Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

Modelagem de Desempenho do Banco de Dados Cassandra

Juccelino Rodrigues Alves de Barros

Recife

Maio de 2018

Juccelino Rodrigues Alves de Barros

Modelagem de Desempenho do Banco de Dados Cassandra

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Coorientador: Prof. Dr. Glauco Estácio Gonçalves

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Maio de 2018

À

meus pais, meus irmãos e minha namorada que sempre acreditaram em mim.

Agradecimentos

Primeiramente agradeço à Deus.

Agradeço especialmente ao meus pais e irmãos pelo apoio. Também agradeço à minha namorada, pela paciência nos dias estressantes e pela companhia nos melhores momentos da minha vida. Agradeço também ao grupo OS PRIMOS e os demais membros da família

Aos professores Gustavo Callou e Glauco Gonçalves, pela confiança e apoio neste trabalho. Agradeço também à toda equipe do DEINFO, professores, coordenadores e servidores. A todos amigos da UFRPE.

À TELEIN e a USTORE, empresas que me deram a oportunidade de estagiar e incrementar meu conhecimento pessoal como também profissional. Aos amigos do NTI da UFPE pelo apoio.

Também gostaria de agradecer à FACEPE e ao CNPq pelo suporte financeiro a esta pesquisa.

Por último, agradeço aos meus amigos que sempre acreditaram na realização do trabalho.

Resumo

Armazenar grande quantidade de dados é um desafio, e a demanda para gerenciar volumes massivos de dados vem crescendo. Entre os fatores que contribuem para este aumento, destacam-se as redes sociais, a Internet das coisas e a computação em nuvem. Diante disso, surgiram novos sistemas de armazenamento de dados, entre eles os bancos de dados não relacionais NoSQL. Esses bancos têm sido amplamente utilizados em empresas que gerenciam grandes quantidade de dados com intuito de atender suas demandas de escalabilidade e alta disponibilidade. Uma das principais estratégias para escolher um banco de dados é analisar o desempenho em operações típicas como inserção e consulta utilizando técnicas de medição. Entretanto, escolher o banco de dados correto para determinada aplicação utilizando esta técnica demanda tempo e investimento. Uma alternativa para avaliar o desempenho desses sistemas, com menos custo e em menor tempo, é através do uso de técnicas de modelagem de desempenho. Diante desse cenário, este trabalho realiza uma análise do desempenho de um banco de dados NoSQL utilizando técnicas de modelagem e simulação, avaliando o tempo de resposta em operações de inserção. O banco de dados NoSQL escolhido foi o Cassandra e o modelo de desempenho foi desenvolvido em rede de Petri estocástica. Para fins de validação do modelo de desempenho, um ambiente de medição foi montado utilizando a ferramenta de *benchmark Yahoo! Cloud Serving Benchmarking*. Além disso, este trabalho avalia o desempenho e o consumo de energia do Cassandra em cenários com dados distribuídos. Os resultados mostraram que o modelo de desempenho pode ser uma alternativa para avaliar o desempenho do Cassandra durante a inserção de dados.

Palavras-chave: Desempenho, Banco de dados, NoSQL, Cassandra, redes de Petri.

Abstract

Performing large amount of data storage is a challenge since the demand to manage the massive volume of data is growing. Among the reasons that provide this growing, there are social networks, Internet of things and cloud computing. In this context, new data storage systems have been emerging, for instance, non-relational databases NoSQL. These databases are been used by companies that manage massive volume of data attending their demands on scalability and high availability. One of the main strategies for choosing a database is to evaluate the performance of typical operations like insert and read data through measurement techniques. However, this technique demands time and investment to choose the suitable database for a specific application. An alternative to evaluate the performance of databases, saving investment and time, is through modeling techniques. Considering such a scenario, this work evaluates the performance of a NoSQL database using simulation and modeling techniques. In this case, the response time in insertion operations is evaluated. The NoSQL database evaluated is Cassandra and the performance model was developed in stochastic Petri net. In order to validate the performance model, measurements were conducted on the environment with the support of the Yahoo! Cloud Serving Benchmarking tool. The results demonstrate that the proposed performance model can be adopted to analyze the performance of Cassandra in data insertion process.

Keywords: Performance, Database, NoSQL, Cassandra, Petri nets.

Lista de Abreviaturas

ACID Atomicidade, Consistência, Isolamento, Durabilidade

BASE *Basically Avaliable, Soft State, Eventual Consistency*

CAP *Consistency, Availability e Partition tolerance*

CCPN Redes de Petri Colorida Condicionais

CQL *Cassandra Query Language*

CPN Redes de Petri Colorida

CTMC Cadeia de Markov de Tempo Contínuo

ECA *Event-Condition-Action*

EFM Modelo de Fluxo de Energia

NoSQL *Not Only SQL*

PN Redes de Petri

QPN *Queued Petri Nets*

RBD *Reliability Block Diagram*

SPN Redes de Petri Estocástica

SQL *Structured Query Language*

SSTable *Sorted Strings Table*

YCSB *Yahoo! Cloud Serving Benchmark*

Sumário

1 Introdução **1**

1.1 Motivação e Justificativa 4

1.2 Objetivos 5

1.3 Estrutura da Dissertação 5

2 Fundamentação Teórica **6**

2.1 Banco de Dados NoSQL 6

2.1.1 Breve histórico 8

2.1.2 Propriedade BASE 8

2.1.3 Modelos de Banco de Dados NoSQL 9

2.2 Cassandra 11

2.2.1 Elementos do Cassandra 12

2.2.2 Escrita do Cassandra 13

2.3 Técnicas de Avaliação de Desempenho 17

2.3.1 Medição 19

2.3.2 Modelagem e Simulação 21

2.4 Redes de Petri (PN) 22

2.4.1 Redes de Petri Estocásticas (SPN) 25

2.5 Considerações Finais 29

3 Trabalhos Relacionados **30**

3.1 Medição 30

3.2 Modelagem 33

3.3 Comparação dos Trabalhos Relacionados 35

3.4 Considerações Finais 36

4 Análise do *Cluster* do Cassandra **37**

4.1 Método de Medição 37

4.1.1 Ambiente de Medição 38

4.1.2 Processo de Obtenção das Amostras 39

4.2 Análise de Desempenho 41

4.3 Análise do Consumo de Energia 44

4.4 Discussão 45

4.5 Considerações Finais 47

5 Modelagem de Desempenho do Cassandra **48**

5.1 Método de Modelagem 48

5.1.1 Análise do Tempo de Resposta no Ambiente de Medição 49

5.2 Modelo de Desempenho 51

5.2.1 Parâmetros e Métrica do Modelo 53

5.3 Considerações Finais 55

6 Estudos de Caso **56**

6.1 Estudo de Caso I - Validação do Modelo 56

6.2 Estudo de Caso II 62

6.3 Estudo de Caso III 63

6.4 Estudo de Caso IV 65

6.5 Considerações Finais 67

7 Conclusão **68**

7.1 Contribuições 70

7.2 Trabalhos Futuros 71

A Descrição dos Elementos do Modelo **77**

A.1 Lugares 77

A.2 Transições 78

A.3 Parâmetros 78

B Teste T com Duas Amostras Independentes **79**

Lista de Tabelas

3.1 Características dos Trabalhos Relacionados. 36

4.1 Teste-T no tempo de execução entre *clusters* com 1 e 2 nós. 41

4.2 Análise das métricas para 4 e 6 nós em relação a 1 nó. 46

5.1 Categorias para análise de impacto dos processos internos do Cassandra no tempo de resposta das operações de escrita. 50

5.2 Funções de Habilitação do *Clock*. 53

6.1 Resultados da Medição. 57

6.2 Tempos da Transição Determinística *T0* $(1/tReq)$. 58

6.3 Impacto da Compactação no tempo de resposta. 59

6.4 Tempos das Compactações. 61

6.5 Validação do Tempo de Resposta com Resultados da Modelagem e Medição . 61

6.6 Análise do Tempo de Resposta para limite da *Memtable* igual a 1300 63

6.7 Comparação entre os estudos de caso II e IV 66

Lista de Figuras

2.1 Teorema CAP. 9

2.2 Cluster com 6 Nós do Cassandra (ACADEMY, 2017). 13

2.3 Estrutura da escrita do Cassandra. 15

2.4 Processo de Compactação (DATASTAX, 2018b). 16

2.5 Avaliação de Desempenho. 18

2.6 Arquitetura do YCSB (COOPER et al., 2010). 20

2.7 Elementos da rede de Petri: Lugar (a), Transição (b), Arco (c) e Token (d) . 23

2.8 Estados do modelo que representa um sistema cliente-servidor em rede de Petri 24

2.9 Elementos da SPN: Transição Temporizada (a) e Arco Inibidor (b). 26

2.10 Representação de um sistema de atendimento de um banco com Redes de Petri Estocástica (SPN). 27

4.1 Visão geral dos componentes do ambiente de medição. 38

4.2 Visão geral dos componentes do ambiente de medição. 39

4.3 Processo de obtenção das métricas de desempenho e consumo de energia. . . 40

4.4 Tempos de execução de acordo com o número de registros inseridos. 41

4.5 Comportamento da Vazão. 42

4.6 Comparação da vazão em relação a 1 nó para 100K e 1M de operações. . . . 43

4.7 Análise Tempo de Resposta. 43

4.8 Consumo de Energia para 10K (a), 100K (b) e 1M (c) 44

4.9 Aumento do consumo de energia em relação a 1 nó. 45

5.1 Visão Geral dos componentes do ambiente de medição. 49

5.2 Atividades realizadas para obtenção dos resultados no ambiente de medição. 51

5.3 Modelo de Desempenho Proposto. 51

6.1 Visão geral do ambiente de medição para validação do modelo. 57

6.2 Valores atribuídos em T0, T1, T2 e T3. 58

6.3 Tempos das Compactações 1 (a) e 2 (b) 60

6.4 Resultado da Modelagem e da Medição de Desempenho. 61

6.5 Redução do Tempo de Resposta com a *Memtable* 10 vezes maior. 62

6.6 Redução do Tempo de Resposta com $ssdb = 8$. 64

6.7 Redução do Tempo de Resposta com $mdb = 1300$ e $ssdb = 8$. 65

6.8 Comparação dos resultados entre as configurações analisadas. 66

Capítulo 1

Introdução

A demanda para gerenciar grandes quantidades de dados vem crescendo ao longo dos anos. Entre os fatores que aumentaram esta demanda, destacam-se as redes sociais, a Internet das coisas e a computação em nuvem. Diante disso, novos sistemas de armazenamento foram surgindo com conceitos e estratégias de armazenamento com o foco em fornecer alto desempenho em operações típicas, como inserção e consulta, e disponibilidade dos dados para os usuários. Entre os mais recentes sistemas de armazenamento de dados, estão os bancos de dados não relacionais, mais especificamente os bancos de dados *Not Only SQL* (NoSQL). Esses bancos de dados podem gerenciar grandes volumes de dados, semi-estruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade. Além disso, eles não necessariamente utilizam a *Structured Query Language* (SQL), comum nos bancos de dados relacionais (LÓSCIO; OLIVEIRA; PONTES, 2011).

Os bancos de dados NoSQL são utilizados em empresas como Facebook, Amazon e Google com o intuito de atender às suas demandas de escalabilidade, alta disponibilidade e dados não estruturados (LÓSCIO; OLIVEIRA; PONTES, 2011). Eles são facilmente escaláveis entre várias máquinas e têm o foco no desempenho em operações típicas, como inserção e consulta. Os bancos de dados NoSQL apresentam características diferentes dos bancos de dados relacionais, pois focam na disponibilização dos dados ao invés da consistência e não apresentam os padrões rígidos encontrados no modelo relacional (DIANA; GEROSA, 2010). Por ser facilmente escalável, muitos bancos de dados NoSQL são tolerantes à falhas.

