possui o valor *true*, indica que a operação só será bem sucedida após a quantidade de servidores informados no parâmetro w gravarem a operação em disco (MONGO, 2019).

No manual do mongo, indica-se que a implementação do conjunto de réplicas padrão para o sistema é um *replica set* de três membros, visto que esse conjunto já fornece redundância e tolerância a falhas. No entanto, algumas restrições devem ser consideradas, por exemplo, um conjunto de réplicas pode ter até 50 membros, mas apenas 7 membros votantes. Em um conjunto de réplicas, membros podem se tornar indisponíveis, portanto, deve-se observar quantos membros restaram para eleger um nó primário (ver a Tabela 1).

Tabela 1 – Efeito do tamanho do conjunto de réplicas na tolerância a falhas

Número de membros	Maioria necessária para eleger um novo primário	Tolerância ao erro
3	2	1
4	3	1
5	3	2
6	4	2

Fonte: Mongo (2019).

Considera-se que os níveis mais significativos de consistência são: Consistência estrita, Consistência causal, Consistência monotônica, Consistência Eventual e Consistência fraca. Por padrão, em uma implantação de servidor único, um banco de dados MongoDB fornece consistência estrita de documento único. Dessa forma, quando um documento do MongoDB está sendo modificado, ele é bloqueado contra leituras e gravações de outras sessões. No entanto, quando os conjuntos de réplica do MongoDB são implementados, é possível configurar algo mais próximo da consistência eventual, permitindo que as leituras sejam concluídas em servidores secundários que podem conter dados desatualizados (HARRISON, 2015, capítulo 9).

Além da consistência de leituras e gravações, em sistemas de armazenamento distribuído, surge o conceito de durabilidade, que é a capacidade de um determinado sistema de persistir dados, mesmo na presença de falhas. Isso faz com que os dados sejam gravados em vários dispositivos de memória não volátil antes de informar o sucesso de uma operação a um cliente. Em sistemas eventualmente consistentes, existem mecanismos para calibrar a durabilidade e a consistência do sistema (CORBELLINI et al., 2017).

A Tabela 2 mostra diferentes configurações de W e R, bem como o resultado da aplicação de tais configurações. Nesse contexto, cada valor refere-se ao número de réplicas necessárias para confirmar o sucesso de uma operação, assim: N é o número de nós em que uma chave é replicada, W é o número de nós necessários para considerar uma escrita como bem-sucedida e R é o número de nós em que uma leitura é executada (CORBELLINI et al., 2017).

Tabela 2 – Configurações de consistência eventual

Valor	Escrita(W)	Leitura (R)
0	Nenhuma confirmação é esperada de qualquer nó (pode falhar)	N/D
1	Uma confirmação de nó único é suficiente (otimizada para escritas)	A leitura é realizada a partir de uma única réplica (otimizada para leituras)
M, com $M < N$ (Quorum)	Confirmações de várias réplicas são aguardadas	A leitura é realizada a partir de um determinado conjunto de réplicas (os conflitos no lado do cliente podem precisar ser resolvidos)
N (todos os nós)	Confirmações de todas as réplicas são esperadas (reduz a disponibilidade, mas aumenta a durabilidade)	A leitura é realizada a partir de todas as réplicas, aumentando a latência de leitura

Fonte: adaptado de Corbellini et al. (2017).

Em Corbellini et al. (2017), os autores descrevem que a consistência forte é alcançada

cumprindo $W + R > N$, ou seja, o conjunto de escritas e leituras sobrepõem-se de tal forma que uma das leituras sempre obtém a versão mais recente de um dado. Normalmente, os SGBDRs têm $W = N$, ou seja, todas as réplicas são persistentes e $R = 1$, pois qualquer leitura retornará os dados atualizados. Consistência fraca ocorre quando $W + R \leq N$, em que as leituras podem obter dados desatualizados. A consistência eventual é um caso especial de consistência fraca, na qual há garantias de que, se um dado for gravado no sistema, ele eventualmente atingirá todas as réplicas. Isso dependerá da latência da rede, da quantidade de réplicas e da carga do sistema, dentre outros fatores.

Vale lembrar que os valores da Tabela 2, no contexto de configurações do mongo, corresponde ao parâmetro w, em que 0 e 1 têm o mesmo principio (sendo 1 o padrão), ao *Quorum*, que equivale a $w = majority$, e ao N (todos nós), que se expressa como $w =$ *total de nós do replica set*. Com relação à leitura, o mongo permite configurar, com mais especificidade, o parâmetro R, como sendo *ReadConcern*, que pode assumir as seguintes atribuições: *primary* (modo padrão, todas as operações são lidas a partir do nó principal), *primaryPreferred* (as operações são lidas a partir dos nós principais, mas, se estiverem indisponíveis, as operações são lidas a partir de membros secundários), *secondary* (todas as operações serão lidas nos membros secundários do conjunto de réplicas), *secondaryPreferred* (as operações são lidas a partir de membros secundários, mas, se nenhum membro secundário estiver disponível, as operações são lidas a partir do primário) e *nearest* (as operações serão lidas nos membros do conjunto de réplicas com a menor latência de rede, independentemente do tipo de membro).

3.3 AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho é uma técnica bastante aplicada no campo de pesquisa para comprovar a evolução de novos produtos e serviços utilizados de forma comparativa, e está aliada às ferramentas de estatísticas para expressarem, de forma clara e científica, os resultados (MONTGOMERY, 2017). Tal pratica pode ser realizada por meio de medições no sistema real ou a partir de modelos que representam as principais características e comportamentos de um sistema.

De forma geral, um sistema computacional recebe requisições (pedidos, *Jobs*, pacotes etc.) dos clientes (usuários, entidades, softwares etc.) para realizar alguma ação, isto é, realizar algum tipo de serviço para o usuário. A avaliação de desempenho dos sistemas computacionais deverá considerar os tipos de requisições recebidas, os tipos de clientes e os tipos de serviços a serem realizados. Basicamente, existem três técnicas de avaliação de desempenho: modelagem, simulação e medição. Cada técnica possui vantagens e desvantagens, e devem ser escolhidas segundo alguns critérios, mas, na prática, o avaliador usa a técnica com a qual se sente mais familiarizado (JOHNSON; COUTINHO, 2000).

Tendo em vista que o desempenho é um critério fundamental para a concepção, aquisição ou uso de sistemas computacionais, faz-se necessário definir métricas, fatores e/ou

níveis para quantificar o serviço ou sistema. Por exemplo, a taxa de transferência (vazão) é uma métrica frequentemente utilizada para avaliar o desempenho em uma rede de computadores. De acordo com alguns trabalhos, como Pande et al. (2005), a vazão significa o valor máximo do tráfego aceito, que está relacionada à taxa de dados de pico sustentável pelo sistema. A medição da vazão da rede envolve o envio de um arquivo de tamanho médio por meio do canal de comunicação e da medição do tempo necessário para transmiti-lo. Assim, dividir o tamanho do arquivo pelo tempo de transmissão fornece a medida da vazão (ZAFAR; TARIQ; MANZOOR, 2016).

Um estudo de avaliação de desempenho pode ser discutido em 10 etapas: i) estudar o sistema e definir os objetivos; ii) determinar os serviços oferecidos pelo sistema; iii) selecionar métricas de avaliação; iv) determinar os parâmetros que afetam o desempenho do sistema; v) determinar o nível de detalhamento da análise; vi) determinar a técnica de avaliação apropriada; vii) determinar a carga de trabalho característica; viii) realizar a avaliação e obter os resultados; ix) analisar e interpretar os resultados; e x) apresentar os resultados. Tais passos podem ser agrupados em: Planejamento de Experimentos (de i a v), Técnica de Avaliação (vi) e Análise dos Resultados (ix e x) (JAIN, 1991).

3.3.1 Planejamento de Experimento

O Planejamento de Experimento é um método utilizado para avaliar o impacto de cada parâmetro de um determinado sistema, em que o objetivo é obter o máximo de informações com o mínimo de experimentos, reduzindo o trabalho com a coleta dos dados, além de facilitar a separação dos efeitos de vários fatores que podem influenciar o desempenho(JAIN, 1991).

Para projetar um experimento é necessário definir fatores, níveis e uma variável de resposta. Através dessa técnica, é possível averiguar a influência dos fatores sobre a variável de resposta, e, para analisar os efeitos de cada fator na variável de resposta, é necessário obter valores que representam os níveis desses fatores (JAIN, 1991). Com esse método, é possível avaliar a importância de cada um dos parâmetros (fatores) do sistema e, além disso, pode ser usado para determinar simultaneamente os efeitos individuais e interativos de fatores que podem afetar as medidas de saída (MATHEWS, 2005).

Existem vários tipos de planejamentos de experimento e os mais usados são: planejamento simples, planejamento fatorial completo e planejamento fatorial fracionário (JAIN, 1991):

- O planejamento fracionário economiza tempo e dinheiro, mas fornece menos informações que o método completo. Por exemplo, pode-se obter algumas, mas não todas as interações entre os fatores. Fatoriais fracionários são comumente usados para reduzir o número de execuções necessárias para construir um experimento. Eles são

recomendados para os casos em que o experimento possui 2^k modelos com cinco ou mais variáveis (MATHEWS, 2005; JURISTO; MORENO, 2013);

- No planejamento simples, inicia-se com uma configuração típica e varia-se um fator por vez, para ver como esse fator afeta o desempenho. No entanto, esse método não faz o melhor uso do esforço gasto. Não é estatisticamente eficiente. Além disso, se os fatores tiverem interação, esse método pode levar a conclusões erradas (JAIN, 1991);

- Planejamento fatorial completo exige muitos experimentos, pois todas as combinações possíveis de configuração e carga de trabalho são examinadas. Considerando um sistema, no qual o seu desempenho é afetado por k fatores, e cada fator tem N níveis possíveis de valores, e o número de experimentos a serem executados é calculado por $\mathrm{N}^k(MATHEWS, 2005; JURISTO; MORENO, 2013)$.

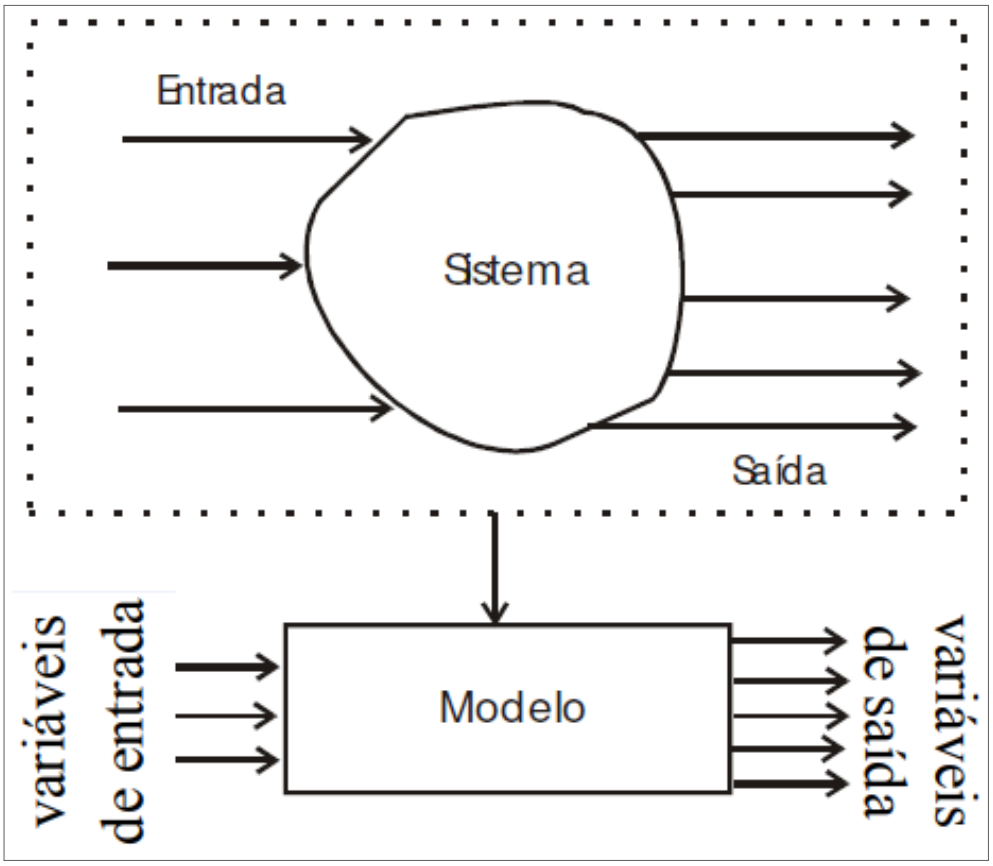
3.3.2 Técnicas de Avaliação

As três técnicas abrangentes para avaliação de desempenho são modelagem analítica, simulação e medição de um sistema real. A seleção da técnica correta depende do tempo e dos recursos disponíveis para resolver o problema e do nível de precisão desejado (JAIN, 1991).

A escolha da técnica de avaliação depende muito do status de implementação/construção do sistema a ser avaliado. Para o design de um novo sistema é possível usar simulação ou modelagem(JOHNSON; COUTINHO, 2000). O uso de modelos ou simuladores nos permite a avaliação de diferentes cenários e a comparação entre eles. Em um modelo, é possível criar diversas situações de uso: execuções de diferentes cargas de trabalho, variação dos tempos de processamento de transações etc., dando ao avaliador diversas opções para modificar e, então, selecionar a configuração mais vantajosa. No caso da medição, primeiro é necessário já ter o sistema a ser medido ou algum protótipo. Assim, esse protótipo pode ser usado nas fases de avaliação para a verificação da melhor configuração, antes que o produto esteja em sua fase final de implementação/construção. O custo de realizar diversas modificações em um sistema/protótipo será menor do que modificar um produto finalizado. Com a medição também é possível comparar um sistema existente com as suas futuras versões, sendo possível confirmar se as novas funcionalidades ou modificações levam a um ganho de desempenho e quantificar esse ganho. Além disso, os modelos de medição também podem ser usados como entrada de um modelo de simulação ou de modelagem, gerando assim, um modelo híbrido.