Contudo, devido as suas características, alguns desses bancos sacrificaram a capacidade de realizar consultas complexas, como junções e agrupamentos de dados (COOPER et al., 2010).

Algumas das principais características dos bancos de dados NoSQL são: escalabilidade horizontal, ausência de esquema ou esquema flexível e alta disponibilidade. A escalabilidade horizontal consiste em aumentar o número de servidores, com cópias ou partes do banco de dados (LÓSCIO; OLIVEIRA; PONTES, 2011). A ausência de esquema facilita a escalabilidade, pois os dados são armazenados de forma mais flexível sem se preocupar com o formato em que o registro deve estar antes de ser inserido. A ausência também contribui para o aumento da disponibilidade. A alta disponibilidade consiste em fazer com que o banco de dados seja tolerante a falhas, visto que em um cenário com vários servidores é comum acontecer uma falha. Logo, os bancos de dados NoSQL apresentam técnicas e estratégias para prover alta disponibilidade mesmo quando os servidores apresentam problemas (DIANA; GEROSA, 2010). Outra característica importante é a elasticidade, que compreende em adicionar mais servidores com o banco de dados em execução sem afetar os demais componentes negativamente, podendo até distribuir os dados entre as instâncias do sistema de armazenamento seguindo critérios previamente definidos.

Devido a essas características, os bancos de dados NoSQL ficaram populares tanto na academia quanto na indústria. Atualmente, existem vários bancos de dados NoSQL diferentes, os quais podem ser divididos em 4 modelos: chave-valor, orientado a coluna, orientado a documentos e orientado a grafos. Cada modelo possui vantagens e desvantagens dependendo de onde for aplicado. A Seção 2.1.3 descreve sobre cada um desses modelos. O Cassandra é um exemplo de banco de dados NoSQL que é orientado a coluna. É um banco de dados distribuído massivamente escalável, criado para armazenar uma grande quantidade de dados espalhados por vários servidores oferecendo alta disponibilidade de dados consistentes (ACADEMY, 2017). A Seção 2.2 apresenta as características do Cassandra.

Entretanto, escolher o banco de dados correto para determinada aplicação demanda tempo e conhecimento entre os diversos bancos existentes. Uma das principais estratégias para escolher um banco de dados é analisar o desempenho em operações de inserção e consulta utilizando técnicas de medição. Através da medição, é possível obter, com precisão, métricas importante como vazão, tempo de resposta, tempo de execução, entre outras. Utilizando

ferramentas de *benchmarks*, é possível experimentar diferentes opções de bancos de dados e tomar decisões com base no desempenho de cada um. Contudo, a medição é fortemente dependente da infraestrutura de experimentação, seja ela montada em ambiente controlado em laboratório ou em ambiente de produção.

A realização da medição demanda tanto tempo quanto investimento. Isso ocorre porque os experimentos realizados em ambiente controlado necessitam de computadores e demais componentes que reflitam, ainda que parcialmente, o ambiente de produção. Além disso, os experimentos demandam tempo para serem realizados e seus resultados analisados. Por conta dos resultados precisos e por realizar uma análise mais próxima do ambiente real em que o sistema avaliado está inserido, a medição é bastante explorada na academia (COOPER et al., 2010), (LI; MANOHARAN, 2013), (ABUBAKAR; ADEYI; AUTA, 2014), (ABRAMOVA; BERNARDINO, 2013), (KAUR; SACHDEVA, 2017).

Diante desse cenário, uma alternativa para avaliar o desempenho desses sistemas, economizando tempo e recursos, é através da simulação desses ambientes utilizando modelos computacionais. Esta técnica pode ser uma boa alternativa, pois dessa forma se economiza tempo e se reduz investimentos. Em muitos casos, os experimentos realizados no ambiente de medição podem levar horas ou até mesmo dias para obter os resultados. Através da simulação, é possível obter métricas equivalentes às encontradas no ambiente de medição em pouco tempo. Uma vez obtido esses valores, consegue-se simular novas situações, que não são possíveis obter em um ambiente de medição, extrapolando os modelos computacionais. Com isso, é possível identificar pontos de melhorias para um determinado cenário almejado, direcionar melhor futuros investimentos, economizar tempo e otimizar o planejamento de recursos.

O uso de técnicas de modelagem e simulação em bancos de dados NoSQL pode trazer contribuições no ambiente científico e industrial, tornando-se uma alternativa mais barata na escolha do banco de dados ideal para aplicações que gerenciam grande quantidades de dados. Portanto, este trabalho propõe um modelo, utilizando redes de Petri estocásticas, para avaliação de desempenho do banco de dados Cassandra durante o processo de inserção de dados. Para fins de validação do modelo proposto, um ambiente de medição foi montado utilizando a ferramenta de *benchmark Yahoo! Cloud Serving Benchmarking* (COOPER et

al., 2010), em que o tempo de resposta para inserir registros foi analisado. Além disso, este trabalho realiza uma análise integrada de desempenho e consumo de energia do Cassandra em cenários com dados distribuídos. O objetivo desta análise é encontrar a melhor configuração do *cluster* do Cassandra quando o consumo de energia também é fator determinante na escolha do melhor cenário.

1.1 Motivação e Justificativa

A geração de dados por aplicações Web e demais serviços encontrados na Internet está aumentando cada vez mais. Em julho de 2012, apenas a Microsoft armazenou 4 trilhões de objetos, já em janeiro de 2015 esse valor já ultrapassava os 10 trilhões de objetos (AZURE, 2012). Analistas da indústria avaliam que esse mercado deve crescer saindo do faturamento de $118 bilhões em 2015 para acima dos $200 bilhões em 2018 (WORLD, 2014). Estes números tendem a aumentar cada vez mais, devido ao surgimento de novas plataformas de armazenamento de dados, redes sociais e novos dispositivos com acesso à Internet, devido à Internet das Coisas.

Diante do cenário apresentado, a principal motivação para realizar este trabalho é encontrar alternativas para avaliar banco de dados em aplicações que gerencia grande quantidade dados. Além disso, existe a necessidade de fornecer uma maneira mais rápida e barata de avaliar bancos de dados, através de técnicas de modelagem e simulação. Dessa forma, é possível diminuir os gastos com a montagem de ambientes e o tempo para coletar as amostras e realizar a análise dos resultados.

O banco de dados escolhido para realização deste trabalho foi o Cassandra, pois é um dos bancos de dados NoSQL mais utilizado conforme o *DB-Engines*[1] e que foi projetado para gerenciar grande quantidade de dados (LAKSHMAN; MALIK, 2010). Além disso, o Cassandra é utilizado em diversas aplicações conhecidas, como por exemplo o Netflix e Facebook, sendo este último o criador do Cassandra. Para o desenvolvimento do modelo computacional, foi utilizado a rede de Petri estocástica, por ser aplicada em diversas áreas, entre elas na avaliação de sistemas de armazenamento de dados.

[1] https://db-engines.com/en/ranking

1.2 Objetivos

Essa dissertação tem como objetivo avaliar o desempenho do banco de dados Cassandra através de técnicas de modelagem e medição. Em particular, foi avaliado o desempenho durante a inserção de dados. A pesquisa foi dividida em etapas, que iniciou-se no entendimento do banco de dados, sua arquitetura, componentes e estratégia para armazenar dados até a validação do modelo através de resultados obtidos pela medição do banco de dados instalado em um ambiente de testes em laboratório, seguida pela aplicação do modelo em configurações do banco não testadas no ambiente de medição.

De forma mais específica os objetivos deste trabalho são:

- Analisar o desempenho do banco de dados Cassandra em ambiente montado em laboratório com uso de ferramenta de *benchmark*;
- Propor modelo de desempenho do banco de dados Cassandra que capture aspectos do comportamento interno do banco;
- Simular situações não observadas no ambiente de medição através do modelo desenvolvido;

1.3 Estrutura da Dissertação

Esta dissertação está organizada como segue. O Capítulo 2 apresenta a fundamentação teórica. O Capítulo 3 mostra os trabalhos relacionados. O Capítulo 4 realiza uma análise integrada de desempenho e consumo de energia do *Cluster* do Cassandra. O Capítulo 5 realiza a modelagem de desempenho do Cassandra, em que se apresenta o modelo proposto para realizar a análise de desempenho do Cassandra durante a operação de inserção. O Capítulo 6 apresenta um conjunto de estudos de caso realizados com o modelo, com o objetivo de avaliar o Cassandra. Por fim, o Capítulo 7 conclui a dissertação e descreve direcionamentos para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo explana os conceitos e conhecimentos necessários para a fundamentação teórica deste trabalho. O capítulo está dividido da seguinte forma: a primeira seção descreve os bancos de dados NoSQL, apresentando um breve histórico, a propriedade *Basically Available, Soft State, Eventual Consistency* (BASE) e os modelos de bancos de dados NoSQL. As características, componentes e o processo de inserção do banco de dados Cassandra serão descritos na Seção 2.2. Em seguida, a Seção 2.3 apresentará as principais técnicas de avaliação de desempenho, incluindo alguns objetivos e conceitos importantes sobre esta abordagem. Por fim, as características e conceitos da rede de Petri serão apresentadas na Seção 2.4.

2.1 Banco de Dados NoSQL

Um dos motivos para o surgimento de novos paradigmas e tecnologias de armazenamento de dados foi atender a grande demanda de volume de dados gerados por aplicações, principalmente aplicações WEB, serviços de armazenamento em nuvem e de demais dispositivos conectados à Internet. Os bancos de dados não relacionais surgiram com o objetivo de atender essa demanda, com foco em gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade (LÓSCIO; OLIVEIRA; PONTES, 2011).

Os bancos de dados não relacionais não apresentam todas as características referentes à Atomicidade, Consistência, Isolamento, Durabilidade (ACID), encontradas nos bancos de dados relacionais. Os bancos de dados orientado a objetos e os bancos de dados NoSQL são exemplos de bancos que se encaixam nesse contexto (ANICETO; XAVIER, 2014). Os bancos de dados NoSQL apresentam um conjunto de conceitos que permitem o processamento de dados de forma rápida e eficiente com o foco em desempenho. É uma alternativa para modelar dados sem se preocupar com os padrões rígidos proposto pelo modelo relacional.

Os banco de dados NoSQL têm uma estrutura distribuída e tolerante a falhas que se baseia na redundância de dados em vários servidores. Em consequência disso, o sistema pode ser escalado facilmente agregando mais servidores, e assim a falha de um deles pode ser tolerada. Bancos NoSQL são projetados para trabalhar com uma grande quantidade de dados distribuídos. Para isso, algumas características são bem comuns nesses bancos (LÓSCIO; OLIVEIRA; PONTES, 2011):

- **Escalabilidade Horizontal:** consistem em cada aplicação ser capaz de aumentar o número de nós (máquinas) sem que afete o sistema negativamente.
- **Ausência de esquema ou esquema flexível:** a ausência de esquema facilita a escalabilidade, pois os dados são armazenados de forma mais flexível sem se preocupar com o formato em que o registro deve estar antes de ser inserido. Porém não dá garantias quanto à integridade dos dados
- **Suporte nativo à replicação:** permite a replicação de forma nativa, diminuindo o tempo para recuperar informações.

Essas características fizeram com que os bancos de dados NoSQL ficassem populares quando se trata de armazenar grandes volumes de dados. A utilização de bancos de dados NoSQL se dá principalmente, devido ao aumento gradativo de informações a serem armazenadas, no qual o desempenho é prejudicado e o tempo de resposta se torna um fator preocupante.

2.1.1 Breve histórico

O termo NoSQL foi usado pela primeira vez em 1998 para citar um banco de dados relacional *open-source* que omitia o uso da linguagem SQL, o Strozzi NoSQL, criado por Carlo Strozzi (FOWLER, 2012). O nome veio pelo fato que o banco de dados utilizava *shell script* como linguagem de consulta, não a SQL. Hoje, o uso do termo não se refere mais ao banco desenvolvido por Strozzi. Em 2009, na conferência "NoSQL Meetup", o termo foi utilizado novamente, mas agora referenciando os bancos de dados não relacionais (STRAUCH; SITES; KRIHA, 2011).

Na conferência ficou claro que o uso do Amazon's Dynamo e do Google's Bigtable, precursores dos primeiros bancos de dados NoSQL, estava crescendo. Hoje existem vários tipos de bancos de dados NoSQL diferentes, que podem ser dos seguintes modelos: chave-valor, orientado a colunas, orientado a documentos e orientado a grafos. Esses modelos serão detalhados na Seção 2.1.3.

2.1.2 Propriedade BASE

Diferente dos bancos de dados relacionais, os bancos NoSQL não se baseiam na propriedade ACID e sim na propriedade BASE. A propriedade BASE (*Basically Avaliable, Soft State, Eventual Consistency*) significa basicamente disponível, estado leve e consistência eventual.

A consistência eventual é uma característica dos bancos NoSQL relacionada ao fato da consistência nem sempre ser mantida entre os diversos pontos de distribuição de dados. Essa característica é baseada no teorema *Consistency, Availability e Partition tolerance* (CAP) que diz que em um dado momento, só é possível garantir duas das três propriedades entre a consistência, disponibilidade e a tolerância à partição (DIANA; GEROSA, 2010). A Figura 2.1 mostra uma representação do teorema CAP. Logo, para um sistema ser tolerante a partição e disponível, por exemplo, ele vai ter que abrir mão da disponibilidade, representando o espaço DT da Figura 2.1.

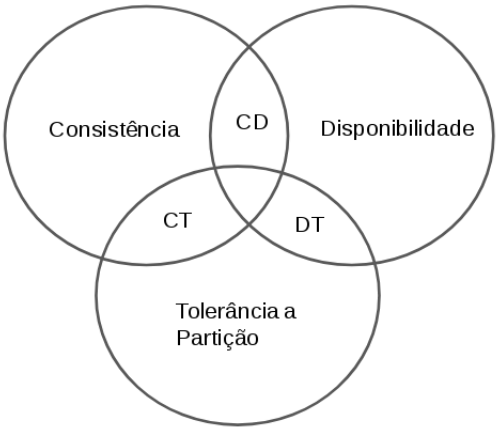


Figura 2.1: Teorema CAP.

A consistência diz respeito à preservação da integridade dos dados ao final da execução das instruções proveniente das aplicações clientes. Disponibilidade é a probabilidade de o sistema se encontrar apto a responder a todas as requisições. A tolerância à partição é a propriedade de um sistema continuar funcionando mesmo quando um problema ocorre na rede dividindo o sistema em uma ou mais partições.

Baseada nesses conceitos, a propriedade BASE indica que se deve planejar um sistema de forma a tolerar inconsistências temporárias quando se quer priorizar a disponibilidade (DIANA; GEROSA, 2010). A disponibilidade da propriedade BASE é garantida tolerando falhas parciais no sistema, sem que o sistema todo falhe. Por exemplo, se um banco de dados está particionado em cinco nós e um deles falha, apenas os clientes que acessam aquele nó serão prejudicados, pois o sistema como todo não irá interromper seu funcionamento (SOARES; BOSCARIOLI, 2013).

2.1.3 Modelos de Banco de Dados NoSQL

Atualmente é possível encontrar diversos modelos de bancos de dados NoSQL. Entre os modelos de dados mais importantes, encontram-se os modelos de chave-valor, orientado a coluna, orientado a documentos e orientado a grafos. Cada modelo possui vantagens e desvantagens dependendo de onde for aplicado. É importante lembrar que cada modelo de dados tem formas diferentes de armazenamento e consulta. A seguir será detalhado cada modelo de dados.

No modelo **chave-valor**, o sistema armazena dados estruturados como pares de chaves e valores. Uma chave é um identificador para diversos valores, que podem ser expressos por índices *hash*. Dessa forma, é o modelo de estrutura mais simples. Inserções de dados e consultas nesse modelo são realizadas intrinsecamente sobre as chaves (CARNIEL et al., 2012). A desvantagem deste modelo é que não é possível realizar consultas mais complexas (LÓSCIO; OLIVEIRA; PONTES, 2011), utilizando junções e agrupamentos entre os dados consultados. Exemplos de banco de dados que seguem o modelo chave-valor são: Redis[1], Riak[2] e o Dynamo[3], este último foi criado pela Amazon e foi usado como base para o desenvolvimento do Cassandra.

No modelo **Orientado a Coluna**, os dados são armazenados em colunas de uma tabela. Porém, diferentemente do modelo relacional, essas tabelas não possuem relacionamento e são armazenadas separadamente. Portanto, cada coluna é exclusivamente independente de cada tabela. Além disso, as colunas possuem índices padrões e forma de compressão dos dados para melhorar o processamento de consultas e o armazenamento (CARNIEL et al., 2012). Aqui existe o conceito de *Column Family* (Família de Colunas), que é usado com o intuito de agrupar colunas que armazenam os mesmos tipos de dados. Alguns exemplos de bancos deste tipo são o Hbase[4] e o Cassandra[5], este último será detalhado na Seção 2.2.

Já no **Orientado a documentos**, o armazenamento dos dados ocorre em coleções de documentos. Um documento é um objeto com um identificador único e um conjunto de campos, que podem ser textos, listas ou outros documentos. Neste modelo, encontra-se um conjunto de documentos em que cada um contém um conjunto de campos (chaves) e o valor deste campo. O modelo não depende de um esquema rígido, ou seja, não existe uma estrutura fixa como nos bancos relacionais. Esta flexibilidade é uma das principais características deste modelo. Exemplos de bancos deste tipo são CouchDB[6] e MongoDB [7].

[1]http://redis.io/
[2]http://basho.com/products/
[3]http://aws.amazon.com/pt/documentation/dynamodb/
[4]http://hbase.apache.org/
[5]http://cassandra.apache.org/
[6]http://couchdb.apache.org/
[7]https://www.mongodb.org/

O modelo **Orientado a grafos** possui 3 componentes básicos: os nós (vértices dos grafos), os relacionamentos (as arestas), e os atributos. Neste caso, o banco de dados pode ser visto como um conjunto de grafos rotulado e direcionado. A vantagem de utilização do modelo baseado em grafos é permitir a execução rápida de consultas complexas. Exemplos de bancos: Neo4j[8] e AllegroGraph[9].

Cada modelo de banco de dados NoSQL organiza o armazenamento seguindo uma estratégia diferente, mais apropriada para uma determinada aplicação ou outra. Por exemplo, para manipulação de dados estatísticos, frequentemente escritos mas raramente lidos, pode-se utilizar um banco de dados do tipo chave e valor ou um banco orientado a documentos. Caso exista a necessidade de uma aplicação com alta disponibilidade, recomenda-se utilizar um banco de dados orientado a colunas como o Cassandra, por exemplo. Para aplicações que necessitam de alto desempenho de consultas e com muitas junções, o ideal é usar um banco de dados orientado a grafos (CARNIEL et al., 2012). Fica claro conhecer a importância de cada um desses modelos, pois dependendo da aplicação em questão, pode-se obter bons resultados ou não no quesito gerenciamento de dados.

2.2 Cassandra

O Cassandra é um banco de dados distribuído massivamente escalável, criado para armazenar uma grande quantidade de dados espalhados por vários servidores e oferecer bom desempenho a dados consistentes (LAKSHMAN; MALIK, 2010). Avinash Lakshman e Prashant Malik, na época funcionários do Facebook, criaram esse banco de dados que foi lançado em 2008 como um projeto *open source*. Em 2009, foi adotado pela *Apache Software Foundation*[1].

O Cassandra foi baseado em outros 2 bancos NoSQL, o Dynamo da Amazon e o BigTable da Google. A arquitetura foi projetada se baseando no banco de dados Dynamo, enquanto o modelo de dados foi baseado no banco de dados BigTable. Mesmo que seu modelo de dados seja voltado a coluna, o Cassandra permite consultas como no modelo chave-valor,

[8]http://neo4j.com/
[9]http://franz.com/agraph/allegrograph/
[1]https://www.apache.org/

podendo ser considerado um modelo híbrido. As principais característica do Cassandra são: distribuído, descentralizado, escalável, altamente disponível, tolerante a falhas e alto desempenho. Em outras palavras, o Cassandra armazena dados em várias máquinas, através de uma distribuição em que todos os componentes realizam as mesmas tarefas, ou seja, não existe a abordagem *master-slave* (mestre-escravo). Dessa forma, o Cassandra fornece alta disponibilidade dos dados e bom desempenho em operações de leitura e escrita em uma estrutura tolerante a falhas.

O Cassandra é utilizado no Facebook para otimização do sistema de busca e gerenciamento de mensagens dos usuários (LAKSHMAN; MALIK, 2010). Também é utilizado para dar suporte à replicação, detecção de falhas e armazenamento de cache.

O Twitter utiliza o Cassandra para armazenar resultados da mineração de dados realizada sobre a base de usuários, resultados de *trend topics*, *@toptweets* e análises em tempo real em larga escala. A utilização do Cassandra trouxe vantagens tanto na implementação da modelagem dos dados relacionados *tweets, timeline*, entre outros, como no desempenho com relação aos campos de busca de usuários ou por palavras-chaves. Também aumentou a disponibilidade dos serviços. A empresa Netflix, que fornece serviço de *streaming* de vídeos, migrou toda suas informações de um banco de dados relacional para o Cassandra (NETFLIX. . . , 2011). Outras grandes empresas também utilizam o Cassandra, como Ebay, Cisco e Spotify (LABS, 2014).

2.2.1 Elementos do Cassandra

Os principais elementos do Cassandra são: Cluster, Nó, Data Center, Keyspaces, famílias de colunas, tabelas, colunas e linhas. O *Cluster* é uma coleção de *Datacenter*. O *Datacenter* é um grupo de nós ou apenas um nó que pertencem ao mesmo *cluster*, não correspondendo necessariamente a um *datacenter* físico. Enquanto que o nó é uma instância física do Cassandra onde se armazena os dados, ou seja, uma máquina que esteja executando o Cassandra.

A arquitetura do *Cluster* é do tipo anel, onde todos os nós se comunicam entre si, não existe a abordagem do tipo "*master-slave*" como também não existe a distribuição de tarefas exclusivas para um nó específico, ou seja, todos os nós armazenam dados e podem executar

as instruções requisitadas pela aplicação cliente. Dessa forma, o Cassandra não apresenta um ponto comum de falha. A Figura 2.2 ilustra um *Cluster* com 6 nós do Cassandra.

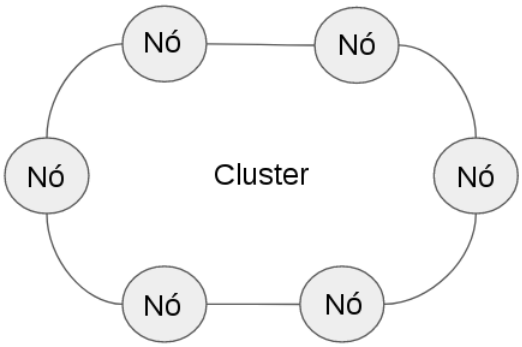


Figura 2.2: Cluster com 6 Nós do Cassandra (ACADEMY, 2017).