Um modelo pode ser considerado como uma representação de um ou mais pontos de vistas de um sistema em um determinado nível de abstração. Tal modelo deve ser representado através de um conjunto de variáveis, funções e relações matemáticas que simbolizam uma ou mais perspectivas do sistema. Ou seja, enquanto um sistema é “algo real”, o modelo é uma “abstração”, como mostra a Figura 5.

Figura 5 – Processo de modelagem simples



Fonte: adaptado de Cassandras e Lafortune (2009).

Diversas técnicas de modelagem matemática de sistemas, em várias áreas da ciência, têm sido propostas para representar sistemas computacionais, dentre elas:

- **Técnicas Baseadas em estados:** fornecem uma descrição abstrata e explícita sobre estados e operações que transformam os estados, no entanto, não oferecem meios explícitos para especificar concorrência;
- **Técnicas Baseadas em Álgebra de Processos:** essas técnicas possibilitam meios explícitos para especificar concorrência. O comportamento dos processos é representado através de ações observáveis;
- **Técnicas Baseadas em Lógica:** uma grande variedade de técnicas baseadas em lógica tem sido levantadas, nas quais se analisam as relações causais e os aspectos relacionados à temporização;
- **Técnicas Baseadas em Redes:** essas técnicas modelam concorrência, por meio de mecanismos implícitos de fluxo de *tokens* na rede. Esse fluxo é controlado por condições que habilitam a realização de tarefas (eventos). Ex.: Redes de Petri.

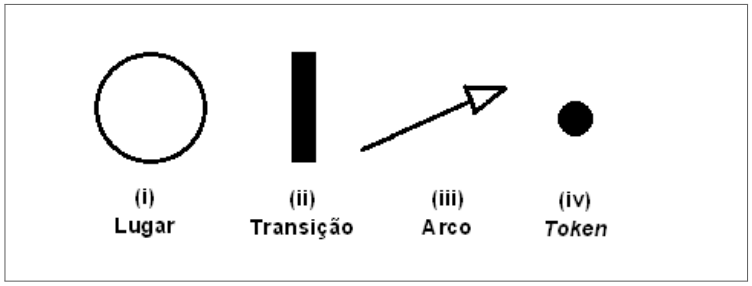
Descrições mais detalhadas dessas técnicas e modelos, citados anteriormente, podem ser encontradas em Cassandras e Lafortune (2009). No entanto, focando nos modelos de avaliação, que serão utilizados neste trabalho, é interessante notar que esse tipo de modelagem permite, dentre outros fatores, representar relações mais complexas entre os componentes do sistema, como dependências que envolvem subsistemas e restrições de recursos. Além disso, o recurso da modelagem é simples e evita custos adicionais, como o tempo de configuração em N equipamentos e/ou os custos de aquisição desses equipamentos.

3.3.3 Rede de Petri

Redes de Petri é um termo genérico utilizado para denotar um conjunto de formalismos com uma interpretação gráfica empregada para representar o comportamento de sistemas de eventos discretos. É uma ferramenta usada para descrever e estudar sistemas de processamento caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos. Seu uso permite que desenvolvedores e administradores mantenham a eficiência e economizem tempo e trabalho experimental para tomada de decisões. Assim, o uso da família de formalismos de Redes de Petri (PN) tornou-se comum na especificação de sistemas.

As Redes de Petri básicas (Lugar/Transição ou PN) são, em sua descrição mais simples, grafos bipartidos com dois tipos de vértices: lugares e transições, que são conectados por arcos direcionais (observar a Figura 6).

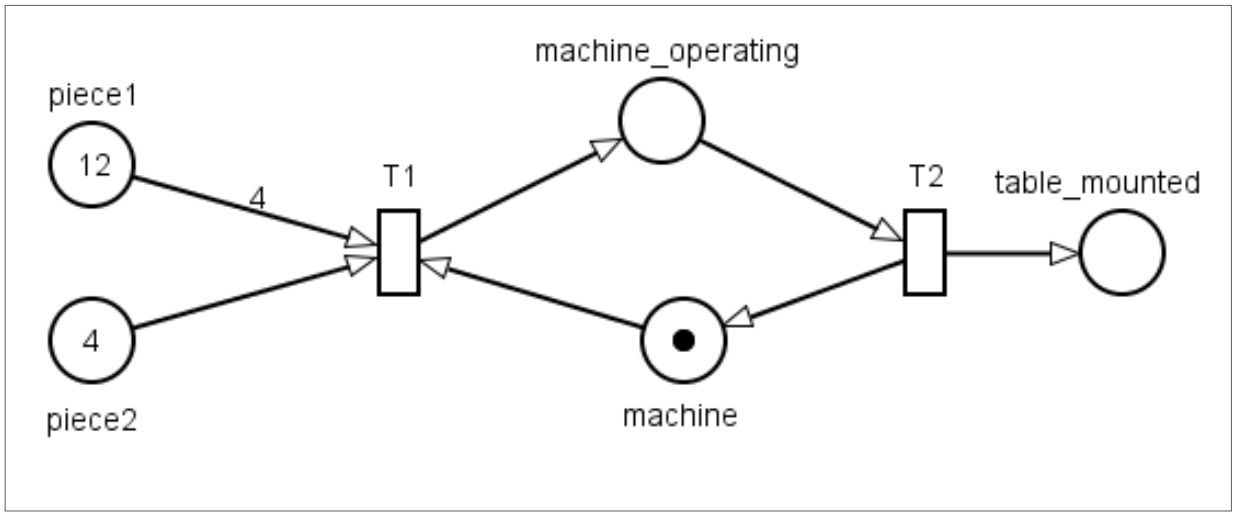
Figura 6 – Elementos de uma Rede de Petri



Fonte: adaptado de Maciel, Lins e Cunha (1996).

A definição formal de Redes de Petri pode ser da seguinte forma: 5-tuple N = (P, T, F, W, M_0), em que P e T são conjuntos finitos disjuntos de lugares e transições, respectivamente; F$\rightarrow \subseteq$ (P x T) $\cup$ (T x P) é um conjunto de arcos (ou relações de fluxo entre a matriz de entrada, que representa as pré-condições, e a saída matriz, que representa as pós-condições); W: F$\rightarrow$ {0,1,2,3,...} é uma função de peso; e $M_0 : P \rightarrow$ {0,1,2,3,...} é a marcação inicial (MURATA, 1989). Um exemplo é demonstrado a seguir, na Figura 7, em que a rede representa a montagem de uma mesa.

Figura 7 – Exemplo de uma Rede de Petri



Rede de Petri com peso 4 no arco atendendo a necessidade de 4 pernas para a montagem de uma mesa simples, o modelo possui marcação inicial $M_0 =$ (12,4,1,0,0).**Fonte:** própria autoria.

As PNs podem ser usadas como um auxílio de comunicação visual semelhante a fluxogramas, diagramas de bloco e redes. Além disso, os *tokens* são usados nessas redes para simular as atividades dinâmicas e simultâneas dos sistemas. Como uma ferramenta matemática, é possível configurar equações de estado, equações algébricas e outros modelos matemáticos que governam o comportamento dos sistemas. Desde o trabalho seminal de Petri, muitas representações e extensões foram propostas, permitindo descrições mais concisas e representando características de sistemas não observadas nos primeiros modelos (MURATA, 1989).

3.3.4 Redes de Petri Estocásticas Generalizadas (GSPN)

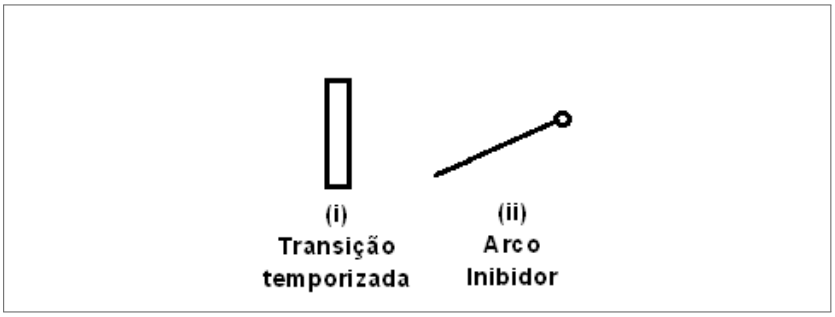
O uso de formalismos, como as Redes de Petri (PN), tornou-se comum na modelagem, na análise e na simulação de sistemas. Os modelos, bem como as suas extensões (modelos GSPN), genericamente denominados SPN, representam a união da teoria das Redes de Petri e o processo de modelagem estocástica (LIMA et al., 2014). As SPN são casos especiais de PN. Os modelos SPN foram propostos com o objetivo de desenvolver uma ferramenta que permitisse a integração de descrição formal, a comprovação de correção e a avaliação de sistemas (SILVA et al., 2017).

Uma Rede de Petri Estocástica SPN é uma extensão da Rede de Petri original. Uma SPN é formada ao se associar uma taxa de disparo l a cada transição da rede. Uma vez habilitada a transição *ti*, o intervalo de tempo até o seu disparo é exponencialmente distribuído com média *1/li*. Um modelo de SPN permite o cálculo das probabilidades estacionárias de todas as marcações alcançáveis, ou seja, de todos os estados do sistema. A partir dessas probabilidades, pode-se obter as medidas de desempenho (ALMEIDA, 1990).

Uma Rede de Petri Estocástica Generalizada GSPN é uma extensão das SPNs, em que cada transição pode disparar com um tempo nulo ou com um tempo exponencialmente distribuído. Dessa forma, as transições (componentes ilustrados na Figura 8) são divididas em dois conjuntos: transições imediatas (com tempo zero) e transições temporizadas (ALMEIDA, 1990).

As GSPN ou modelos derivados da SPN dispõem, ainda, de arcos inibidores (componente demonstrado na Figura 8), que permitem testar se um lugar não possui *tokens*. Com a presença do arco, a transição estará habilitada se a quantidade de *tokens* em um lugar p associado ao arco for menor que o peso do arco n, ou seja M(p) < n (MARSAN et al., 1994).

Figura 8 – Elementos de uma GSPN



Fonte: própria autoria.

A adição de transições temporizadas introduz o conceito de habilitação múltipla, que deve ser considerado em transições temporizadas com grau de habilitação maior que um. Nesse caso, a semântica de disparo deverá levar em conta a quantidade de *tokens* que podem ser disparados paralelamente. As possibilidades de semântica são:

- Exclusive Server: a transição dispara sequencialmente, ou seja, somente após um disparo essa se torna novamente habilitada, independentemente do grau da transição;
- Infinite Server: uma única habilitação de uma transição suporta a transferência de infinitos *tokens* simultaneamente;
- Multiple Server: todo o conjunto de *tokens* será processado em paralelo até o máximo grau de paralelismo *(k)* definido para essa semântica.

As GSPN marcadas com um número finito de lugares e transições são isomórficas às cadeias de Markov. O isomorfismo de um modelo GSPN com uma cadeia de Markov é obtido a partir do gráfico de alcançabilidade reduzido, que é dado por meio da eliminação dos estados voláteis, do rótulo dos arcos com as taxas das transições temporizadas e dos pesos das transições imediatas(MARSAN et al., 1994).

A utilização de uma GSPN permite a construção de um grafo de cobertura dos estados acessíveis em casos simples em que a rede de Petri subjacente é limitada. Entretanto, o tamanho do grafo de cobertura explode rapidamente quando os intervalos de tempo são muito diferentes. O problema principal vem do fato de que a descrição do estado dessas redes deve compreender não somente a marcação, mas também as informações temporais. Para poder utilizar a potência da análise markoviana, é necessário que o sistema não tenha memória do passado, isto é, se um evento produzir um disparo da transição t e transformar a marcação M_1 em M_2, a evolução futura das transições, que estavam sensibilizadas por M_1 antes do disparo de t, deverá ser idêntica às transições sofridas caso viessem a ser sensibilizadas por M_2. Somente as distribuições geométricas e exponenciais verificam esse fato. As Redes de Petri Estocásticas, e suas extensões, são, portanto, as redes em que as durações de sensibilização associadas às transições são definidas por tais distribuições, a fim de poder construir um processo markoviano equivalente e, assim, analisar o comportamento da rede (CARDOSO; VALETTE, 1997).

Formalmente, na definição de Marsan et al. (1994), uma GSPN é uma 8-tupla GSPN $= (P, T, \prod(.), I(.), O(.), H(.), W(.), M_0)$, em que:

- P é um conjunto finito de lugares, isto é, P $= p_1, p_2, ..., p_n$;
- T é um conjunto finito de transições, ou seja,T $= t_1, t_2, ..., t_n$;
- $\prod : T \rightarrow N$ é a função de prioridade que mapeia uma transição imediata a um número natural que representa a prioridade da mesma;
- I $\subseteq$ (P X T) é um conjunto de arcos que marca relação de fluxo de entrada;
- O $\subseteq$(T X P) é um conjunto de arcos que marca relação de fluxo de saída;
- H : P X T $\rightarrow N$ é a função de mapeamento de que representam os arcos inibidores;
- W : T $\rightarrow R^+$ é a função que mapeia uma transição a uma função real positiva com imagem no conjunto dos reais;
- $M_0 : P \rightarrow$ é a marcação inicial da rede.

As medições de desempenho são obtidas a partir de simulações e de análises em estado estacionário e transiente baseados na Cadeia de Markov embutida no modelo SPN (BLOCH et al., 1998). Lembrando que é comum o uso do acrônimo SPN para representar toda a família de modelos derivados da SPN, como as GSPN, conforme Haas (2006).