O *Keyspace* é o agrupamento de dados, similares ao banco de dados no modelo relacional. Porém, diferentemente do que ocorre no modelo relacional, o *Keyspace* possui informações como o fator de replicação e a estratégia de armazenamento. As famílias de colunas ou tabelas são agrupamento de colunas ordenadas por nome e pesquisada por linha. A Coluna é a menor unidade para armazenar dados, sendo composta pelos campos: nome, valor e um campo que registra o momento de alguma alteração do dado. Esse último campo é importante, pois é uma estratégia utilizada para gerenciar a consistência desse dado a ser consultado. Uma coluna em uma família de colunas deve ter, pelo menos, em cada um do seus campos uma chave primária, chamada de *row key*.

O Cassandra possui uma linguagem própria para executar as instruções no banco de dados chamada *Cassandra Query Language* (CQL), muito semelhante à SQL. A diferença entre as duas linguagem é que a CQL é mais simplificada e não suporta alguns recursos como junções ($JOIN$) e agrupamentos ($GROUP\ BY$).

2.2.2 Escrita do Cassandra

O Cassandra emprega diversas estratégias que podem impactar o desempenho para a escrita e leitura de dados. Entre as estratégias encontram-se o fator de replicação e o nível de consistência do banco de dados para cada operação.

O fator de replicação é um número inteiro definido durante a criação do *Keyspace* e consiste em executar a mesma operação de inserção em diferentes nós do *Cluster* com o objetivo de aumentar a disponibilidade do banco de dados. Quanto maior o fator de replicação menor será o desempenho da escrita do Cassandra, contudo maior será o desempenho da operação de leitura do registro. O nível de consistência é definido para cada operação (leitura e escrita). Ele indica a quantidade de *nós* que devem responder com sucesso determinada operação realizada antes de enviar a resposta para aplicação cliente. Quanto menor o nível de consistência maior será o desempenho da operação. Diante das diversas estratégias encontradas nas operações de escrita e leitura no Cassandra, este trabalho detalha somente a operação de escrita.

A escrita no Cassandra passa por algumas etapas. Quando uma escrita ocorre, o Cassandra armazena o registro na memória RAM, em uma estrutura chamada *Memtable*, enquanto a instrução de escrita é gravada em outra estrutura no disco chamada *Commit log*. O *Commit log* é importante, pois garante a durabilidade da inserção. Dessa forma, caso ocorra alguma falha no Cassandra, os dados que estão na memória são perdidos. Porém, quando o Cassandra for inicializado novamente, verifica-se as informações no *Commit log*, atualizando os dados se necessário (DATASTAX, 2018b).

Quando a *Memtable* atinge um limite de tamanho em *bytes* pré-configurado (*memtable_heap_space_in_mb* e *memtable_offheap_space_in_mb*) acontece o processo de *flush*, em que os dados são descarregados da *Memtable* para outra estrutura em disco chamada de *Sorted Strings Table* (SSTable). A *SSTable* é uma estrutura imutável, ou seja, é impossível adicionar, remover ou alterar informações. Além da *SSTable*, o *flush* também gera um conjunto de arquivos com informações dos índices dos registros e detalhes do processo (tempo de *flush*, quantidade de registros descarregados e etc.) Após o *flush*, a *Memtable* é apagada e o *Commit log* é reciclado. Para aumentar a tolerância a falhas, é necessário diminuir a memória da *Memtable* para que os dados sejam escritos mais rapidamente na *SSTable*. Contudo, isto diminui o desempenho da inserção. A Figura 2.3 ilustra a estrutura da escrita do Cassandra.

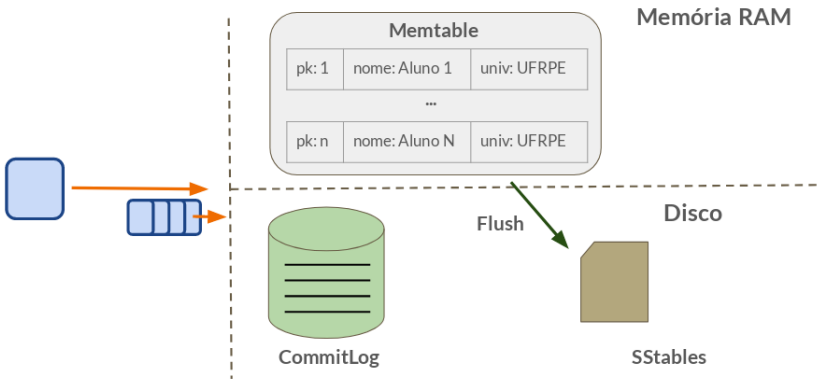


Figura 2.3: Estrutura da escrita do Cassandra.

Periodicamente ocorre o processo de compactação das *SSTable*. Este processo realiza a mesclagem de várias *SSTable* gerando uma nova *SSTable* com os registros mais atualizados. O processo de compactação é importante para economizar espaço em disco e manter a consistência dos dados, visto que instruções de *Update* no Cassandra criam novos registros ao invés de sobrescrever os registros já existentes. Dessa forma, quando uma compactação ocorre, é verificado o *timestamp* do registro, permanecendo o mais recente. Além disso, quando uma instrução de *Delete* ocorre, o registro é marcado para ser apagado através da flag *Tombstone*. Durante a compactação, este registro é removido. A Figura 2.4 ilustra o processo de compactação. Observa-se que uma *SSTable* possui 2 registros com mesmo identificador, representados pela cor vermelha. Neste caso, ocorre a mesclagem, onde é verificado o *timestamp* dos registros, permanecendo o mais recente.

O Cassandra possui 3 estratégias de compactação. O processo de definição de qual estratégia a ser adotada ocorre durante a criação de uma tabela e pode ser escolhida entre: *SizeTieredCompactionStrategy*, *DateTieredCompactionStrategy* e *LeveledCompactionStrategy*(DATASTAX, 2018a).

A **SizeTieredCompactionStrategy (STCS)** é a estratégia definida por padrão durante a criação da tabela. Esta estratégia consiste em iniciar o processo de compactação quando existir um número fixo, que pode ser parametrizado (por padrão é 4), de *SSTable* com tamanho similares. Já **DateTieredCompactionStrategy (DTCS)** consiste em realizar a compactação baseada no *timestamp* dos registros, ou seja, o Cassandra pode gerar novas *SSTable* baseadas nos tempos dos registros. Por exemplo, o Cassandra realiza a com-

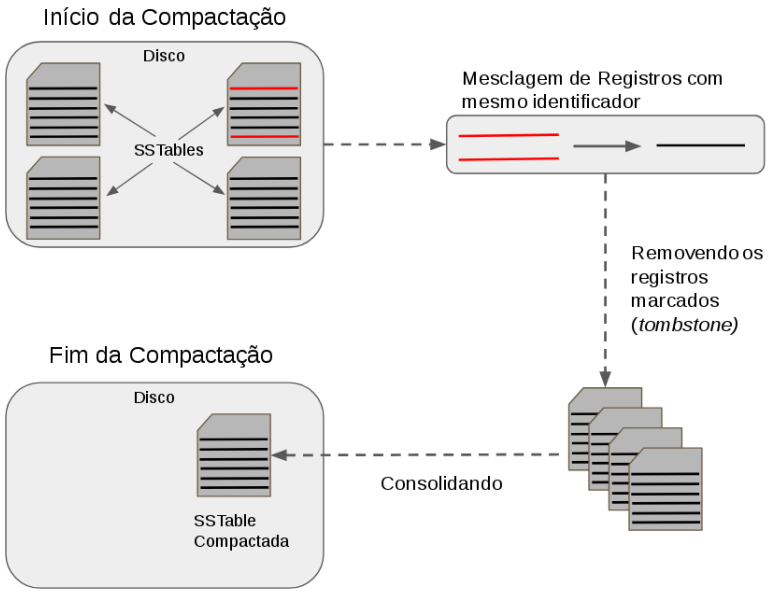


Figura 2.4: Processo de Compactação (DATASTAX, 2018b).

pactação dos registros inseridos na última hora em uma *SSTable* e os registros das últimas 4 horas em outra *SSTable*. Por último, a **LeveledCompactionStrategy (LCS)** cria *SSTable* de tamanho fixo que são agrupadas em níveis. Dentro do nível, as *SSTable* são protegidas para não serem sobrepostas. Cada nível (L0, L1, L2...) é 10 vezes maior que o anterior.

Cada estratégia tem o objetivo de prover alto desempenho, dependendo de como será utilizado o banco de dados (muitas operações de inserção ou muitas operações de leitura, etc.). Por exemplo, a estratégia *LeveledCompactionStrategy* oferece alto desempenho para operações de leitura, enquanto a *DateTieredCompactionStrategy* oferece bom desempenho para operações de leitura dos registros mais recentes. A estratégia padrão, *SizeTieredCompactionStrategy*, oferece bom desempenho para operações de escrita (DATASTAX, 2018b). Este trabalho focou na estratégia de compactação **SizeTieredCompactionStrategy**, por ser focado no desempenho nas operações de escrita e por ser definida por padrão durante a criação da tabela.

2.3 Técnicas de Avaliação de Desempenho

O desempenho é um dos critérios mais importante na concepção e aquisição de um sistema (BUKH, 1992). Um dos objetivos de quem desenvolve um sistema é fornecer alto desempenho no menor custo possível. Dessa forma, é necessário conhecer as principais métricas de desempenho, com a finalidade de se encontrar os limites do sistema e ter insumos suficiente para tomar decisões referentes ao comportamento em situações de sobrecarga.

Para realizar a avaliação de desempenho de um sistema, é necessário escolher um conjunto de métricas. Entre as formas de escolher essas métricas, pode-se listar os serviços fornecidos pelo sistema. Para cada solicitação de serviço feita ao sistema, há vários resultados possíveis. Por exemplo, um banco de dados oferece o serviço de responder a consultas. Quando executada uma consulta, o banco pode responder corretamente, responder de forma incorreta ou não responder. Se o banco de dados executa a consulta corretamente, o seu desempenho é medido pelo tempo necessário para realizar o serviço, pela taxa de leitura na qual o serviço é executado e os recursos consumidos durante a execução do serviço (ARAÚJO, 2016). Essas três métricas se referem a capacidade de resposta (BUKH, 1992).

Entre os vários objetivos que a avaliação de desempenho fornece, pode-se listar os seguintes (LILJA, 2005): comparar alternativas, analisar impacto, ajuste do sistema, identificar o desempenho relativo e definir expectativas. **Comparar alternativas** consiste em realizar uma análise comparativa baseada na necessidade dos usuários. É muito comum realizar comparações entre banco de dados em relação às operações típicas, como inserção e consulta. A partir dessa análise, é possível obter informações quantitativas a respeito das principais métricas como vazão e tempo de resposta.

A **análise de impacto** implica em conhecer o impacto do sistema quando este for alterado. Diante disso, é importante ter conhecimento do tamanho do impacto ao realizar alterações. Já o **ajuste do sistema** consiste em combinar parâmetros do sistema a fim de encontrar a melhor combinação para determinada aplicação, levando em consideração critérios de desempenho. A **identificação do desempenho relativo** é quando se realizam comparações entre elementos. Analisar o desempenho relativo em sistemas é uma prática bem comum e tem como objetivo identificar pontos fortes e fracos de sistemas da mesma

categoria. Por último, através da **definição de expectativas** espera-se encontrar resultados por meio da análise de desempenho no tocante ao comportamento do sistema em situações preestabelecidas.

Existem várias formas de realizar a análise de desempenho conforme ilustra a Figura 2.5 (CALLOU et al., 2011). Dependendo das condições de tempo e recursos, utilizar mais de uma técnica pode acarretar no melhor entendimento do sistema e fornecer insumos para melhores tomadas de decisão. Com auxílio de ferramentas de modelagem, simulação e medição é possível obter informações detalhadas do sistema em questão.

Utilizando-se técnicas de medição é possível obter resultado mais preciso, pois é a técnica que mais se aproxima do ambiente real analisado. Contudo, a modelagem apresenta várias alternativas para avaliação de desempenho, podendo ser por simulação, através de Sistemas de Eventos Discretos (SED) ou modelagem analítica, que também se aproxima do ambiente real. Neste trabalho foram utilizadas técnicas de modelagem e simulação utilizando redes de Petri estocásticas e técnicas de medição, através de ferramentas de *benchmark*, com o objetivo de validar o modelo proposto.

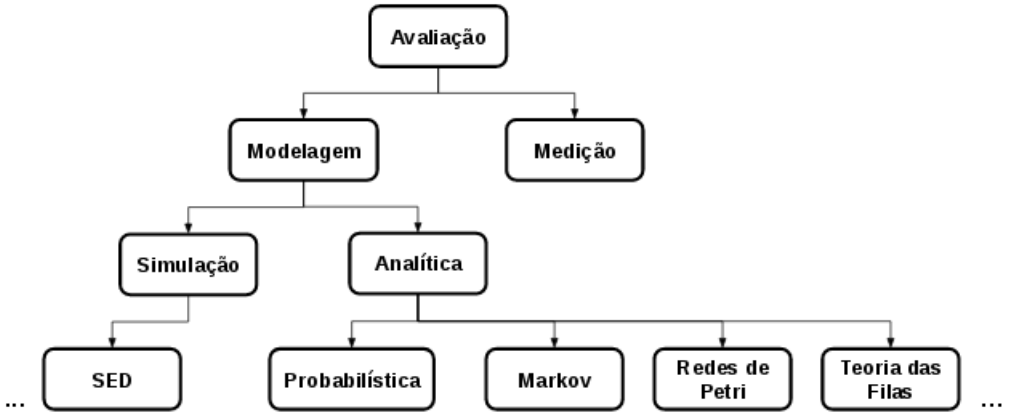


Figura 2.5: Avaliação de Desempenho.

2.3.1 Medição

A análise de desempenho através da medição é uma das técnicas mais comuns de avaliar sistemas. Na medição, pode-se gerar uma carga de trabalho para o sistema processar enquanto se realiza o monitoramento do seu comportamento (BUKH, 1992). A escolha da carga de operações que serão submetidas ao sistema deve ser feita com bastante cuidado, visto que uma escolha errada pode gerar resultados que não demonstrem o comportamento típico do sistema.

Normalmente, muitos analistas de desempenho utilizam ferramentas de *benchmark* para realizar as devidas análises nos sistemas. Essas ferramentas são responsáveis por fazer o sistema executar um conjunto determinado de operações e monitorar o comportamento fornecendo informações durante a execução do experimento. Algumas dessas ferramentas também fornecem resultados estatísticos e valores de métricas de desempenho. A análise de desempenho por medição consome tempo e recursos, uma vez que para realizar os experimentos são necessários a montagem do laboratório com os componentes do sistema (computadores, *switch*, monitores e etc.) e tempo para obter os resultados das amostras.

Durante o desenvolvimento deste trabalho, empregou-se, para medição, a ferramenta *Yahoo! Cloud Serving Benchmark* (YCSB) (COOPER et al., 2010) para avaliação de desempenho de banco de dados. Esta ferramenta é bem difundida na comunidade acadêmica e foi utilizada para obter o tempo de resposta do ambiente de medição, utilizado para validar o modelo SPN (ver Seção 5.2).

Yahoo! Cloud Serving Benchmark (YCSB). A ferramenta de *benchmark* YCSB foi desenvolvida pela *Yahoo!* em conjunto com vários membros dos mais variados sistemas de armazenamento de dados. Os objetivos dessa ferramenta são: fornecer uma ferramenta padrão para auxiliar na escolha do banco de dados, facilitar a comparação de desempenho provendo as principais métricas e automatizar o ambiente de medição. É uma ferramenta de código aberto[2], desenvolvida em Java e com suporte para vários bancos de dados. A principal característica do YCSB é a sua extensibilidade. Por meio desta ferramenta é possível criar cargas de trabalho por arquivos de configuração e experimentar diferentes bancos de dados escrevendo novas classes e aproveitando os métodos e as interfaces já desenvolvidos

[2]https://github.com/brianfrankcooper/YCSB/wiki

(COOPER et al., 2010).

O YCSB possui duas fases de execução. A primeira fase é denominada de *load*, quando dados são inseridos. Nesta fase é definida a quantidade de dados inseridos no banco de dados. A outra fase é a *transaction*, que é executada através do parâmetro *run*, e consiste em realizar operações de leitura e atualização. Em ambas as fases é possível determinar a quantidade de usuários simultâneos, a carga de trabalho, o banco de dados e definir uma vazão fixa. Após a execução de cada fase, o YCSB computa diversas métricas, entre elas: o tempo de execução, a vazão média, o tempo de resposta (latência) das requisições e a quantidade de operações bem sucedidas e que falharam.

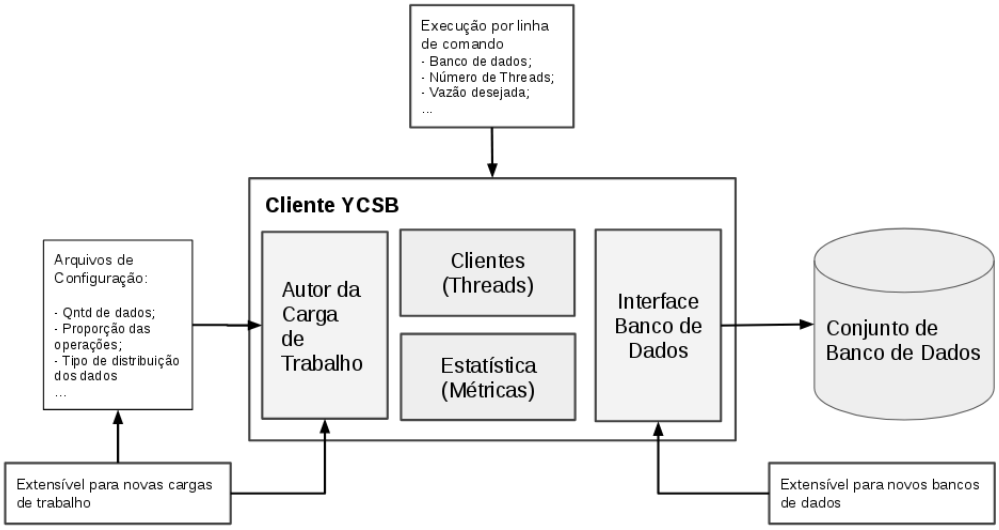


Figura 2.6: Arquitetura do YCSB (COOPER et al., 2010).

A Figura 2.6 ilustra a arquitetura do YCSB que é dividida em 4 camadas: Autor da Carga de Trabalho, Clientes (*Threads*), Estatística (Métricas) e a Interface do Banco de Dados. O Autor da Carga de Trabalho é definida por arquivos de configurações (*".properties"*), em que é possível definir a quantidade de dados que serão utilizados nas operações de leitura, escrita ou atualização, o tamanho de cada registro e outros parâmetros. Esta camada é extensível, visto que é possível criar novos arquivos de configurações de acordo com a necessidade da avaliação. A camada do Cliente é responsável por definir atributos necessários para execução do experimento, via linha de comando. É possível definir a quantidade de *threads*, fixar a vazão da ferramenta e os parâmetros de conexão com o banco de dados. A camada de Estatística é responsável por realizar os cálculos das métricas de desempenho, como tempo

de execução e tempo de resposta. A Interface do Banco de Dados apresenta um conjunto de classes e métodos responsáveis por realizar a comunicação com o banco de dados. Esta interface também é extensível, visto que pode adicionar novos bancos de dados aproveitando os métodos existentes nessas classes.

2.3.2 Modelagem e Simulação

Uma alternativa em avaliar o desempenho do sistema é através de técnicas de modelagem e simulação. O desenvolvimento de um modelo que represente o comportamento de um sistema real requer muito conhecimento prático sobre o sistema em questão. A partir do modelo desenvolvido, é possível aplicar técnicas de simulação e análise estacionária com o objetivo de prover métricas previamente definidas.

O desenvolvimento do modelo de desempenho passa por algumas etapas, entre elas a verificação e a validação (BUKH, 1992). A verificação consiste em analisar se o modelo está realizando o fluxo correto de operações, tal qual o sistema real analisado executa. Esta etapa é muito importante, pois pode-se encontrar possíveis gargalos. Após a verificação do modelo tem-se a etapa de validação, que consiste em validar as métricas desejadas. A validação é feita comparando os resultados obtidos pelo modelo, podendo ser através de simulação ou análise estacionária, com os resultados encontrados na medição. Além do valor médio das métricas, também são comparados os intervalos de confiança das médias encontrados na simulação e medição, visto que essas métricas muitas vezes apresentam variações em relação a média. Uma vez verificado e validado o modelo, é possível extrapolar o modelo realizando análises de situações que sejam muito custosas de obter em ambiente de medição em escala reduzida ou que demandem a parada de sistemas em produção. Muitas vezes, se utilizam técnicas de simulação desses modelos a fim de analisar as métricas em situações diferentes.

A simulação é a execução de um modelo que procura reproduzir o comportamento ao longo do tempo do sistema que ele representa. Uma simulação é conduzida por meio da geração estocástica de entradas que são submetidas ao modelo do sistema para observar como o seu comportamento e de seus componentes impactam nas métricas em estudo. Atualmente, é possível realizar simulações e análise estacionárias de modelos em redes de Petri por meio

de *softwares* especializados. Entre eles, podem-se destacar o *TimeNet*[3] e o *Mercury*[4] (SILVA et al., 2015). Estes *softwares* permitem calcular métricas de desempenho utilizando vários conceitos, entre eles, Teoria das Filas. O *Mercury* permite também realizar outras formas de modelagem como Cadeia de Markov de Tempo Contínuo (CTMC)(BOLCH et al., 2006), *Reliability Block Diagram* (RBD) (BOLCH et al., 2006) e Modelo de Fluxo de Energia (EFM) (CALLOU et al., 2014).

2.4 Redes de Petri (PN)

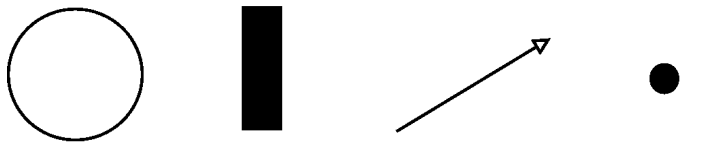
As Redes de Petri (PN) foram introduzidas em 1962 por Carl Adams Petri (PETRI, 1962), na tese de doutorado na *Technical University of Darmstandt*, Alemanha. O objetivo inicial da rede de Petri era modelar e analisar sistemas de comunicação (CALLOU et al., 2011). Contudo, a rede de Petri é uma ferramenta gráfica e formal que fornece mecanismos para modelar e analisar vários tipos de sistemas, sejam eles paralelos, concorrentes, assíncronos ou não-determinísticos (MURATA, 1989).

Através das redes de Petri é possível encontrar informações úteis sobre a estrutura do sistema, bem como encontrar gargalos e prover soluções alternativas dos sistemas modelados (SOUSA, 2015). Devido à sua alta aplicabilidade, as redes de Petri possuem diversas extensões tais como redes temporizadas (MERLIN; FARBER, 1976), estocásticas (MARSAN, 1988), alto-nível (JENSEN, 1989) e orientadas a objetos (JANOUŠEK, 1998).