Diversas propriedades podem ser obtidas a partir de modelos, permitindo, assim, revelar as mais variadas características do sistema. Essas propriedades podem ser subdivididas em comportamentais e estruturais:

- **Propriedades Comportamentais**: As propriedades comportamentais são as propriedades que dependem da marcação. A seguir, as principais propriedades comportamentais, segundo Murata (1989) e Maciel, Lins e Cunha (1996):
  - **Alcançabilidade**: a propriedade de alcançabilidade indica a possibilidade de atingir uma determinada marcação pelo disparo de um número infinito de transições a partir de uma dada marcação inicial. Uma marcação M_0 é dita alcançável a partir de M_i, se existir uma sequência de disparo que transforme M_0 em M_i. A sequência de disparo é denotada pelo conjunto $\sigma = \{ t_1; t_2; ...t_n \}$. Nesse caso, M_i é alcançável a partir de M_0 por σ, em que σ é formalmente descrito por $M_0 [\sigma > M_i$;
  - **Limitação e Safeness**: uma rede é dita *k-limitada* se todos os seus lugares forem limitados, ou seja, o número de *tokens* em cada lugar não deve ultrapassar um número infinito k, para qualquer marcação alcançável a partir de M_0. Uma rede de Petri é dita *safeness*, se k = 1;
  - **Liveness**: uma rede é dita *live* se não importam quais marcações sejam alcançáveis a partir de um marcação inicial M_0, se for possível disparar qualquer transição através do disparo de alguma sequência de transições L(M_0). O conceito de *deadlock* está fortemente conectado ao conceito de *liveness*. No entanto, o fato de um sistema ser livre de *deadlock* não resulta que esse seja *liveness*. Contudo, um sistema *liveness* implica um sistema livre de *deadlocks*. A análise de *liveness* de uma rede permite verificar se os eventos modelados ocorrem, efetivamente, durante o funcionamento do sistema, ou se foram definidos eventos mortos no modelo;
  - **Cobertura**: a propriedade de cobertura está fortemente conectada ao conceito de alcançabilidade e *liveness*, apresentados anteriormente. Quando se deseja saber se alguma marcação M_i pode ser obtida a partir de uma marcação M_j, tem-se o problema denominado cobertura de uma marcação. Uma marcação M_i é dita coberta se existe uma marcação M_j tal que $M_j \geq$M$_i$. Fora isso, em alguns sistemas, deseja-se apenas observar o comportamento de determinados lugares. Assim, restringe-se a pesquisa a apenas um conjunto de lugares de particular interesse (cobertura de submarcações);
  - **Reversibilidade e Home State**: uma rede é dita reversível se, para cada marcação M em $R(M_0)$, M_0, é alcançável a partir de M. Assim, a rede possui a capacidade de retornar a uma marcação inicial. Além disso, em algumas aplicações não é necessário voltar à marcação inicial, mas sim a uma marcação específica. Essa marcação específica é denominada *Home State*.
- **Propriedades Estruturais**: As propriedades estruturais são aquelas que não dependem da marcação, ou seja, possuem dependência exclusivamente da topologia

da rede. Abaixo, serão descritas as principais propriedades estruturais baseadas em Murata (1989) e Maciel, Lins e Cunha (1996):

- **Limitação Estrutural**: uma rede é dita limitada estrutural se o número de *tokens* é limitado para qualquer marcação inicial;
- **Conservação**: a conservação é uma importante propriedade das PN, pois permite a verificação da não destruição de recursos a partir da conservação de *tokens*;
- **Repetitividade**: uma rede é considerada repetitiva se, para uma marcação e uma sequência de transições disparáveis, todas as transições dessa rede são disparadas ilimitadamente;
- **Consistência**: uma rede é dita consistente se, dada uma sequência de transições disparáveis a partir de uma marcação inicial M_0, ele retorna a M_0, porém todas as transições da rede são disparadas pelo menos uma vez.

4 METODOLOGIA E MODELAGEM DO SISTEMA

Neste capítulo são descritos os procedimentos metodológicos adotados nesta pesquisa, os quais estão organizados na Visão geral da metodologia (seção 4.1) e os Modelos GSPN propostos (seção 4.2).

4.1 VISÃO GERAL

Após realizarmos o levantamento dos estudos mais recentes que têm sido feitos pelos pesquisadores da área de banco de dados e de avaliação de desempenho, observamos que poucos estudos, ou quase nenhum, abordam a comunicação entre nós NoSQL como um fator importante. Diferentemente, a nossa proposta, em resumo, combina modelagem com foco em desempenho de um *cluster* NoSQL, levando em consideração o fator da rede, a quantidade de nós em uma infraestrutura com replicação de dados e a consistência configurada no SGBD para planejamento da infraestrutura computacional(rede de comunicação) para esse tipo de aplicação.

Inicialmente, buscamos entender e implementar uma infraestrutura simples – afinal, se fosse complexa não haveria a necessidade de modelos para simplificar o trabalho – para servir de base de estudo e validação do modelo. Depois, escolhemos um banco para representar o paradigma NoSQL, no qual decidimos pelo sistema MongoDB, que entre outros fatores, tem uma implementação leve e de fácil configuração, além de ser um dos mais populares desse gênero.

Em seguida, definimos o objetivo do plano de avaliação, ou seja, os elementos/componentes que impactam no desempenho da infraestrutura, para assim, planejarmos os cenários a serem modelados.

Por último, geramos os modelos de desempenho e avaliamos os cenários, considerando o impacto na métrica adotada, a vazão, especificamente, a vazão do nó primário do *cluster* NoSQL, pois observando-a como medida preventiva, o sistema pode tornar-se mais eficiente e mais preparado para lidar, por exemplo, com as restrições de largura de banda em momentos de uso intenso.

Salientamos que a metodologia proposta – avaliação e planejamento experimental a partir da modelagem de desempenho baseada em Redes de Petri Estocásticas – pode ser utilizada por usuários com experiência em modelos combinatórios ou modelos baseados em espaço de estados para o planejamento e tomada de decisão. É importante ressaltar que essa metodologia também pode ser adotada para auxiliar na modelagem de diversos sistemas que apresentem características semelhantes às abordadas neste trabalho.

Embora nossos modelos visem facilitar, a administradores ou a projetistas de rede, o planejamento da infraestrutura computacional, especificamente, a utilização da rede

para o uso de SGBDs NoSQL, com mensuração da vazão (um dos requisitos necessário para que aplicações, como os SGBDs possam executar com qualidade), pode ser útil criar uma ferramenta para simplificar o entendimento desses modelos para quem não tem conhecimento prático com Redes de Petri Estocásticas, fato esse que se coloca como proposta para trabalhos futuros.

4.2 MODELOS GSPN

Neste trabalho, optamos pelo uso do formalismo de **GSPN**. Justificamos a adoção desse formalismo ao observar que:

- Levando em conta que, embora o formalismo de **Redes de Petri (PN)** permita a especificação de sistemas, possibilitando, inclusive, uma representação matemática através dos mecanismos de análises que permitem, por sua vez, a verificação de propriedades e da corretude do sistema especificado (MACIEL; LINS; CUNHA, 1996), só é possível, apenas, descrever a estrutura lógica de sistemas, já que tal formalismo não inclui nenhum conceito de tempo;

- Mesmo que as **Redes de Filas** (LAZOWSKA et al., 1984) configurem-se como uma das técnicas de modelagem para análise de desempenho mais populares na área de Computação (FLORIN; FRAIZE; NATKIN, 1991), esse tipo de modelo é o mais apropriado para expressar, por exemplo, a representação gráfica dos elementos de fila e/ou servidor ou o comportamento assíncrono de clientes. Nelas, a única forma de interação entre os elementos do sistema, que pode ser modelada de forma direta, é a contenção no acesso aos serviços. Além disso, ela não permite posse simultânea de recursos;

- Ainda que as **Cadeias de Markov**, que é uma técnica para modelagem analítica (BRAGHETTO; FERREIRA; VINCENT, 2010) que possui grande expressividade e permite, por exemplo, descrever arquiteturas e sistemas complexos, para usa-las, especificamente, na modelagem, precisamos enumerar todos os possíveis estados do sistema e todas as possíveis transições entre eles. Assim, essa abordagem, além de ser muito suscetível a erros, torna-se inviável para sistemas com muitos estados.

Portanto, além de considerar esses pontos, levamos em conta que um dos requisitos do sistema em estudo é a concorrência, assim, os modelos estocásticos baseados em Rede de Petri satisfazem os requisitos do sistema, mostrando-se ser uma ferramenta de alto nível de abstração e compacta, se comparada a outras metodologias.

No universo dos processos estocásticos, uma maior ênfase é dada aos processos que dependem apenas do momento presente, os chamados processos Markovianos, representados pelas distribuições exponenciais. Apesar dessas distribuições serem matematicamente tratáveis, as análises das cadeias de Markov podem ser complexas e trabalhosas. Nesse

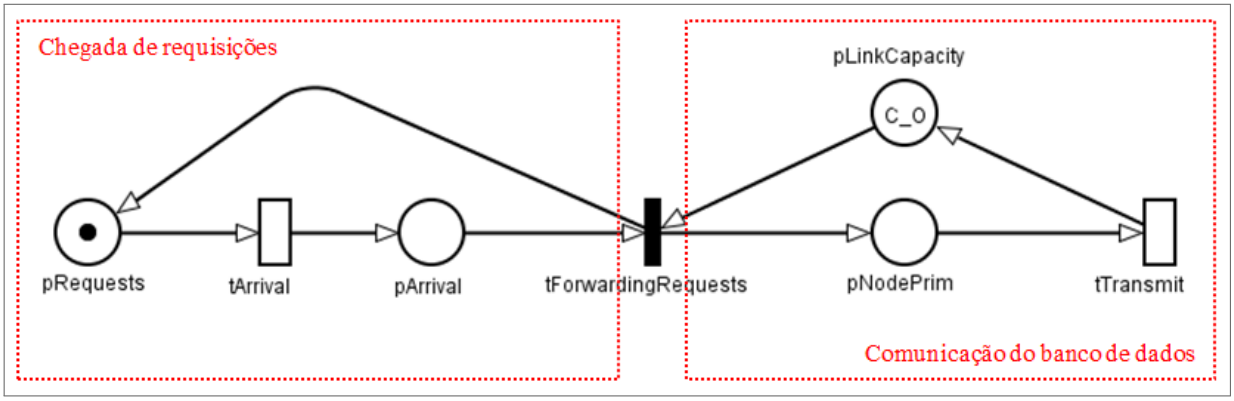
contexto, destaca-se uma característica distinta do formalismo de GSPN em relação ao formalismo de Redes de Petri: o isomorfismo com Cadeias de Markov, devido à propriedade de ausência de memória da distribuição exponencial do atraso dos disparos, que permite a avaliação quantitativa do desempenho de sistemas. Logo, o mapeamento de uma GSPN para um modelo Markoviano equivalente é feito de forma transparente, superando a dificuldade da construção direta (definição dos estados) da Cadeia de Markov.

Da mesma forma em que dividimos a experimentação (experimentos sem concorrência e experimentos com concorrência de tráfego), também dividimos a modelagem do sistema em modelos GSPN sem e com concorrência de tráfego, os quais serão detalhados nas próximas subseções.

4.2.1 Modelos sem tráfego concorrente

De início, propomos um modelo simples para validar o experimento com base no sistema real, também em sua configuração mais simples, como demostrado na Figura 9. Vale lembrar que, para a confecção e análise (estacionária) dos modelos apresentados no decorrer desta seção, utilizamos a ferramenta TimeNet (SYSTEMS; GROUP, 2019).

Figura 9 – Modelo validado



Fonte: própria autoria.

Nesse modelo, Figura 9, representamos como as requisições chegam ao sistema e como esse sistema comporta-se. Assim, medimos a quantidade de dados processados no período de execução do banco de dados, isto é, calculamos a vazão dos dados processados na comunicação. Para tanto, consideramos:

- Requisições chegando a um *cluster NoSQL*. Assim, 100 requisições por *token* equivalem a 100 requisições enviadas, especificamente, para o nó principal (*pNodePrim*);
- Para calcular a taxa de chegada semelhante ao resultado de um banco de dados em operações por segundo (op/s), definimos na transição $tArrival = 1/\text{taxa de chegada}$. Desse modo, podemos atribuir o valor 2 e estimar *tArrival* como 1/2, por exemplo;

- O termo “largura de banda” é usado para descrever a capacidade de transferência de dados em uma unidade de tempo. Assim, definimos uma fórmula para calcular essa capacidade (C_O) e as outras definições envolvidas: C_O = (Bandwidth/(requests_per_token x request_size). Aqui, se tomarmos 12313.6/(100 x 10) $\approx$ 12, o valor 12313.6 será a capacidade máxima em Kbytes por segundo (ou 96,2 Mbit/s), 100 será o número de requisições enviadas ao banco e 10, o tamanho das requisições (em Kbytes). Desse modo, o número de objetos (requisições) que o *link* pode manipular é de $\approx$ 12;

- A fim de observar se a tese levantada, o mecanismo proveniente de SDN pode auxiliar no desempenho da comunicação dos SGBDs, fizemos alusão a esse mecanismo por meio da possibilidade de se configurar os fluxos de dados que podem ser mapeados para uma "fila específica"com uma taxa máxima ou mínima padrão, por exemplo, com 96 Mbit/s ou 48 Mbit/s de largura de banda máxima ou miníma disponível;

- O *link* é capaz de transferir até 12313,6 Kbytes em 1 segundo, portanto a transição *tTransmit* tem valor igual a 1;

- Por fim, para obter a taxa efetiva de transferência de dados, ou seja, a vazão, fizemos da seguinte maneira: *TPnodeprim = E{(#pNodePrim) x requests_per_token x request_size x 1}*. Essa métrica consiste em calcular o valor esperado no nó principal x a quantidade de requisições x o tamanho das requisições x 1, sendo 1 o tempo em que o *link* transfere até 12313,6 Kbytes.

A fim de prover mais informações sobre os componentes do modelo, o Quadro 2 descreve os *lugares* do modelo validado e o Quadro 3 discorre sobre as *transições* desse mesmo modelo.

Quadro 2 – *Lugares* do modelo

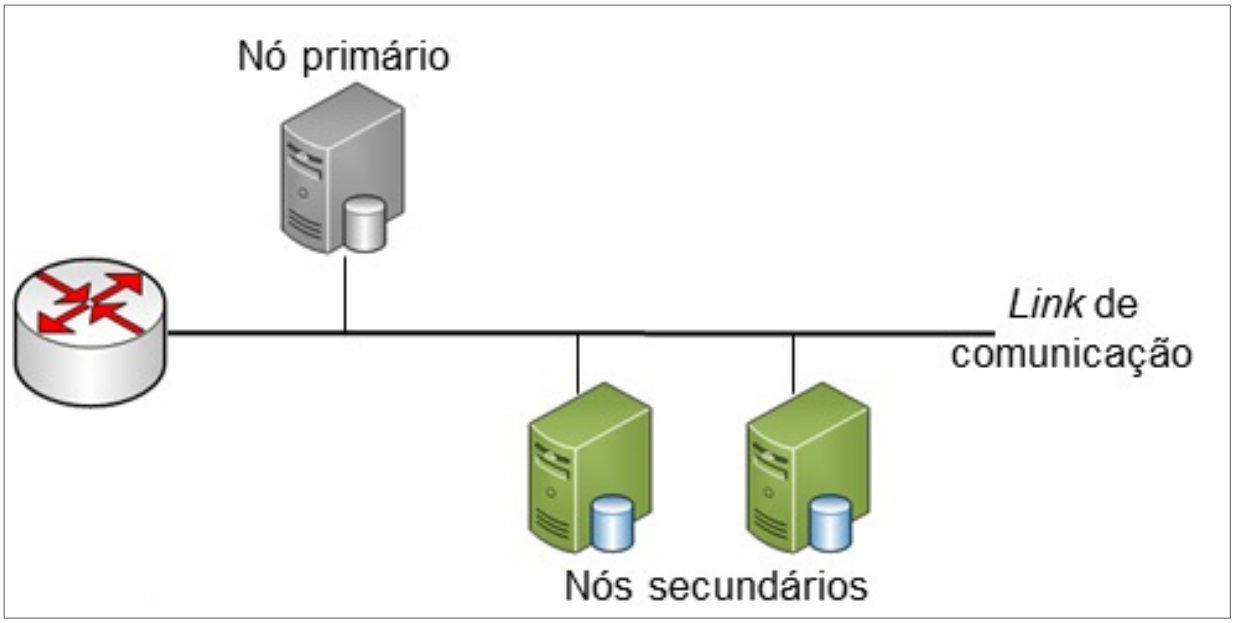
Nome	**Descrição**
pRequests	Representa a solicitação de uma requisição ao banco
pArrival	Representa uma requisição feita ao banco e sua transmissão
pNodePrim	Representa a requisição feita ao servidor primário (principal) do *cluster*
pLinkCapacity	Representa a capacidade de processamento das requisições pelo *link*

Quadro 3 – *Trasições* do modelo

Nome	Tipo	Descrição	Semântica
tArrival	Temporizada	Representa o intervalo entre requisições	Exclusive Server
tForwardingRequests	Imediata	Representa alocação de largura de banda	-
tTransmit	Temporizada	Representa o intervalo de transmissão das requisições	Infinite Server

A seguir, buscamos representar a replicação de dados feita por um *cluster* NoSQL, que pode consistir em um nó primário e em dois ou mais nós secundários, tal como está representado na Figura 10.

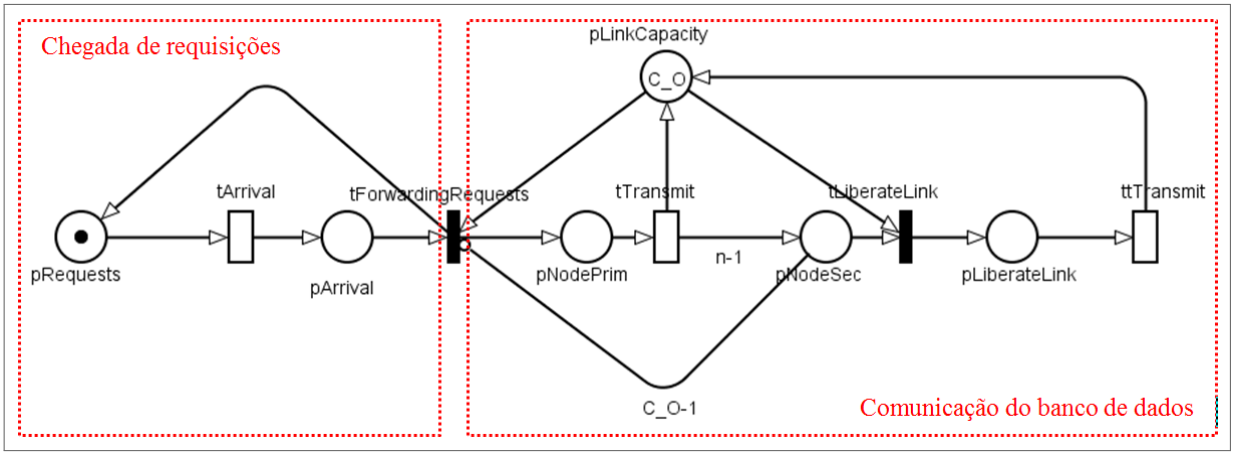
Figura 10 – Representação do sistema para abstração do modelo com replicação de dados



Fonte: própria autoria.

Para esse cenário, é necessário entender o modelo de consistência do sistema adotado, que é o modelo de consistência eventual. Esse modelo de consistência, caracterizado pelo BASE, define que o sistema, em algum momento, será consistente, ou seja, todos os nós do sistema terão "eventualmente"o mesmo estado. Com base nisso, neste segundo modelo (Figura 11), podemos observar que o *pNodePrim* continua recebendo requisições ao replicar dados para o (s) nó(s) secundário(s). Portanto, em algum momento, esses dados estarão completamente sincronizados, caso não haja falha de rede, como uma sobrecarga no *link* ou uma falha do sistema antes do final da sincronização.

Figura 11 – Modelo com replicação



Fonte: própria autoria.

Para esse modelo, além da abstração sobre os componentes que envolvem o sistema (as requisições chegando ao um *cluster NoSQL*, a taxa de chegada, a capacidade de transferência de dados - C_O - e a métrica adotada - a vazão), os quais são similares ao modelo anterior, adotamos:

- Uma expressão de guarda para que a transição *tForwardingRequests* pare de ser disparada quando *C_O* não estiver disponível, ou seja, o sistema para de encaminhar dados se não houver largura de banda disponível;
- A variável *n* representa a definição sobre a quantidade de nós envolvidos na replicação de dados;
- Que ao receber os dados replicados, o(s) *pNodeSec*, que representam os servidores secundários, libera (m) o *link* de rede a partir de (*ttTransmit*) ao mesmo tempo (1 segundo) e passo que *tTransmit*, como veremos nos quadros seguintes.

Quadro 4 – *Lugares* do modelo com replicação de dados (sem concorrência de tráfego)

Nome	**Descrição**
pNodeSec	Representa a requisição replicada no servidor secundário do *cluster*
pLiberate*link*	Representa a finalização das solicitações de requisições ao *cluster*

Quadro 5 – *Trasições* do modelo com replicação de dados (sem concorrência de tráfego)

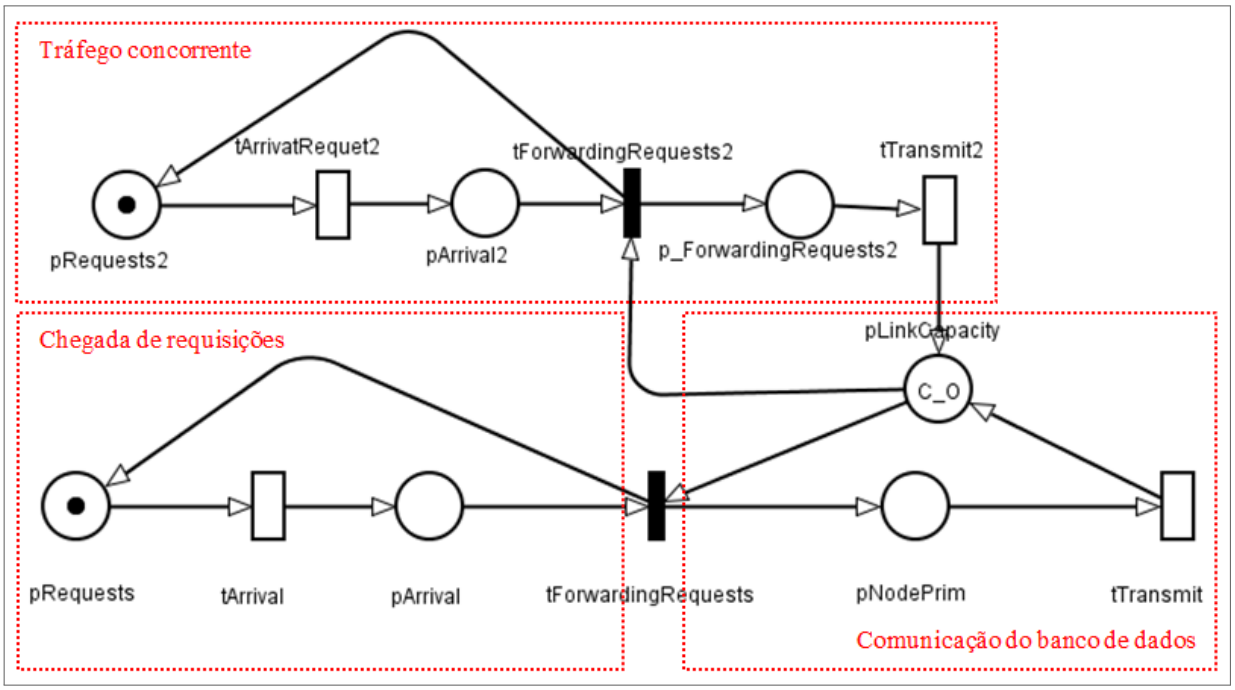
Nome	Tipo	Descrição	Semântica
tLiberateLink	Imediata	Representa a liberação de largura de banda ao final das solicitações de requisições ao banco	-
ttTransmit	Temporizada	Representa o intervalo de transmissão das requisições	Infinite Server

O Quadro 4 relata sobre os *lugares* adicionais desse modelo (representado na Figura 11), com relação ao modelo anterior (demonstrado na Figura 9). Já o Quadro 5 descreve sobre as *transições* desse mesmo modelo (Figura 11).

4.2.2 Modelos com tráfego concorrente

Haja vista que em uma rede é comum a execução, por vezes simultânea, de n aplicações, buscamos, de forma simples, representar essa concorrência no *link* de comunicação. Para tanto, consideramos apenas mais uma outra aplicação concorrente ao banco, como demonstra o modelo seguinte (Figura 12).

Figura 12 – Modelo com tráfego concorrente



Fonte: própria autoria.

Em resumo, esse modelo, Figura 12, configura-se como o sistema sem replicação de dados, apenas recebendo requisições, mas compartilhando seu *link* de comunicação (a largura de banda) com outra aplicação. Nesse contexto, para uma melhor compreensão

dessa configuração do sistema, podemos ver os componentes que constituem esse modelo no Quadro 6, que descreve sobre os *lugares* do modelo e no Quadro 7, que discorre sobre as *transições*.

Quadro 6 – *Lugares* do modelo sem replicação de dados (com concorrência de tráfego)

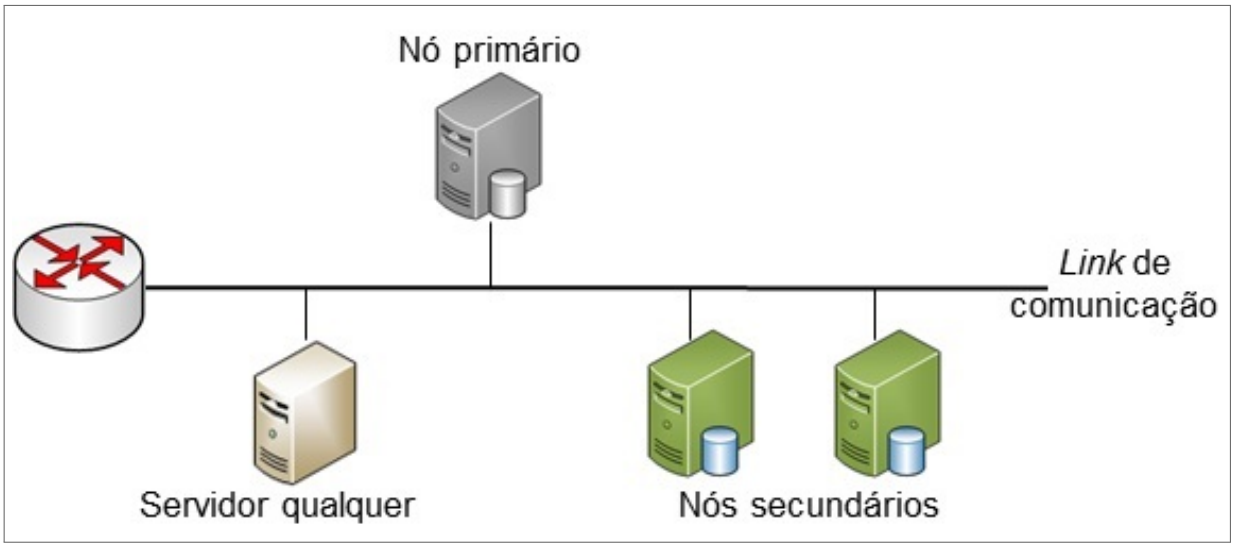
Nome	**Descrição**
pRequests2	Representa a solicitação de uma requisição qualquer a um servidor, também, qualquer
pArrival2	Representa uma requisição feita a um servidor qualquer e sua transmissão
pForwardingRequests2	Representa a solicitação e processamento de uma requisição qualquer

Quadro 7 – *Transições* do modelo sem replicação de dados (com concorrência de tráfego)

Nome	Tipo	Descrição	Semântica
tArrival2	Temporizada	Representa o intervalo entre requisições concorrentes as do banco	Exclusive Server
tForwardingRequests2	Imediata	Representa alocação de largura de banda para as requisições concorrentes as do banco	-
tTransmit2	Temporizada	Representa o intervalo de transmissão das requisições do servidor	Exclusive Server

Todos os modelos, apresentados anteriormente, têm sua contribuição e exemplifica bem, separadamente, os fatores que consideramos importantes para construção do modelo genérico (Figura 14), que foi concebido para a experimentação do sistema. Para esse modelo genérico, adotamos o nível de abstração que está representado na Figura 13.