A rede de Petri elementar é composta por 4 elementos (ANDRADE, 2014), são eles: lugares, transições, arcos, e *tokens*. Os lugares são representados por círculos e têm a função de representar os possíveis estados do sistema. As transições são representadas por barras e são responsáveis por mudar o estado do sistema, sendo úteis para modelar eventos internos ou externos ao sistema. A mudança de estado pode acontecer quando um conjunto de condições são satisfeitas (regra de habilitação), fornecendo assim a possibilidade do disparo da transição habilitada. Os arcos representam o fluxo do sistema, são ilustrados por setas e conectam o lugar a uma transição e vice-versa. O lugar só pode ser conectado à transição, podendo

[3]https://timenet.tu-ilmenau.de/template/index
[4]http://www.modcs.org/?page_id=2330

existir vários lugares conectados a uma única transição ou várias transições conectadas a um único lugar. Os arcos possuem pesos para representar a quantidade de *tokens* necessários no lugar de origem do arco para habilitar a transição. Por último, a marcação da rede de Petri é definida pelos *tokens* no modelo. Sendo assim, os *tokens* representam o estado em que o sistema se encontra no momento, eles ficam nos lugares e podem ser adicionados ou consumidos à medida que as transições são disparadas. A Figura 2.7 ilustra os elementos de uma PN.



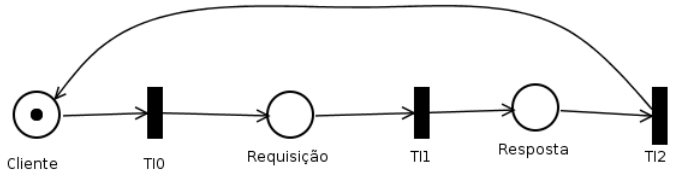
(a) Lugar (b) Transição (c) Arco (d) Token

Figura 2.7: Elementos da rede de Petri: Lugar (a), Transição (b), Arco (c) e Token (d)

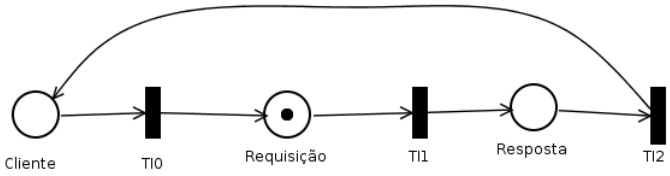
Diversas características fizeram das redes de Petri uma ferramenta interessante para modelagem e análise de sistemas (GIRAULT; VALK, 2013). As PN têm uma representação gráfica de fácil entendimento associada a um forte formalismo matemático. Elas fornecem mecanismos de refinamento e abstração que são de grande importância para o projeto de sistemas complexos. Além disso, possuem uma variedade de ferramentas computacionais disponíveis para modelagem, análise e verificação das redes de Petri.

Diante da sua aplicabilidade, as PN são utilizadas em muitas áreas das ciências aplicadas e engenharias. Portanto, vários resultados são encontrados na literatura para os diferentes domínios da sua aplicação. Hoje, existem várias extensões do modelo básico das redes de Petri que permitem tanto a representação de características básicas, no estudo da concorrência, como também possibilita a análise de problemas práticos das organizações.

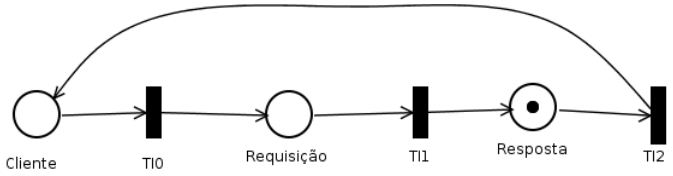
A representação formal de uma rede de Petri é a 5-tupla $PN = \{P, T, F, W, M_0\}$ (MURATA, 1989), onde: $P = \{p_1, p_2, ..., p_m\}$ é o conjunto finito de lugares; $T = \{t_1, t_2, ..., t_n\}$ é o conjunto finito de transições; $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos; $W : F \rightarrow \{1, 2, 3, ...\}$ é a função de atribuição de peso aos arcos . Por fim, $M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ é a marcação inicial.



(a) TI0



(b) TI1



(c) TI2

Figura 2.8: Estados do modelo que representa um sistema cliente-servidor em rede de Petri

Um exemplo de rede de Petri encontra-se na Figura 2.8(a). Este modelo representa um sistema cliente-servidor simples em que o cliente envia uma requisição para o servidor e este processa a requisição e envia a resposta para o cliente. Inicialmente, o modelo está indicando que o cliente pode enviar a requisição ao servidor, visto que existe um *token* no lugar *Cliente* enquanto os demais lugares não possuem *tokens*. As ações, neste caso o envio, o processamento da requisição e envio de resposta, são representadas pelas transições *TI0, TI1, TI2*, respectivamente. Observando o modelo da Figura 2.8(a), somente a transição *TI0* está habilitada, pois existe um *token* no lugar *Cliente* e um arco saindo deste lugar e chegando na transição *TI0*. Uma vez disparada esta transição, o *token* sairá do lugar *Cliente* e chegará no lugar *Requisição*, desabilitando a transição *TI0* e habilitando a transição *TI1* (ver Figura 2.8(b)). Quando *TI1* for disparada, o *token* irá para o lugar *Resposta* e somente a transição *TI2* estará habilitada (Figura 2.8(c)). Por fim, o *token* voltará para o lugar *Cliente* quando disparado *TI2*.

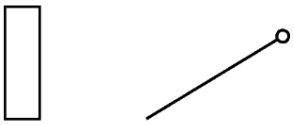
2.4.1 Redes de Petri Estocásticas (SPN)

As redes de Petri estocásticas (MARSAN et al., 1998) são uma extensão das redes de Petri, em que o componente tempo é adicionado nas transições, possibilitando a avaliação de desempenho e dependabilidade (TORRES et al., 2016). Diferente das redes de Petri tradicionais, as SPN consideram o tempo de disparo das transições como variáveis aleatórias que podem ter valores baseados em distribuição exponencial de probabilidade (TRIVEDI, 2008). Além do atraso inserido nas transições, as SPN possuem o conceito de transição imediata, em que o tempo de disparo é zero.

A partir de uma SPN, é possível mapear o modelo para uma CTMC, que pode, por sua vez, ser solucionada, permitindo obter um resultado analítico para as métricas em estudo. As SPN também permitem a utilização de técnicas de simulação para obtenção de métricas de desempenho como tempo de resposta e vazão (ARAÚJO, 2009). Os modelos SPN apresentam uma forte base matemática e são adequados para representar e analisar sistemas paralelos com componentes heterogêneos e que exibem aspectos de simultaneidade e sincronização (MACIEL et al., 2011).

Além dos elementos de uma Rede Petri tradicional, a SPN possui mais 2 elementos diferentes: transição temporizada e o arco inibidor. As Figuras 2.9(a) e 2.9(b) ilustram respectivamente a transição temporizada e o arco inibidor. As regras de conexão entre lugares e transições são as mesmas encontradas numa PN assim como a marcação para representar o estado atual do sistema. Entretanto, o arco inibidor é um tipo de arco especial que possui um pequeno círculo branco em uma das extremidades ao invés de uma seta, e que é utilizado para modelar de forma diferente a ativação das transições. No caso, a transição é desativada se houver *tokens* no lugar de origem do arco inibidor, e será ativada quando não houver *tokens* presentes neste lugar (SILVA, 2016).

A SPN também possui o conceito de prioridade de disparo das transições, em que uma transição imediata tem maior prioridade que as transições temporizadas (SOUSA, 2015). É possível atribuir prioridades de disparo nas transições entre as transições imediata como também atribuir uma função de habilitação. As prioridades são importantes para evitar situações de conflitos (BALBO, 2001).



(a) Transição Temporizada (b) Arco Inibidor

Figura 2.9: Elementos da SPN: Transição Temporizada (a) e Arco Inibidor (b).

A representação formal de um modelo em rede de Petri estocástica é a 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$ (GERMAN, 2000), onde:

- $P = \{p_1, p_2, ..., p_m\}$ é o conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é o conjunto finito de transições temporizadas, onde $P \cup T = \emptyset$;
- $I \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \to \mathbb{N})^{m \times n}$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações);
- $\Pi \in \mathbb{N}^m$ é um vetor que associa o nível de prioridade a cada transição;
- $G \in (\mathbb{N}^n \to \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada a marcação do lugar à cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, W, Markdep, Policy, Concurrency)^m$ compreende o conjunto de atribuições para as transições, onde:
  - $Dist \in \mathbb{N}^m \to \mathcal{F}$ é uma função de distribuição de probabilidade associada ao tempo de uma transição, sendo que o domínio de $\mathcal{F}$ é $[0, 1)$ (a distribuição pode ser dependente de marcação);
  - $W \in \mathbb{N}^m \to \mathbb{R}^+$ é a função peso, que associa um peso (w_t) às transições imediatas e uma taxa λ_t às transições temporizadas, onde:

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

- $Markdep \in \{constant, enabdep\}$ define se a distribuição de probabilidade associada ao tempo de uma transição é constante (*constant*) ou dependente de marcação (*enabdep*);
- $Policy \in \{prd, prs\}$ é a política de preempção adotada pela transição (*prd - preemptive repeat different* corresponde à *resampling* e *prs -preemptive resume* possui significado idêntico à *age memory policy*);
- $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde ss representa a semântica do servidor único e is significa a semântica do servidor infinito (mesmo sentido dos modelos de fila).

Um exemplo de SPN é encontrado na Figura 2.10 que representa um sistema de atendimento de um banco. A transição temporizada *TE0* representa a chegada de clientes no banco, cuja probabilidade de disparo segue uma distribuição cujos parâmetros dependem das chegadas de clientes no banco. Uma das formas de se obter os valores para esta é medindo diretamente os tempos de chegada de cada cliente e utilizando ferramentas como o *Input Data Analyzer* que permitem determinar o tipo de distribuição de probabilidade (exponencial, poisson, uniforme, etc) mais adequado aos dados coletados e seus parâmetros. Ao chegarem, os clientes ficam no lugar *P0*, que modela a fila do banco, e um *token* nesse lugar vai habilitar a transição temporizada *TE1*. Essa transição representa o tempo de atendimento de um cliente e pode ser medida em ambiente real e calculada com o *Input Data Analyzer*, da mesma que forma em *TE0*. Ao ser disparada, o *token* em P0 será removido, representando a saída do cliente do banco.

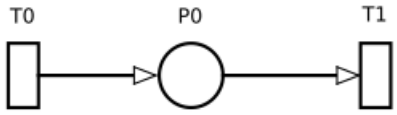


Figura 2.10: Representação de um sistema de atendimento de um banco com SPN.

A partir do modelo é possível calcular métricas de desempenho, entre elas, o tempo de resposta, que representa o tempo decorrido entre o pedido da realização de um serviço pelo usuário e a chegada da resposta do pedido ao usuário requisitante. Essa métrica é representada em unidade de tempo (SOUSA, 2015). Neste exemplo, o tempo de resposta representa o tempo que o cliente chega no banco, permanece na fila e é atendido. O tempo de resposta é calculado através da lei de *Little* (TRIVEDI, 2008), conforme a equação 2.1:

$$T = \frac{N}{\lambda} \tag{2.1}$$

Na equação, T representa o tempo de resposta, N indica o número de usuários no sistema e λ representa a taxa de chegada de usuários no sistema (SOUSA, 2015). Dessa forma, $\mathrm{N} = \mathrm{E}\{\#P0\}$, que é a esperança do número de clientes aguardando na fila e $\lambda = \mathrm{P}\{\#\ P0 > 0\} \times \mathrm{W}(TE1)$ é a taxa de chegada de clientes no banco. De forma mais detalhada, E representa o número médio de *tokens* no lugar indicado entre chaves e P representa a probabilidade de ocorrência do evento expresso entre chaves, que nesse caso indica quando o número de *tokens* (clientes) no lugar *P0* é maior que 0. Já $\mathrm{W}(TE1)$ representa o tempo associado à transição *TE1* (tempo de atendimento de um cliente) (ZIMMERMANN et al., 2006).

2.5 Considerações Finais

Este capítulo apresentou os conceitos fundamentais para o entendimento do trabalho. Inicialmente foram apresentados as características principais e modelos dos Bancos de dados NoSQL. Em seguida, foi apresentado o banco de dados NoSQL Cassandra, que é avaliado neste trabalho. Nesta seção foram apresentados os componentes e estratégias de armazenamento e compactação de dados. As técnicas de avaliação de desempenho foram apresentadas, entre elas, as técnicas de medição, detalhando a ferramenta de *benchmark* YCSB, e modelagem de desempenho. Por último, os conceitos e formalismo das redes de Petri e redes de Petri estocásticas foram apresentados, utilizando exemplos básicos para melhor compreensão.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados na área de avaliação de desempenho de bancos de dados. Os trabalhos estão agrupados em seções baseados nas técnicas utilizadas para avaliar o desempenho. A Seção 3.1 descreve os trabalhos que utilizaram somente técnicas de medição enquanto a Seção 3.2 descreve os trabalhos que utilizaram técnicas de modelagem e modelagem. Por último, este capítulo apresenta uma comparação entre os trabalhos listados e o trabalho aqui proposto.

3.1 Medição

A avaliação de desempenho de bancos de dados é uma área de pesquisa ativa e que tem se intensificado recentemente devido ao aparecimento de modelos de bancos de dados alternativos ao relacional. Muitos trabalhos focaram em técnicas de medição, realizando comparações entre modelos de banco de dados em cenários diferentes. Alguns trabalhos realizaram análises baseados em operações típicas como inserção, atualização e consulta de dados.

Abramova et. al.(ABRAMOVA; BERNARDINO, 2013) realizaram uma análise comparativa de desempenho de dois bancos de dados *NoSQL*, Cassandra e MongoDB. Basicamente, foi analisado o tempo de execução para executar cargas de trabalho diferentes geradas pela

ferramenta YCSB. Os resultados mostraram que o MongoDB apresentou um desempenho inferior à medida que a carga de trabalho aumentava enquanto o Cassandra apresentou um bom desempenho na mesma situação. Apesar de realizar uma análise de desempenho no mesmo banco de dados avaliado no presente trabalho, os autores não abordaram técnicas de modelagem de desempenho.

Em (ABUBAKAR; ADEYI; AUTA, 2014) foi feita uma análise de desempenho em operações de inserção, leitura e atualização comparando quatro banco de dados em um ambiente com recursos limitados. Neste trabalho foi utilizada a ferramenta de *benchmark* YCSB para computar métricas e realizar as operações de inserção, atualização e consulta. Os resultados dos experimentos mostraram que cada banco apresenta comportamento específicos para cada situação. No trabalho, há uma breve descrição do ambiente de experimento como também foram abordadas somente técnicas de medição para avaliação de desempenho dos bancos de dados selecionados.

Os autores em (COOPER et al., 2010) propõem a ferramenta de *benchmark Yahoo! Cloud Serving Benchmark* (YCSB) descrita na Seção 2.3.1. Uma análise de desempenho foi realizada entre quatro banco de dados que estavam na forma distribuída. Foram executadas 4 cargas de trabalhos diferentes, que variavam a distribuição das operações nos banco de dados. Neste caso, os bancos de dados já estavam povoados com 120 milhões de registros cada. As cargas de trabalho variavam entre operações de leitura, atualização dos dados já existentes. Apesar de utilizar a mesma ferramenta de *benchmark*, este trabalho não abordou técnica de modelagem de desempenho dos ambientes analisados.

Li et. al. (LI; MANOHARAN, 2013) também realizaram uma análise comparativa de desempenho, porém o foco era comparar os paradigmas relacional e não relacional. No total foram analisados seis bancos de dados seguindo modelos diferentes do relacional e com diferentes arquiteturas e um banco de dados relacional. Para realizar os experimentos, foi desenvolvido um *framework* específico para aquele ambiente. Neste caso, foram analisados os tempos de instanciamento do banco de dados, realização de conjuntos de operações de leitura, escrita e remoção. Os resultados mostraram que alguns bancos de dados do paradigma não relacional obtiveram melhor desempenho em operações de escrita e remoção do que o banco de dados relacional. Entretanto, em operações de leitura, o banco de dados relacional

apresentou melhor desempenho do que alguns bancos de dados não relacional.

Em (MACIEL et al., 2014) foi realizada uma análise de desempenho e escalabilidade no *Sheepdog*, que é um sistema de armazenamento de dados distribuídos que fornece alta disponibilidade utilizando computadores comuns. Foram analisadas operações de escrita e leitura em quatro cenários variando o número de instâncias e o fator de replicação. Os resultados mostraram melhor desempenho nas operações de leitura do que nas operações de escrita nos cenários com um pequeno número de instâncias do *Sheepdog*, enquanto as operações de escrita apresentaram melhor desempenho em relação a operação de escrita nos *cluster* com maior quantidade de elementos. Diferentemente, essa dissertação realiza a modelagem de desempenho de um banco de dados NoSQL.

Gomes et. al.(GOMES; TAVARES; JUNIOR, 2016) analisaram o desempenho e o consumo de energia de três bancos de dados *NoSQL* diferentes. Para realizar a análise de desempenho, foi utilizada a ferramenta *YCSB* e para analisar o consumo de energia foram utilizados um *hardware* específico junto com um *framework* e *scripts* desenvolvidos no trabalho. A métrica de desempenho analisada foi tempo de execução em operações de leitura, escrita e atualização de dados variados nos níveis 1.000, 10.000 e 100.000 operações. Os resultados mostraram que existe uma variação no consumo de energia significativa entre os bancos de dados analisados para cada operação. Apesar de avaliar o desempenho do mesmo banco de dados analisado no presente trabalho, os autores se restringiram a adotar técnicas de medição para analisar o desempenho.

Os autores em (ASSIS et al., 2017) realizaram uma análise comparativa entre três bancos de dados *NoSQL* e um banco de dados relacional em operações envolvendo mídias digitais. Além disso, foi analisado o comportamento desses bancos de dados em um ambiente de rede social. Os resultados mostraram que os bancos de dados *NoSQL* são mais eficientes que o banco de dados relacional em ambientes com gerenciamento de mídias digitais.

Em (BARROS; GONÇALVES; MEDEIROS, 2015) foi realizada uma análise de desempenho de um banco de dados relacional, representado pelo *MySQL*, e dois bancos de dados NoSQL, representados pelo Cassandra e MongoDB, em operações envolvendo dados genômicos. Foram analisadas operações de inserção, extração e consulta de dados obtidos através do sequenciamento de material genético. Os resultados mostraram que o banco de

dados relacional apresenta limitações no gerenciamento de grande quantidade desses dados. Enquanto que os bancos de dados NoSQL apresentaram melhor desempenho em relação ao *MySQL*, principalmente em operações de inserção e consulta. Vale ressaltar que este trabalho ficou restrito a medição para avaliar o desempenho.

Os autores em (MELO; GOUVEIA; ALENCAR, 2016) realizaram uma análise de desempenho de 3 bancos de dados NoSQL de diferentes modelos no cenário de dados abertos educacionais. Os bancos de dados avaliados foram Cassandra, MongoDB e DynamoDB. Foram avaliadas as operações de inserção, remoção, alteração e consulta. Os resultados mostraram que o MongoDB apresentou melhores resultados em comparação aos demais bancos de dados nas operações analisadas.

Por fim, os autores em (KAUR; SACHDEVA, 2017) descrevem uma nova classe de bancos de dados relacionais conhecida como *NewSQL*. De acordo com os autores, esses bancos de dados fornecem a mesma escalabilidade encontrada nos bancos de dados NoSQL. Uma avaliação de desempenho em quatro bancos de dados *NewSQL* foi realizada, em que o tempo de execução e o tempo de resposta em operações de leitura, escrita e atualização foram analisadas. Além disso, foi realizada uma análise qualitativa entre os bancos de dados escolhidos. Os resultados mostraram que os bancos de dados possuem desempenho diferentes em relação a execução das operações analisadas mesmo pertencendo a mesma classe.

3.2 Modelagem

Diferentemente dos trabalhos que utilizam técnicas de medições, o número de trabalhos que aplicaram técnicas de modelagem e simulação para avaliar o desempenho de banco de dados é menor. A despeito disso, deve-se notar que muito dos trabalhos que empregaram modelagem também empregaram técnicas de medição com o objetivo de validar os modelos propostos. Diversos modelos propostos foram em redes de Petri, comprovando a popularidade desse formalismo para modelagem de sistemas de armazenamento de dados.

Li et. al. (LI; MEDINA; CHAPA, 2007) propôs um modelo utilizando Redes de Petri Colorida Condicionais (CCPN) para modelar e simular os *Active Databases*. Esses bancos de dados são utilizados em ambientes com grande quantidade de informações complexas que

precisam ser processadas em tempo hábil. Mais especificamente, foram modeladas regras específicas desses bancos de dados, denominadas *Event-Condition-Action* (ECA). Os resultados mostraram que a CCPN pode auxiliar na especificação ideal das regras trazendo benefícios no tocante ao desempenho de execução das instruções desse tipo de banco. Este trabalho não teve como foco um banco de dados NoSQL.

Em (NIEMANN, 2015) e (NIEMANN, 2016) foram propostos modelos computacionais utilizando *Queued Petri Nets* (QPN) para modelar o desempenho e o consumo de energia de um sistema de armazenamento de dados, representado pelo Cassandra. Ambos os trabalhos realizaram medições em um ambiente controlado utilizando a ferramenta de *benchmark* YCSB. A métrica de desempenho analisada foi o tempo de execução para executar operações de inserção, leitura e atualização de dados em cenários variando a quantidade de nós no *cluster*. Nesse caso, o banco de dados já estava com dados suficientes para executar a carga de trabalho. De acordo com o autor, o modelo de desempenho proposto em (NIEMANN, 2015) atingiu uma precisão de 80% em relação aos valores encontrados no ambiente de medição e o modelo de consumo de energia atingiu a precisão em torno de 50%. Entretanto, devido a algumas melhoras nos modelos em (NIEMANN, 2016), os modelos de desempenho e de consumo de energia atingiram a precisão de 92% e 84%, respectivamente. Apesar de avaliar o mesmo banco de dados com a mesma ferramenta de *benchmark* utilizada nessa dissertação, os autores realizaram uma modelagem de desempenho de alto nível, desconsiderando os processos internos que o Cassandra realiza durante a inserção de dados, os quais são investigados no trabalho proposto.

Os autores em (GAUR; JOSHI; SRIVASTAVA, 2017) propõem um modelo utilizando Redes de Petri Colorida (CPN) para avaliar o desempenho de um banco de dados enquanto o número de usuários simultâneos está crescendo. Para fins de validação, algumas instruções de consulta e atualização foram executadas utilizando o *Apache JMeter*[1] no banco de dados Postgres[2]. Os resultados mostraram que o modelo auxilia na decisão em escolher o número de servidores necessários caso o número de usuários simultâneos aumenta. O modelo também auxiliou em encontrar o número máximo de requisições que o banco de dados pode suportar.

[1]http://jmeter.apache.org/
[2]https://www.postgresql.org/

3.3 Comparação dos Trabalhos Relacionados

Conforme apresentado anteriormente, muitos trabalhos relacionados focaram em técnicas de medição para avaliar o desempenho de diferentes bancos de dados. Muitos desses trabalhos realizaram comparações entre bancos de dados com modelos e arquiteturas diferentes. A maioria dos trabalhos utilizaram a ferramenta de *benchmark* YCSB para executar conjuntos de operações e computar métricas como tempo de execução e tempo de resposta.

Entretanto, o número de trabalhos que aplicaram técnicas de modelagem e simulação é inferior aos que aplicaram técnicas de medição. Apesar disso, esses trabalhos demonstraram que é possível utilizar extensões das redes de Petri para modelar e avaliar o desempenho de banco de dados. Mesmo utilizando modelos para representar banco de dados, alguns trabalhos também realizaram medições em ambiente controlado a fim de validar os modelos propostos.