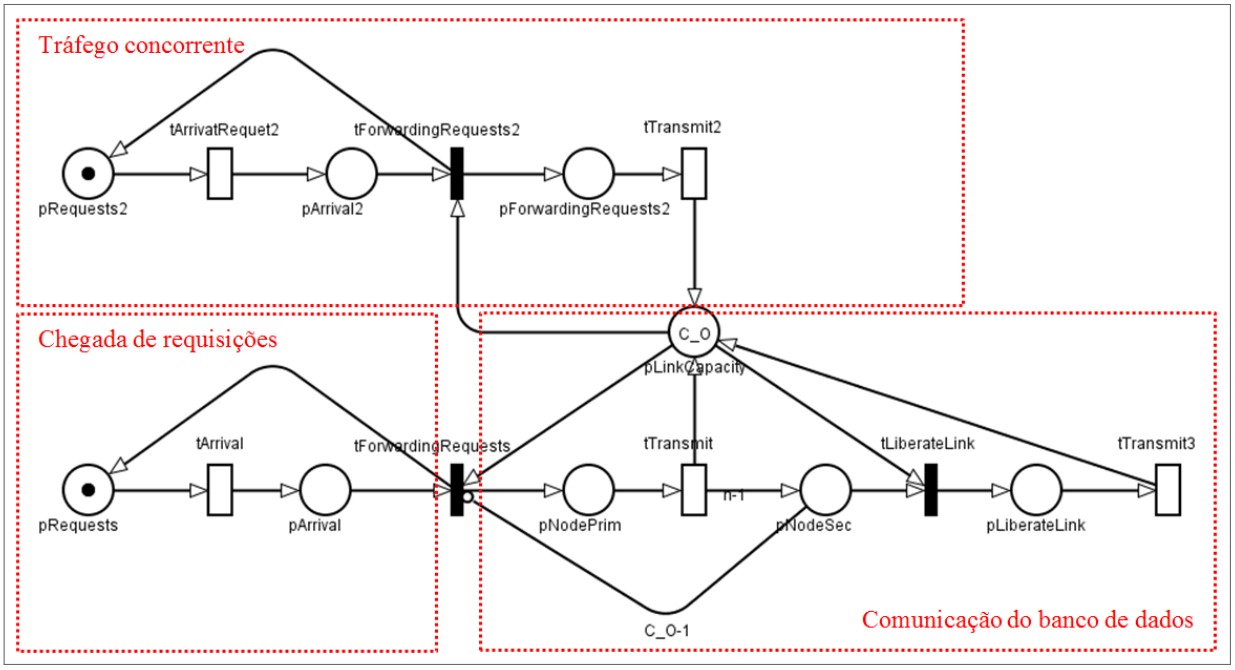
Figura 13 – Representação do sistema para abstração do modelo com replicação de dados



Fonte: própria autoria.

Nessa figura (Figura 13), temos a ideia da replicação de dados do nó primário para os nós secundários e de uma aplicação qualquer concorrente à largura de banda disponível, que foi o nível de abstração adotado para confecção do modelo genérico.

Figura 14 – Modelo com replicação e com concorrência



Fonte: própria autoria.

Esse modelo (Figura 14) é a aglutinação dos modelos anteriores, no qual representa o sistema avaliado como um todo – chegada de requisições, encaminhamento dos dados, replicação e trafego adjacente. O conjunto de marcações alcançáveis dessa rede serviu-nos de base para extrair as propriedades do modelo. Utilizamos o modelo mais complexo, ou

seja, esse modelo genérico (Figura 14), pois o mesmo reflete os outros modelos apresentados anteriormente. Assim, refizemos esse modelo na ferramenta Snoopy(HEINER et al., 2019) e, posteriormente, exportamos para a aplicação Chalie(HEINER; SCHWARICK; WEGENER, 2019), que dentre outros fatores, responsabiliza-se por analisar as propriedades de uma rede. Desse modo, com auxílio do Charlie, a partir do *reachability/ coverability graph* (Gráfico de Alcançabilidade/cobertura), concluímos que:

- A rede possui a característica de ser Homogênea;
- A rede apresentada é Fortemente Conectada (para cada dois nós a e b, existe um caminho direcionado de a para b e vice-versa);
- A rede não é Estruturalmente Limitada, considerando que uma Petri Nets (Redes de Petri) (PN) é k-limitada se o número de marcas de cada lugar da rede não exceder k (k é o número de marcas que um lugar pode acumular) em qualquer marcação;
- A rede em questão não é segura, pois os lugares podem acumular mais do que uma marca, assim, a rede não pode ser dita *Safeness*;
- Tendo em vista que o sistema só usa a largura de banda disponível, o que limita o espaço de estado, a rede atende à propriedade *Boundedness*;
- Podemos dizer que atende à propriedade *liveness*, pois os eventos efetivamente modelados ocorreram durante o funcionamento do sistema, e é livre de *deadlocks*;
- A rede é Limitada, portanto, pode ser considerada Viva sob a regra de disparo seguro, assim, é sempre possível voltar a marcação inicial (*Home State*), o que a faz atender à propriedade de Reversibilidade.

A escolha da técnica de avaliação depende muito do *status* de implementação/construção do sistema a ser avaliado e do custo sobre ele. O uso de modelos nos permite a avaliação de diferentes cenários e a comparação entre eles. Tanto para administradores de recursos de rede, que tem que lidar com a comunicação de aplicações, como SGBDs NoSQL, quanto para outros pesquisadores que queiram seguir este estudo como base para outras implementações, já que modelos de medição também podem ser usados como entrada de um modelo de simulação ou de modelagem, gerando assim, um modelo híbrido, as técnicas de modelagem matemática, especificamente, o formalismo baseado em GSPN, possibilita planejar as seguintes situações de uso: executar diferentes cargas de trabalho, variar os intervalos de envio da requisições, limitar ou dimensionar a largura de banda para as aplicações, diversificar a quantidade de nós envolvidos na replicação, e variar cargas de outras aplicações (concorrência), dando, ao avaliador, diversas opções para modificar e, então, selecionar a configuração mais vantajosa.

O custo de realizar diversas modificações em um modelo, como os apresentados aqui, é quase zero, pois, em alguns casos, existe um custo de tempo para algumas modificações, as quais exigem um tempo maior para a análise do modelo.

Um modelo pode ser considerado uma representação de um ou mais pontos de vista de um sistema em um determinado nível de abstração. Assim, definimos a taxa de chegada, o número de requisições, a capacidade do *Link* e a configuração do *replica set* como o conjunto de variáveis, e a capacidade de transferência (vazão) do nó primário de SGBD NoSQL como a métrica de avaliação, os quais representam nossa abstração do sistema.

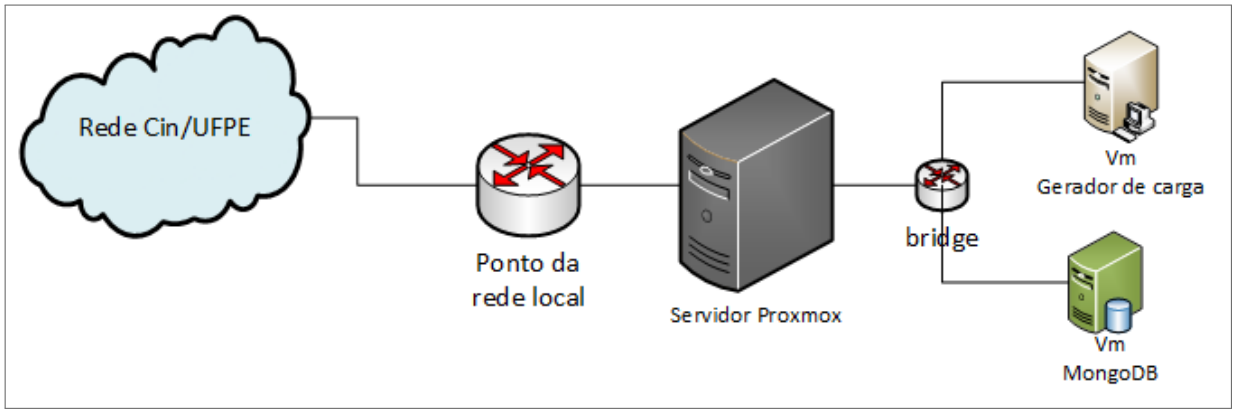
5 RESULTADOS EXPERIMENTAIS

Esta seção apresenta os resultados experimentais que demonstram a viabilidade prática da técnica de modelagem adotada. Inicialmente, a validação do modelo é apresentada na seção 5.1, em seguida, é demonstrado, na seção 5.2, o projeto de experimentos realizado nesta pesquisa, e, por fim, os resultados experimentais de desempenho são apresentados na seção 5.3.

5.1 VALIDAÇÃO

Para validar a modelagem do sistema, a configuração experimental consistiu em duas máquinas virtuais criadas, usando o Poxmox 5.0 (em um servidor com 4 GB de RAM, 2 de CPU com dois núcleos de 2,33 GHz e HD de 150 GB), sendo uma Vm para o gerador de carga de trabalho YCSB e uma Vm para o sistema NoSQL (Mongodb). Todas as máquinas virtuais possuíam a mesma configuração: 512 MB de RAM, 1 CPU com dois núcleos, 50 GB de HD e 1 Adaptador de Rede. As máquinas virtuais residiam na mesma *LAN* com capacidade de 96,2 Mbits ($\approx 100 Mbit/s$). O sistema operacional usado nas máquinas virtuais foi o Debian 8.0 (jessie). A infraestrutura montada, segue representada na Figura 15.

Figura 15 – Infraestrutura montada para validação dos modelos



Fonte: própria autoria.

Para realização desta parte do trabalho, foi necessário a utilização de alguns *scripts* simples para a automatização das execuções do *benchmark* YCSB (COOPER, 2019) e para controlar a captura dos dados trafegados pela rede. Utilizamos a ferramenta Tshark (WIRESHARK, 2019) para capturar a taxa de transferência dos dados (vazão) do nó principal do sistema. Além desses, utilizamos o Minitab(MINITAB, 2019) para calcular as estatísticas necessárias.

Tendo em vista que a média harmônica é uma medida de localização usada principalmente quando os dados consistem em um conjunto de taxas (KOMIĆ, 2011), revertemos, para facilitar o nosso trabalho, as taxas obtidas com a execução do sistema, vazão = Kbyte/segundo em tempo (1/ vazão). Assim usamos uma faixa de valores ideal para o cálculo da média aritmética da população, por meio do *One-Sample T: T* do software Minitab. Neste sentido, a Tabela 3 constata os resultados do modelo em comparação ao sistema real - resultado médio de 30 execuções.

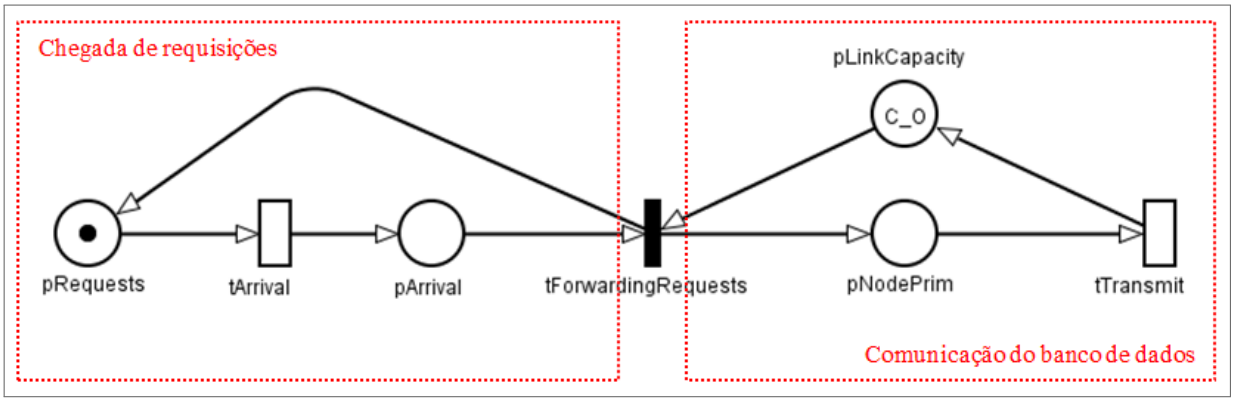
Tabela 3 – Resultados dos experimentos e do modelo

Tempo do sistema físico (μs)	I.C.(95%) (μs)	Tempo do modelo (μs)
501	[499; 503]	500

Fonte: própria autoria.

O resultado do modelo a que se refere à validação foi obtido a partir da análise estacionária do modelo simples, sem concorrência de tráfego. Vejamos tal modelo, novamente, na Figura a seguir:

Figura 16 – Modelo utilizado na validação



Fonte: própria autoria.

5.2 DESIGN OF EXPERIMENTS

O Design of Experiments (DoE) é uma técnica utilizada para se planejar experimentos, ou seja, para definir quais dados, em que quantidade e em que condições devem ser coletados durante um determinado experimento, buscando, basicamente, satisfazer dois grandes objetivos: a maior precisão estatística possível na resposta e o menor custo. Tal técnica pode ser aplicada de várias formas: Tratamento em pares, Tratamento em blocos, Quadrado Latino, Quadrado Greco-Latino, Quadrado Hiper-Greco-Latino e Experimentos Fatoriais – esse último sendo adotado neste trabalho. Para a determinação dos processos do DoE, os seguintes elementos devem ser considerados:

- **Entrada**: Quais variáveis são do interesse para desenvolvimento do produto/serviço;
- **Saída**: Quais variáveis são mais influentes na resposta em y;
- **Fatores de Entrada Controláveis e não Controláveis**:
  - Valor atribuído aos “x” influentes de modo que a variabilidade em “y” seja pequena;
  - Valor atribuído aos ‘x’ influentes de modo que ‘y’ esteja perto da exigência nominal;
  - Valor a ser atribuído aos ‘x’ influentes de modo que os efeitos das variáveis não controláveis sejam minimizadas.

Uma vez estabelecidos os parâmetros do DoE, são necessárias definir as diretrizes para a sua utilização. Para isso, alguns passos são essenciais, dentre eles, destacamos: i) o completo reconhecimento e relato dos problemas que afetam o produto e/ou o serviço (esse relato faz parte do planejamento pré-experimento, que, em nosso caso, deu-se a partir das análises de outros trabalhos e de pequenos testes durante a montagem do ambiente de experimentação); ii) a escolha dos fatores e dos níveis de cada parâmetro – aqui, optamos por definir valores mínimos e máximos, que demonstrassem efeitos consideravéis, mas, tomando o cuidando para que a escolha dos valores que não “estourassem” o espaço de estados das redes modeladas; iii) seleção da(s) variável(eis) de resposta (y) – com a análise e o ajuste da vazão como medida preventiva, o sistema pode tornar-se mais eficiente e mais preparado para lidar com restrições extras de largura de banda em momentos de uso intenso, por esse motivo, escolhemos-a como métrica de desempenho e variável resposta no DoE; iv) escolha do planejamento experimental (a técnica ou método matemático e/ou estatístico) – utilizamos o Experimento Fatorial, que é apropriado para quando todas as combinações dos níveis dos fatores de controle são realizadas.

Posto isso, propomos em dois experimentos (sem e com concorrência de tráfego) adotar o planejamento fatorial completo com 4 fatores, totalizando 24 execuções em cada experimento. Desta forma, para maior compreensão do planejamento fatorial, vejamos a tabela seguinte (Tabela 4), que aponta sobre os fatores e os níveis adotados nos experimentos deste trabalho:

Tabela 4 – Componentes do planejamento fatorial

Fatores	Níveis
Taxa de chegada (Tempo entre requisições)	2 e 4
Número de requisições (Operações por segundo)	100 e 200
Largura de banda (Megabit por segundo)	48,1 e 96,2
Configuração do *replica set* (Número de nós no *cluster*)	1, 3, e 50

Fonte: própria autoria.

O número de operações enviadas para o banco de dados, por exemplo, 100 e 200 requisições, para fins do planejamento experimental, é configurável no modelo como *Definition*, chamada, internamente, de *num_requests*. Tais valores parecem não ser tão expressivos, no entanto não geram um grande número de espaços de estado e, ainda sim, representam, consideravelmente, diferentes resultados.

O fator *taxa de chegada* é atrelado a transição *t_Arrival* do modelo, que nada mais é que a taxa na qual as solicitações são processadas, por exemplo, 2 requisições/s. Tal fator é, frequentemente, levado em conta na avaliação de vários sistemas, incluindo SGBDRs e SGBDs NoSQL, tornando-se, assim, útil incluí-lo em nosso experimento.

Variando a capacidade do enlace (largura de banda), que está ligada ao modelo pela variação do *C_O*, em que se calcula, por exemplo, o fator *Largura de banda*, que é calculado como: *C_O* = (Bandwidth/(requests_per_token x request_size), há uma alusão aos trabalhos que alegam que SDN infere no impacto do desempenho de comunicação dos SGBDs na rede, cuja metodologia, em alguns desses casos, dá-se pela limitação ou priorização do tráfego de um *cluster* de banco de dados por meio do recurso *OpenFlow* QoS (protocolo padrão de SDN), mapeando a comunicação para uma taxa mínima ou máxima de largura de banda do *link*, o que poderia implicar uma não sobrecarga na rede devido à replicação ou a concorrência de tráfego.

Em um *replica set* do MongoDB, a capacidade mínima e máxima dos nós que podem ser configurados em um *cluster* é de 3 a 50 nós, respectivamente. A configuração de replicação pode desempenhar um papel fundamental no desempenho do SGBD e na consistência do banco de dados, garantindo pelo menos a consistência eventual. No entanto, no caso de um *failover*, esse processo pode ser interrompido se houver um congestionamento na rede (inclusive devido ao grande número de nós a serem sincronizados), ou esse processo pode consumir mais tempo do que o desejado, devido à existência de muitos nós e/ou saltos na rede. Além disso, a réplica (nó secundário) pode não conter o valor mais atualizado do nó principal (nó primário). Assim, em caso de falha abrupta do sistema principal, a consistência eventual deixa de ser garantida, já que o BASE determina que todos os nós nesses tipos de sistemas terão, em algum momento, o mesmo estado. Posto isso, definimos

no modelo, por meio da opção *Definition*, os seguintes valores para o fator *Configuração do replica set*: *ReplicaSet*=1, com a intenção de representar o sistema sem preocupação com a replicação; *ReplicaSet*=3, que retrata um *cluster* com a configuração mínima, ou seja, com três nós, sendo um nó primário replicando dados para outros dois nós; e *ReplicaSet*= 50, que reproduz a configuração máxima do *cluster*, com 50 nós. Ajustar o fator *ReplicaSet*, significa diversificar a quantidade de nós no *link*, e consequentemente, afetar a consistência do banco, bem como a vazão do nó primário.

5.3 EXPERIMENTO DE DESEMPENHO

Na análise de experimentos, a análise gráfica permite identificar qual o melhor nível de cada fator, por exemplo, os gráficos de efeitos principais ilustram a variação média das respostas em função da mudança no nível de um fator, mantendo os outros fatores importantes. Outro método interessante é o ranking (ranque). O ranque demostra qual é o fator mais influente para a resposta, isto é, qual tem maior ou menor efeito no experimento. Normalmente, o ranque é ordenado por ordem decrescente, levando em conta os valores absolutos de todos os efeitos.

Vale lembrar que o efeito é a mudança na resposta devido a uma mudança no nível do fator, isto é, o efeito de um fator é definido como a mudança na resposta produzida por uma mudança no nível do fator, o que é chamado de efeito principal, porque se referem aos fatores principais em estudo. O efeito principal do fator A é a diferença entre a resposta média no nível alto de A e a resposta média no nível baixo de A, sendo "+"o nível alto e "−"o nível baixo. Essa forma de representação dos níveis dos fatores é bastante utilizada em ranqueamentos que não se consideram os valores absolutos dos efeitos e interações, deixando mais facilmente identificável qual é o melhor nível de cada fator. Já a interação é quando o efeito de um fator depende do nível do outro fator, ou seja, o efeito de interação descreve a variação média de um fator em função dos níveis dos outros fatores.

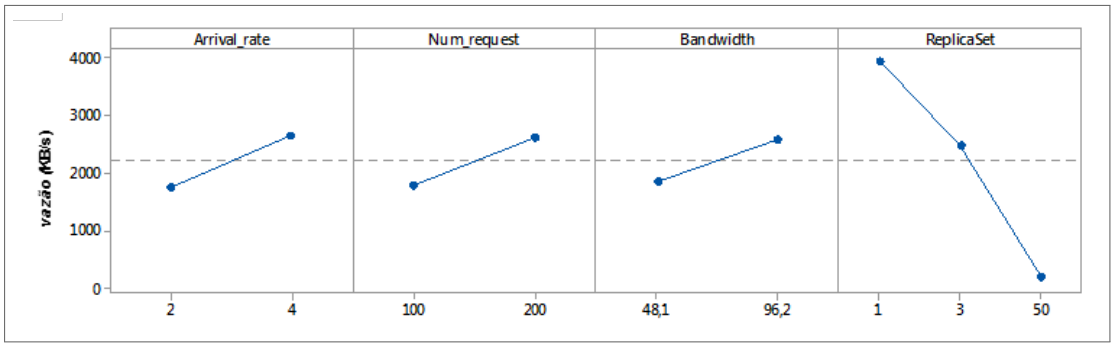
Visto tais métodos de análise e conceitos, a seguir, discutiremos os resultados experimentais do nosso estudo.

5.3.1 Experimentos sem tráfego concorrente

Para os experimentos sem concorrência de tráfego, pode-se considerar tanto os modelos da Figura 9 e 11 quanto o modelo genérico (Figura 14), nesse último, com as devidas adequações.

A Figura 17 e a Tabela 5 ilustram os resultados da abordagem sem concorrência de tráfego, especificamente, o gráfico de efeitos principais e o ranque dos efeitos principais e de interação, considerando os efeitos de segunda ordem, já que as interações de ordem mais alta geralmente são insignificantes (MONTGOMERY, 2017).

Figura 17 – Gráfico de efeitos sem concorrência de tráfego



Fonte: própria autoria.

Tabela 5 – Efeitos dos fatores e interações sem concorrência de tráfego

Fator / Interação	Efeito
ReplicaSet	-1746
Arrival_rate x ReplicaSet	618,1
Num_request x ReplicaSet	599
Arrival_rate	462,32
Num_request	426,9
Bandwidth	364,6
ReplicaSet	-279,1
Bandwidth x ReplicaSet	269,7
Num_request x ReplicaSet	-172,2
Num_request x Bandwidth	-169,4
Arrival_rate x Bandwidth	-159,9
Arrival_rate x ReplicaSet	-155,9
Bandwidth x ReplicaSet	35,01
Arrival_rate x Num_request	-9,909

Fonte: própria autoria.

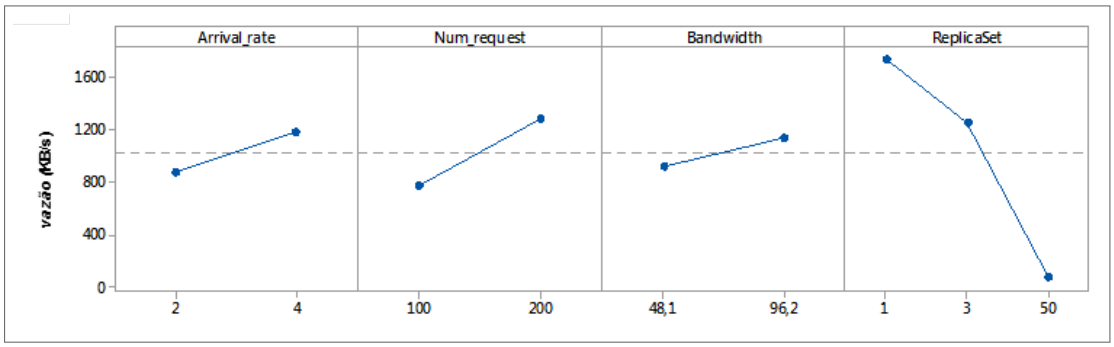
No primeiro experimento, sem concorrência de tráfego, verificamos que o número de nós no *cluster* é o fator que mais impacta na métrica (vazão), ou seja, quanto mais nós são adicionados na rede, menor a capacidade de transferência de dados, como podemos ver no gráfico representado na Figura 17. Com a variação de 1, 3 e 50 nós na replicação, a vazão do nó principal varia de 179,66 KB/s a 3951,30 KB/s. Contudo, se olharmos para o ranque (Tabela 5), podemos concluir que interações como: *Arrival_rate* x *ReplicaSet* - interação da taxa de chegada com o número de nós na configuração do *replica set* e *Num_request* x *ReplicaSet* - número de requisições em interação com número de nós na configuração do*replica set*), tem um efeito maior que as variações de alguns dos outros fatores sozinhos,

como: *Arrival_rate* (taxa de chegada), *Num_request* (número de requisições) e *Bandwidth* (largura de banda).

5.3.2 Experimentos com tráfego concorrente

Para os experimentos com concorrência de tráfego, pode-se considerar tanto o modelo da Figura 12 quanto o modelo genérico (Figura 14). A seguir, a Figura 18 e a Tabela 6, demonstram o gráfico e o ranque (efeitos e interações de segunda ordem.) do sistema, considerando um tráfego simultâneo ao banco de dados.

Figura 18 – Gráfico de efeitos com concorrência de tráfego



Fonte: própria autoria.

Tabela 6 – Efeitos dos fatores e interações com concorrência de tráfego

Fator / Interação	Efeito
ReplicaSet	-715,6
Num_request x ReplicaSet	264,2
Num_request	254,6
ReplicaSet	-235,6
Arrival_rate x ReplicaSet	195,3
Arrival_rate	156,3
Bandwidth x ReplicaSet	123,4
Bandwidth	112,9
Num_request x Bandwidth	-74,66
Arrival_rate x ReplicaSet	-39,02
Arrival_rate x Bandwidth	-34,34
Bandwidth x ReplicaSet	-24,96
Arrival_rate x Num_request	-23,77
Num_request x ReplicaSet	-23,26

Fonte: própria autoria.

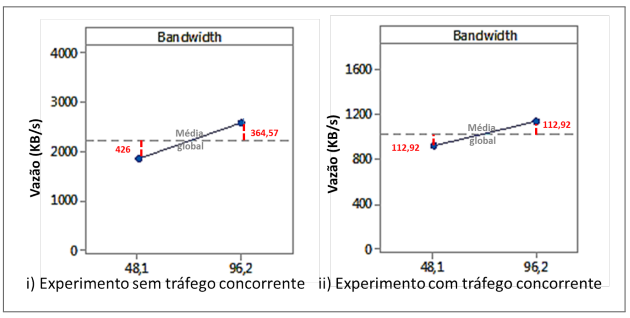
Nesse experimento, com concorrência de tráfego, verificamos que o número de nós no *cluster* é o fator que mais impacta na métrica (vazão), assim como o experimento anterior, como podemos ver no gráfico representado na Figura 18.Só que nesse caso, a vazão varia entre 76,6 KB/s a 1713,40 KB/s. Contudo, se olharmos para o ranque (Tabela 6), podemos concluir que interações como: *Num_request* x *ReplicaSet* - interação da quantidade de requisições com número de nós na configuração do *replica set* - tem maior impacto que os fatores, isolados, *Num_request* (número de requisições) e *ReplicaSet*; *Arrival_rate* x *ReplicaSet* - interação da taxa de chegada com número de nós na configuração do *replica set* - tem maior impacto que o fator Arrival_rate; *Bandwidth* x *ReplicaSet* - interação da largura de banda com número de nós na configuração do *replica set* - tem maior impacto que o fator *Bandwidth*, isoladamente.

5.3.3 Considerações finais sobre os resultados

Nos gráficos de efeitos, a linha tracejada é a média global. Assim,a média global no gráfico sem concorrência de tráfego é de $\approx$ 2205 KB/s e no gráfico com concorrência de tráfego equivale a $\approx$ 1027 KB/s. Diante disso, observando as duas configurações de experimento e comparando-as, notamos que além do fato de *ReplicaSet*(número de nós no *cluster* ou configuração do *replica set*) ser o fator mais marcante na avaliação de ambos os experimentos, os fatores *Arrival_rate*, *Num_request* e *ReplicaSet* têm as diferenças entre as médias dos níveis, com relação à média global, sem concorrência no tráfego são maiores que a diferença dos níveis dos experimentos com concorrência, como pode ser vista nas Tabelas 5 e 6).

O fator mais interessante de se perceber é o *bandwidth*, já que a diferença das médias dos níveis desse fator, sem concorrência e com concorrência de tráfego, implica dizer que a vazão varia menos com ajustes na largura de banda da rede, ou seja, mesmo com concorrência de tráfego e com uso da banda limitada a $\approx 42 Mbit/s$, a variação da vazão é menor (tal variação pode ser vista na Figura 19), confirmando, assim, a tese dos trabalhos que sugerem SDN, especificamente, a limitação de tráfego através do recurso *OpenFlow* QoS), como uma alternativa para aliviar a carga em SGBDs.

Figura 19 – Gráfico de efeitos do fator largura de banda em ambos os experimentos



Fonte: própria autoria.

6 CONCLUSÃO

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs)NoSQL vêm sendo, constantemente, objeto de estudo por serem os mais adequados para trabalhar com grandes volumes de dados e requisições, atendendo, assim, a demandas da Internet das Coisas (IoT) e de aplicações e serviços que integram o *Big Data* (Facebook, Google e Amazon), principalmente por sua capacidade de crescer e se adaptar à carga recebida. No entanto, esses SGBDs são sistemas distribuídos, assim, a comunicação pode ser um gargalo de desempenho.

A popularização do NoSQL bem como a ausência por trabalhos que avaliem a comunicação envolvendo essesSGBDs são a base que nos motivou a realizar este trabalho, que tem como objetivo principal a avaliação de desempenho de Bancos de Dados NoSQL, focando na comunicação de rede de um *cluster* NoSQL, e com base na metodologia Generalized Stochastic Petri Nets (Redes de Petri Estocásticas Generalizadas) (GSPN), visando facilitar, a administradores ou a projetistas de rede, o planejamento da utilização da infraestrutura computacional (rede de comunicação) para esses SGBDs.

Para atingir os objetivos esperados, foi necessário discutir as principais características e diferenças entre os SGBDs NoSQL e os SGBDs relacionais (Capítulo 2), para ficasse evidente a escolha do sistema estudado e a problemática que esse sistema envolve, como por exemplo, a importância da replicação e as desvantagens desse mecanismo para os sistemas NoSQL. Com base na discussão de outros outros trabalhos e nos conceitos descritos no Referencial Teórico (Capítulo 3), mostramos o quão é necessário se observar a infraestrutura que prover o funcionamento do sistema, ou seja, que é importante garantir a fluidez ou o não congestionamento da comunicação. Discutimos, também, conceitos e metodologias que auxiliam na solução para possíveis gargalos de desempenho na comunicação.

Para o desenvolvimento desta pesquisa, foi de grande importância o uso da modelagem como recurso para avaliação de desempenho do SGBD em rede. Esse recurso permitiu representar bem as relações dos componentes do sistema escolhido e evitou custos adicionais, como o tempo de configuração em N equipamentos e os custos de aquisição desses equipamentos. Além disso, os modelos gerados poderão: ser utilizados em outros trabalhos, ou por usuários com experiência em modelos combinatórios e/ou em modelos baseados em espaço de estados para o planejamento e tomada de decisões; e ser adotados para auxiliar na modelagem de diversos sistemas que apresentem características semelhantes.

A definição da taxa de transferência (vazão) como métrica para avaliar o desempenho do *cluster* em rede permitiu observar o valor máximo do tráfego aceito no pico de utilização do sistema, ou seja, no momento em que ocorre a replicação e um tráfego simultâneo adicional. A métrica possibilita ajustar o sistema ou a rede de acordo as necessidades, como medida preventiva para tornar o sistema mais eficiente e mais preparado para lidar

com restrições extras de largura de banda nos momentos de uso intenso, principalmente porque a largura de banda de rede necessária para a sincronização de réplica é diretamente proporcional ao volume e à taxa de requisições que ocorrem no nó primário de um *cluster* NoSQL. Assim, quando a largura de banda da rede não é suficiente, ela pode se tornar o gargalo na replicação e, consequentemente, no desempenho do sistema.

Acreditando ser um diferencial, embora em abstração simplista (tendo em vista que hoje não há um padrão na comunicação para os diversos tipos de sistemas NoSQL), ressaltamos que nossos modelos possibilitam a alteração do uso de largura de banda pelo sistema – fazendo alusão aos trabalhos que sugerem SDN para auxiliar no desempenho de comunicação dos SGBDs na rede, cuja metodologia, em alguns desses casos, é mapear a comunicação para uma taxa mínima ou máxima de largura de banda de *link* de um determinado fluxo de dados. Assim, observamos que essa abstração causou, na análise dos resultados experimentais, uma diferença nas médias com relação à média global do fator *bandwidth*. Nos experimentos essa diferença de médias, sem concorrência e com concorrência de tráfego, implica dizer que a vazão varia menos com ajustes na largura de banda da rede mesmo com a concorrência de tráfego, ou seja, mesmo com concorrência de tráfego e com uso da banda limitada a $\approx 42 Mbit/s$, a variação da vazão é menor, confirmando, assim, a tese levantada de que tal mecanismo proveniente do paradigma SDN pode auxiliar no desempenho de comunicação dos SGBDs.

Portanto, neste trabalho identificamos que o volume e taxa de chegada da carga recebida, a replicação, a concorrência na largura de banda, o nível de consistência configurada, a quantidade de nós na replicação e a metodologia de mapear a comunicação a uma taxa mínima ou máxima de largura de banda impactam no desempenho do *cluster* na rede. Como referenciados em algumas pesquisas e evidenciados neste trabalho, esses aspectos, que têm influência na taxa de transferência de dados (vazão) do SGBD, devem ser levados em conta quando se deseja um melhor desempenho, seja no aspecto da infraestrutura computacional (*link* de comunicação), seja no próprio desempenho do banco.

Este trabalho traz como principal contribuição os modelos e os projetos/análises de experimentos levando em conta esses fatores, proporcionando, assim, mais um método de avaliação de desempenho para administradores de rede, que tem como objetivo prover QoS para os serviços de banco de dados modernos e mais adequados para grandes armazenamento de dados, como os SGBDs NoSQL, além de contribuir para um melhor planejamento da alocação de recursos de rede, como a largura de banda, para funcionamento desses sistemas.

6.1 LIMITAÇÕES DO TRABALHO

A princípio, tentamos reproduzir o sistema por completo, incluindo a parte de configuração de QoS por *openflow* em switch/roteador com Openwrt [4] para limitar/priorizar o tráfego de um *cluster* NoSQL. Mesmo com muitas tentativas, não conseguimos obter êxito nas definições das regras para limitar o tráfego do *cluster* – o que evidenciou, ainda mais, o quão as técnicas de modelagem matemática de sistemas podem contribuir para economia de tempo, trabalho e aquisição de equipamentos.

Acreditando-se que por se tratar de um ambiente virtualizado, sentimos uma limitação ao realizarmos os experimentos com muitas requisições, por exemplo, mais de 100 requisições enviadas pelo gerador de carga (YCSB), diminuindo, assim, a capacidade de testes para a validação do modelo GSPN proposto.

O uso do formalismo de Redes de Petri Estocásticas Generalizadas reduz o espaço de estados da Cadeia de Markov, porém, o problema do aumento do tamanho e da complexidade dos sistemas não é solucionado, pois esse formalismo resolve apenas o ponto de vista da modelagem de sistemas.

O espaço de estados gerado pelos modelos é finitos, pois decidimos por controlar as variáveis quantidade e tamanho das requisições, que estão diretamente ligadas à capacidade de processamento do *link* (C_O). Assim, um aumento considerável dos valores dessas variáveis pode tornar demorada a análise do modelo, além de causar um "estouro" da memória, tornando inviável a execução da análise.

6.2 TRABALHOS FUTUROS

Os testes, os modelos e os experimentos deste trabalho foram direcionados com a utilização do MongoDB. Como trabalho futuro, pretendemos expandir ou agrupar a metodologia para outros tipos de bancos de dados NoSQL, já que, quanto mais abrangente for a metodologia, maior também será o impacto sobre a implementação dos diversos tipos desse SGBD.

Analisar outros fatores, como o consumo de energia sobre o desempenho geral do sistema, ou ainda, definir outras métricas podem agregar ainda mais a pesquisa. Dessa forma, são aspectos a serem considerados no futuro.

Em trabalhos futuros, pretendemos, ainda, adaptar nossos modelos ou propor outros em direção à dependabilidade, principalmente no tocante à mensuração de falhas durante o processo de replicação ou de disponibilidade dos dispositivos de rede.

Tendo em vista que os nossos modelos visam facilitar, a administradores ou a projetistas de rede, o planejamento da utilização de largura de banda de rede para o uso de

[4] OpenWrt é uma distribuição Linux facilmente customizável, que foi originalmente concebida para substituir a firmware padrão (e geralmente proprietária) de roteadores sem-fio (WiFi) comuns, para, desta forma, ganhar a liberdade e a capacidade de integrar novas funcionalidades ao dispositivo, como por exemplo, o suporte a *openflow*

SGBDs NoSQL, pensamos, para trabalhos futuros, criar uma ferramenta para simplificar o entendimento dos modelos para quem não tem conhecimento prático com Redes de Petri Estocásticas.

E por fim, futuramente, buscaremos prevenir o problema de explosão do espaço de estados dos modelos.

REFERÊNCIAS

ABADI, D.; AGRAWAL, R.; AILAMAKI, A.; BALAZINSKA, M.; BERNSTEIN, P. A.; CAREY, M. J.; CHAUDHURI, S.; DEAN, J.; DOAN, A.; FRANKLIN, M. J. et al. The beckman report on database research. *Communications of the ACM*, ACM, v. 59, n. 2, p. 92–99, 2016.

ALMEIDA, G. B. e V. Modelagem de sistemas de software com redes de petri estocásticas. IV Simpósio Brisiieiro de Engenharia de Software - SBC, 1990.

ASAY, M. *NoSQL databases eat into the relational database market*. 2019. Acessado em 08/03/2019. Disponível em: <https://www.techrepublic.com/article/nosql-databases-eat-into-the-relational-database-market/>.

ATTIYA, H.; ELLEN, F.; MORRISON, A. Limitations of highly-available eventually-consistent data stores. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 28, n. 1, p. 141–155, 2017.

BÉRCES, M.; IMRE, S.; PRAKFALVI, A. Performance analysis of entanglement-based competition resolution in distributed systems. In: IEEE. *2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)*. [S.l.], 2018. p. 1–5.

BLOCH, G.; GREINER, S.; MEER, H. de; TRIVEDI, K. S. Queueing networks and markov chains. *book, editor John Wiley and sons*, 1998.

BRAGHETTO, K. R.; FERREIRA, J. E.; VINCENT, J.-M. Performance analysis modeling applied to business processes. In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL. *Proceedings of the 2010 Spring Simulation Multiconference*. [S.l.], 2010. p. 122.

BROACH, J.; MALTZ, D.; JOHNSON, D.; HU, Y. C.; JETCHEVA, J. A performance comparison of multi-hop wireless network routing protocols. In: *proceeding of Mobicom*. [S.l.: s.n.], 1988. v. 98.

BRUNEO, D. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 25, n. 3, p. 560–569, 2014.

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.

CARDOSO, J.; VALETTE, R. *Redes de petri*. [S.l.]: Universidade Federal de Santa Catarina, 1997.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009.

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

CHAUDHARY, R.; AUJLA, G. S.; KUMAR, N.; RODRIGUES, J. J. Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. *IEEE Communications Magazine*, IEEE, v. 56, n. 2, p. 118–126, 2018.

COLOMBO, P.; FERRARI, E. Enhancing mongodb with purpose-based access control. *IEEE Transactions on Dependable and Secure Computing*, IEEE, v. 14, n. 6, p. 591–604, 2015.

COOPER, B. F. *Yahoo! Cloud System Benchmark (YCSB)*. 2019. Acessado em 25/02/2019. Disponível em: <https://github.com/brianfrankcooper/YCSB>.

CORBELLINI, A.; MATEOS, C.; ZUNINO, A.; GODOY, D.; SCHIAFFINO, S. Persisting big-data: The nosql landscape. *Information Systems*, Elsevier, v. 63, p. 1–23, 2017.

COSTA, V. T. da; COSTA, L. H. M. K.; DUARTE, O. C. M. B.; JUNIOR, F. G. V. R. Controle e isolamento de recursos em ambientes de redes virtuais openflow. 2014.

CUI, X.; MIOR, M.; WONG, B.; DAUDJEE, K.; RIZVI, S. Netstore: leveraging network optimizations to improve distributed transaction processing performance. In: ACM. *Proceedings of the Second International Workshop on Active Middleware on Modern Hardware*. [S.l.], 2017. p. 1–10.

DAS, T.; CARIA, M.; JUKAN, A.; HOFFMANN, M. Insights on sdn migration trajectory. In: IEEE. *2015 IEEE International Conference on Communications (ICC)*. [S.l.], 2015. p. 5348–5353.

DAS, V. *Learning Redis*. [S.l.]: Packt Publishing Ltd, 2015.

DB-ENGINES. *Ranking*. 2019. Acessado em 03/03/2019. Disponível em: <https://db-engines.com/en/ranking>.

FARIAS, V. A.; SOUSA, F. R.; MAIA, J. G. R.; GOMES, J. P. P.; MACHADO, J. C. Regression based performance modeling and provisioning for nosql cloud databases. *Future Generation Computer Systems*, Elsevier, v. 79, p. 72–81, 2018.

FLORATOU, A.; PATEL, J. M. Replica placement in multi-tenant database environments. In: IEEE. *2015 IEEE International Congress on Big Data*. [S.l.], 2015. p. 246–253.

FLORIN, G.; FRAIZE, C.; NATKIN, S. Stochastic petri nets: Properties, applications and tools. *Microelectronics Reliability*, Elsevier, v. 31, n. 4, p. 669–697, 1991.

FLOYD, S. *Metrics for the evaluation of congestion control mechanisms*. 2008. Acessado em 18/01/2019. Disponível em: <https://www.rfc-editor.org/info/rfc5166>.

FOUNDATION, O. N. *SDN architecture*. 2014. Acessado em 20/01/2019. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf>.

GESSERT, F.; WINGERATH, W.; FRIEDRICH, S.; RITTER, N. Nosql database systems: a survey and decision guidance. *Computer Science-Research and Development*, Springer, v. 32, n. 3-4, p. 353–365, 2017.

GHOSH, M.; WANG, W.; HOLLA, G.; GUPTA, I. Morphus: Supporting online reconfigurations in sharded nosql systems. *IEEE Transactions on Emerging Topics in Computing*, IEEE, v. 5, n. 4, p. 466–479, 2015.

GILBERT, S.; LYNCH, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, ACM, v. 33, n. 2, p. 51–59, 2002.

GONZÁLEZ-APARICIO, M. T.; OGUNYADEKA, A.; YOUNAS, M.; TUYA, J.; CASADO, R. Transaction processing in consistency-aware user's applications deployed on nosql databases. *Human-centric Computing and Information Sciences*, SpringerOpen, v. 7, n. 1, p. 7, 2017.

GONZÁLEZ-APARICIO, M. T.; YOUNAS, M.; TUYA, J.; CASADO, R. A new model for testing crud operations in a nosql database. In: IEEE. *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. [S.l.], 2016. p. 79–86.

GUPTA, A.; TYAGI, S.; PANWAR, N.; SACHDEVA, S.; SAXENA, U. Nosql databases: Critical analysis and comparison. In: IEEE. *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*. [S.l.], 2017. p. 293–299.

HAAS, P. J. *Stochastic petri nets: Modelling, stability, simulation*. [S.l.]: Springer Science & Business Media, 2006.

HAILE, K.; MEKURIA, T.; FANTAHUN, A.; BELAY, A. Avoiding consistency and update latency problem in lazy master-master replication using a coordinator architecture. In: IEEE. *2017 IEEE AFRICON*. [S.l.], 2017. p. 935–937.

HALEPLIDIS, E.; PENTIKOUSIS, K.; DENAZIS, S.; SALIM, J. H.; MEYER, D.; KOUFOPAVLOU, O. *Software-defined networking (SDN): Layers and architecture terminology*. 2015. Acessado em 18/01/2019. Disponível em: <https://tools.ietf.org/html/rfc7426>.

HARRISON, G. *Next Generation Databases: NoSQLand Big Data*. [S.l.]: Apress, 2015.

HAUGHIAN, G.; OSMAN, R.; KNOTTENBELT, W. J. Benchmarking replication in cassandra and mongodb nosql datastores. In: SPRINGER. *International Conference on Database and Expert Systems Applications*. [S.l.], 2016. p. 152–166.

HEINER, M.; HERAJY, M.; LIU, F.; ROHR, C.; SCHWARICK, M. *Snoopy tool*. 2019. Acessado em 25/02/2019. Disponível em: <https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy>.

HEINER, M.; SCHWARICK, M.; WEGENER, J.-T. *Charlie tool*. 2019. Acessado em 25/02/2019. Disponível em: <https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Charlie>.

HENDAWI, A.; GUPTA, J.; JIAYI, L.; TEREDESAI, A.; NAVEEN, R.; MOHAK, S.; ALI, M. Distributed nosql data stores: Performance analysis and a case study. In: IEEE. *2018 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2018. p. 1937–1944.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1991.

JOHNSON, T. de Melo e S. M.; COUTINHO, M. M. *Avaliação de desempenho de sistemas computacionais*. [S.l.]: Grupo Gen-LTC, 2000.

JUNG, M.-G.; YOUN, S.-A.; BAE, J.; CHOI, Y.-L. A study on data input and output performance comparison of mongodb and postgresql in the big data environment. In: IEEE. *2015 8th International Conference on Database Theory and Application (DTA)*. [S.l.], 2015. p. 14–17.

JURISTO, N.; MORENO, A. M. *Basics of software engineering experimentation*. [S.l.]: Springer Science & Business Media, 2013.

KAMAL, A.; KUMAR, M. S. Consistency improvisation in mongodb during lag on secondary. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 6, n. 3, 2017.

KOMIĆ, J. Harmonic mean. *International Encyclopedia of Statistical Science*, Springer, p. 622–624, 2011.

KRISHNA, H.; ADRICHEM, N. L. van; KUIPERS, F. A. Providing bandwidth guarantees with openflow. In: IEEE. *2016 Symposium on Communications and Vehicular Technologies (SCVT)*. [S.l.], 2016. p. 1–6.

KUNDA, D.; PHIRI, H. A comparative study of nosql and relational database. *Zambia ICT Journal*, v. 1, n. 1, p. 1–4, 2017.

KUZOCHKINA, A.; SHIROKOPETLEVA, M.; DUDAR, Z. Analyzing and comparison of nosql dbms. In: IEEE. *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*. [S.l.], 2018. p. 560–564.

LAN, Y.-L.; WANG, K.; HSU, Y.-H. Dynamic load-balanced path optimization in sdn-based data center networks. In: IEEE. *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*. [S.l.], 2016. p. 1–6.

LAZOWSKA, E. D.; ZAHORJAN, J.; GRAHAM, G. S.; SEVCIK, K. C. *Quantitative system performance: computer system analysis using queueing network models*. [S.l.]: Prentice-Hall, Inc., 1984.

LIMA, M. A. de Q.; MACIEL, P. R.; SILVA, B.; GUIMARÃES, A. P. Performability evaluation of emergency call center. *Performance Evaluation*, Elsevier, v. 80, p. 27–42, 2014.

LOURENÇO, J. R.; CABRAL, B.; CARREIRO, P.; VIEIRA, M.; BERNARDINO, J. Choosing the right nosql database for the job: a quality attribute evaluation. *Journal of Big Data*, Springer, v. 2, n. 1, p. 18, 2015.

MACIEL, P. R.; LINS, R. D.; CUNHA, P. R. *Introdução às redes de Petri e aplicações*. [S.l.]: UNICAMP-Instituto de Computacao, 1996.

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

MALIK, S. U. R.; KHAN, S. U.; EWEN, S. J.; TZIRITAS, N.; KOLODZIEJ, J.; ZOMAYA, A. Y.; MADANI, S. A.; MIN-ALLAH, N.; WANG, L.; XU, C.-Z. et al. Performance analysis of data intensive cloud systems based on data management and replication: a survey. *Distributed and Parallel Databases*, Springer, v. 34, n. 2, p. 179–215, 2016.

MARSAN, M. A.; BALBO, G.; CONTE, G.; DONATELLI, S.; FRANCESCHINIS, G. *Modelling with generalized stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 1994.

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

MATHEWS, P. G. *Design of Experiments with MINITAB*. [S.l.]: ASQ Quality Press, 2005.

MINITAB, L. *Minitab Statistical Software*. 2019. Acessado em 25/02/2019. Disponível em: <http://www.minitab.com/pt-BR/default.aspx>.

MINKOFF, R.; ALLERS, R. Rei leão. *Filme. Produção de Rob Minkoff e Roger Allers. Estados Unidos, Walt Disney Pictures*, v. 89, 1994.

MONGO. *MongoDB Documentation*. 2019. Acessado em 07/01/2019. Disponível em: <https://docs.mongodb.com/manual/>.

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

MORRISON, C.; SPRINTSON, A. An in-network packet processing architecture for distributed data storage. In: IEEE. *2017 IEEE Conference on Network Softwarization (NetSoft)*. [S.l.], 2017. p. 1–5.

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

NARAYAN, S.; BAILEY, S.; DAGA, A. Hadoop acceleration in an openflow-based cluster. In: IEEE. *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. [S.l.], 2012. p. 535–538.

ONOS. *ONOS*. 2019. Acessado em 20/01/2019. Disponível em: <https://onosproject.org/>.

OPENDAYLIGHT. *OpenDaylight Project a Series of LF Projects*. 2018. Acessado em 20/01/2019. Disponível em: <https://www.opendaylight.org/>.

OPENDAYLIGHT. *Project Floodlight*. 2019. Acessado em 20/01/2019. Disponível em: <http://www.projectfloodlight.org/getting-started/>.

OSMAN, R.; PIAZZOLLA, P. Modelling replication in nosql datastores. In: SPRINGER. *International Conference on Quantitative Evaluation of Systems*. [S.l.], 2014. p. 194–209.

PANDE, P. P.; GRECU, C.; JONES, M.; IVANOV, A.; SALEH, R. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE transactions on Computers*, IEEE, v. 54, n. 8, p. 1025–1040, 2005.

PERSICO, V.; MARCHETTA, P.; BOTTA, A.; PESCAPÈ, A. Measuring network throughput in the cloud: The case of amazon ec2. *Computer Networks*, Elsevier, v. 93, p. 408–422, 2015.

PRASAD, R.; DOVROLIS, C.; MURRAY, M.; CLAFFY, K. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE network*, IEEE, v. 17, n. 6, p. 27–35, 2003.

RAUTMARE, S.; BHALERAO, D. Mysql and nosql database comparison for iot application. In: IEEE. *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*. [S.l.], 2016. p. 235–238.

RENIERS, V.; RAFIQUE, A.; LANDUYT, D. V.; JOOSEN, W. Object-nosql database mappers: a benchmark study on the performance overhead. *Journal of Internet Services and Applications*, v. 8, n. 1, p. 1, 2017.

RYU. *Ryu SDN Framework Community*. 2017. Acessado em 20/01/2019. Disponível em: <http://osrg.github.io/ryu/>.

SILVA, A. P.; BURLEIGH, S.; HIRATA, C. M.; OBRACZKA, K. A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*, Elsevier, v. 25, p. 480–494, 2015.

SILVA, F. A.; KOSTA, S.; RODRIGUES, M.; OLIVEIRA, D.; MACIEL, T.; MEI, A.; MACIEL, P. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, IEEE, v. 17, n. 5, p. 1134–1147, 2017.

SOUZA, V. C.; SANTOS, M. V. C. Maturing consolidation and performance of nosql databases-comparative study”.“amadurecimento consolidação e performance de sgbd nosql-estudo comparativo. In: *Anais do XI Simpósio de Sistemas de Informação, SBSI 2015*. [S.l.: s.n.], 2015. p. 235–238.

SWAMINATHAN, S. N.; ELMASRI, R. Quantitative analysis of scalable nosql databases. In: IEEE. *2016 IEEE International Congress on Big Data (BigData Congress)*. [S.l.], 2016. p. 323–326.

SYSTEMS; GROUP, S. E. *TimeNET tool project*. 2019. Acessado em 25/02/2019. Disponível em: <https://timenet.tu-ilmenau.de/#/>.

TABET, K.; MOKADEM, R.; LAOUAR, M. R. Towards a new data replication strategy in mongodb systems. In: ACM. *Proceedings of the 4th ACM International Conference of Computing for Engineering and Sciences*. [S.l.], 2018. p. 2.

TANG, E.; FAN, Y. Performance comparison between five nosql databases. In: IEEE. *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. [S.l.], 2016. p. 105–109.

TARASYUK, O.; GORBENKO, A.; ROMANOVSKY, A.; KHARCHENKO, V.; RUBAN, V. The impact of consistency on system latency in fault tolerant internet computing. In: SPRINGER. *IFIP International Conference on Distributed Applications and Interoperable Systems*. [S.l.], 2015. p. 179–192.

THANTRIWATTE, T.; KEPPETIYAGAMA, C. Nosql query processing system for wireless ad-hoc and sensor networks. In: IEEE. *2011 International Conference on Advances in ICT for Emerging Regions (ICTer)*. [S.l.], 2011. p. 78–82.

VENTURA, L.; ANTUNES, N. Experimental assessment of nosql databases dependability. In: IEEE. *2016 12th European Dependable Computing Conference (EDCC)*. [S.l.], 2016. p. 161–168.

WIRESHARK, F. *TShark*. 2019. Acessado em 25/02/2019. Disponível em: <https://www.wireshark.org/docs/man-pages/tshark.html>.

XIONG, P.; HACIGUMUS, H.; NAUGHTON, J. F. A software-defined networking based approach for performance management of analytical queries on distributed data stores. In: ACM. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. [S.l.], 2014. p. 955–966.

XU, L.; PAVLO, A.; SENGUPTA, S.; LI, J.; GANGER, G. R. Reducing replication bandwidth for distributed document databases. In: ACM. *Proceedings of the Sixth ACM Symposium on Cloud Computing*. [S.l.], 2015. p. 222–235.

YANG, Z.; WANG, J.; EVANS, D.; MI, N. Autoreplica: automatic data replica manager in distributed caching and data processing systems. In: IEEE. *2016 IEEE 35th International performance computing and communications conference (IPCCC)*. [S.l.], 2016. p. 1–6.

ZAFAR, S.; TARIQ, H.; MANZOOR, K. Throughput and delay analysis of aodv, dsdv and dsr routing protocols in mobile ad hoc networks. *International Journal of Computer Networks and Applications (IJCNA)*, v. 3, n. 2, p. 1–7, 2016.