A Tabela 3.1 mostra as características dos quatorze trabalhos relacionados mais o trabalho proposto em ordem cronológica. Apenas quatro trabalhos relacionados utilizaram técnicas de modelagem, entre eles, somente um não realizou medição. Os modelos utilizados nesses trabalhos foram extensões das redes de Petri. Por outro lado, treze trabalhos empregaram técnicas de medição para avaliar o desempenho e alguns ainda validaram modelos propostos.

Diferentemente dos trabalhos relacionados, este trabalho propõe um modelo utilizando SPN para avaliar o desempenho do Cassandra durante a operação de inserção de dados. Esta escolha foi feita pelo fato do Cassandra ter sido projetado para gerenciar grande quantidade de dados e estar entre os 10 bancos de dados mais utilizados, conforme o *DBEngines*[3]. Além disso, muitas empresas famosas utilizam o Cassandra, conforme descrito na Seção 2.2. A escolha da SPN é devida a sua popularidade, em que é possível modelar uma variedade de sistemas através dessas redes. A métrica de desempenho analisada foi o tempo de resposta. Esta métrica é importante, pois a comunicação entre usuários e aplicações que utilizam banco de dados deve ser rápida para prevalecer a boa experiência do usuário. A estratégia de modelagem utilizada é focada no comportamento interno do Cassandra, em que foram

[3]https://db-engines.com/en/ranking

Tabela 3.1: Características dos Trabalhos Relacionados.

Trabalho - Ano	**Medição**	**Modelagem**	**Sist. Armazenamento**
Li et al. - 2007	Não	Sim (CCPN)	*Active Database*
Cooper et al. - 2010	Sim	Não	Relacional e NoSQL
Li e Manoharan - 2013	Sim	Não	Relacional e NoSQL
Abramova e Bernardino - 2013	Sim	Não	NoSQL
Abubakar et al. - 2014	Sim	Não	NoSQL
Maciel et al. - 2014	Sim	Não	*Sheepdog*
Niemann - 2015	Sim	Sim (QPN)	NoSQL
Barros et al. - 2015	Sim	Não	Relacional e NoSQL
Gomes et al. - 2016	Sim	Não	NoSQL
Niemann - 2016	Sim	Sim (QPN)	NoSQL
Melo et al. - 2016	Sim	Não	NoSQL
Kaur e Sachdeva - 2017	Sim	Não	NewSQL
Assis et al. - 2017	Sim	Não	Relacional e NoSQL
Gaur et al. - 2017	Sim	Sim (CPN)	Relacional
Este Trabalho - 2018	Sim	Sim (SPN)	NoSQL

observados e computados os tempos de todos os eventos que ocorrem durante inserção.

3.4 Considerações Finais

Este capítulo apresentou os principais trabalhos correlatos ao trabalho proposto. Apesar de existir vários trabalhos que avaliam o desempenho de diferentes bancos de dados, poucos fazem uso de técnicas de modelagem de desempenho. Os trabalhos relacionados foram agrupados em dois grupos baseados nas técnicas utilizadas entre medição e modelagem. O número de trabalhos que fez somente uso de medição é bem maior que aqueles que abordaram técnicas de modelagem e simulação. Embora existam trabalhos que realizaram técnicas de modelagem de desempenho, nenhum deles utilizou as Redes de Petri Estocástica para avaliação de desempenho de banco de dados.

Capítulo 4

Análise do *Cluster* do Cassandra

Este capítulo apresenta um conjunto de experimentos realizados com o propósito de analisar o desempenho do banco de dados Cassandra no processo de inserção de dados variando a quantidade de nós no *cluster* e a quantidade de operações de inserção. Além disso, também foi analisado o consumo de energia do banco de dados. O principal objetivo desta análise é entender como o desempenho durante o processo de inserção de dados do Cassandra e seu respectivo consumo de energia são impactados pelo aumento da escala do banco.

A primeira seção do capítulo apresenta o método utilizado para medição do desempenho e do consumo de energia. A Seção 4.2 apresenta a análise de desempenho. Enquanto a Seção 4.3 apresenta a análise do consumo de energia dos cenários avaliados. Por último, a Seção 4.4 realiza uma discussão dos resultados obtidos.

4.1 Método de Medição

Para realizar a análise de desempenho e consumo de energia do Cassandra em operações de inserção, são analisados dois fatores: a quantidade de nós no *cluster* e a quantidade de operações de inserção. Esses fatores são variados em níveis. A combinação dos níveis dos fatores indicam a quantidade de cenários que devem ser analisados.

O fator quantidade de nós no *cluster* foi variado em 4 níveis: 1, 2, 4 e 6 nós. Já o fator quantidade de operações apresenta 3 níveis: 10.000 (10K), 100.000 (100K) e 1.000.000 (1M) operações de inserção. Para executar as operações de inserção e computar as métricas de desempenho foi utilizada a ferramenta de *benchmark* YCSB. Os registros gerados pelo YCSB eram do tipo texto e inseridos em uma tabela com 10 colunas. Cada registro tinha 1KB de informação. Com isso, foram inseridos 10MB, 100MB e 1GB de dados para os cenários 10K, 100K e 1M, respectivamente. Para executar as operações de inserção, foram utilizados 10 usuários simultâneos (*threads).*

Para obter métricas de consumo de energia, foi adicionado um dispositivo externo de medição, o *Wattsup Meter*[1]. Este dispositivo registra métricas de consumo de energia, inserindo os valores em arquivos texto. Esses valores são capturados em intervalo de tempo ajustável. Quanto menor o intervalo de tempo, mais preciso são os valores do consumo de energia. O *Wattsup Meter* é conectado entre o estabilizador e a rede elétrica, também servindo como condutor de energia ao sistema. A Figura 4.1 ilustra a distribuição dos componentes no ambiente de medição.

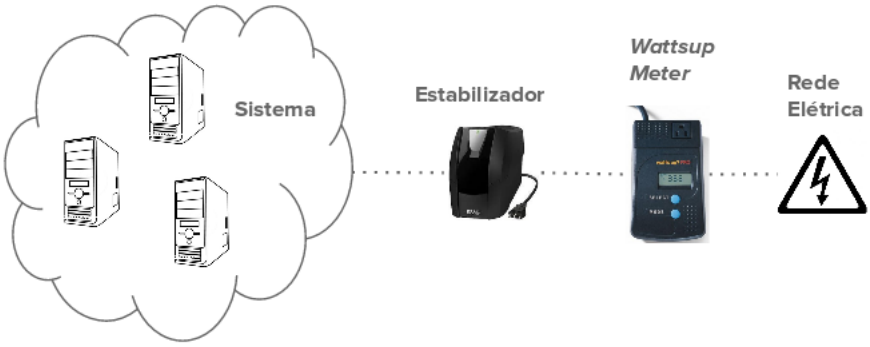


Figura 4.1: Visão geral dos componentes do ambiente de medição.

4.1.1 Ambiente de Medição

O ambiente montado em laboratório foi composto por quatro computadores com processador Intel core i5, 8GB de memória RAM, 500GB de espaço em disco e com sistema operacional Ubuntu 14.04 LTS. Todos conectados através de uma rede local *Gigabit Ethernet*. A ferramenta YCSB foi executada de forma dedicada em um dos computadores. Nos três

[1]https://wattsupmeters.wordpress.com/about/

computadores restantes foram hospedados até dois nós (2 máquinas virtuais - VM) de acordo com o cenário. Os nós do *cluster* Cassandra foram executados em VMs, todas idênticas com 2GB de memória RAM, 30GB de espaço em disco e sistema operacional Ubuntu 14.04LTS. Logo, para os experimentos com 2, 4 e 6 nós foram utilizadas 1, 2 e 3 computadores, respectivamente. Apenas os computadores necessários para cada *cluster* eram ligados, por exemplo, para o *cluster* com 2 nós, somente 1 computador (com as 2 VMs hospedadas) foi utilizado, os demais permaneciam desligados.

Foram utilizados 2 estabilizadores, um para alimentar o computador que executava a aplicação cliente (YCSB) junto com os demais componentes: monitor e *switch*. Já os demais computadores estavam ligados ao segundo estabilizador que, por sua vez, se conectava ao *Wattsup Meter* o qual estava ligado à rede elétrica. Desta forma, o *cluster* Cassandra permaneceu eletricamente isolado do outros componentes do ambiente de teste, de modo que a medição de consumo de energia não foi impactada pelos demais componentes. A Figura 4.2 ilustra o ambiente de medição.

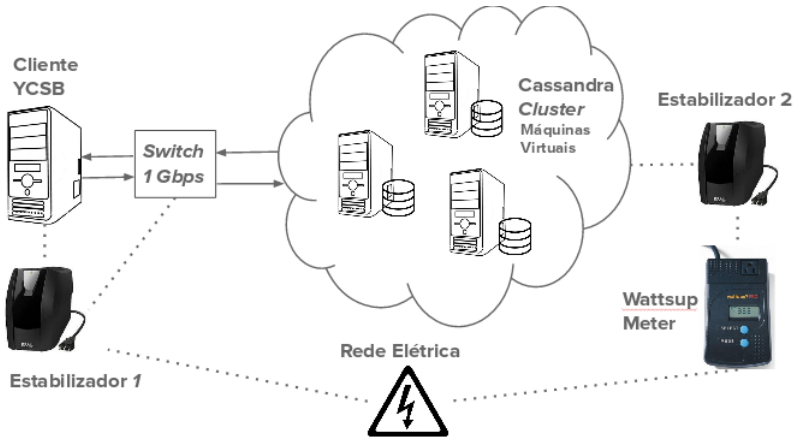


Figura 4.2: Visão geral dos componentes do ambiente de medição.

4.1.2 Processo de Obtenção das Amostras

A Figura 4.3 ilustra a sequência de passos do processo de obtenção das amostras. Inicialmente, a primeira atividade executada foi a preparação dos nós no *cluster* do Cassandra, onde todos os *logs*, *Commit logs* e *caches* gerados pelo *cluster* foram apagados. Em seguida, realizou-se a remoção dos registros gerados pelo *Wattsup Meter*, que armazena, sem

interrupções, as informações do consumo de energia a cada 1 segundo, por exemplo.

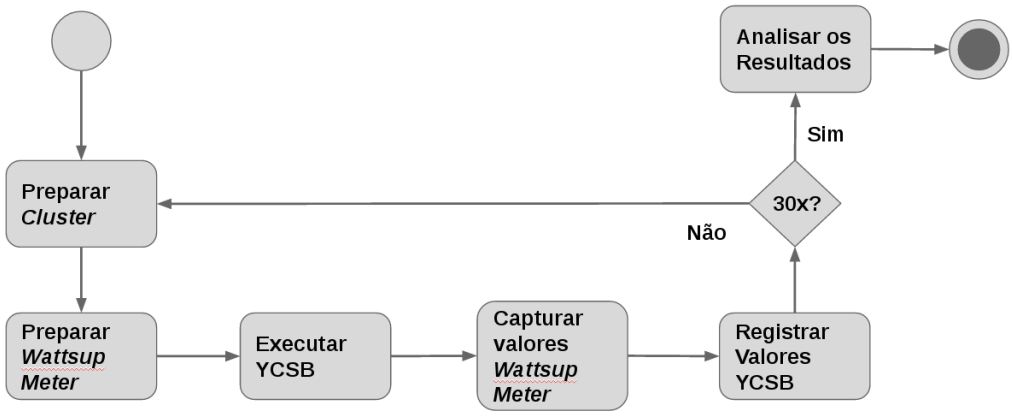


Figura 4.3: Processo de obtenção das métricas de desempenho e consumo de energia.

Após os procedimentos de preparo dos componentes, a carga de trabalho do YCSB é executada na fase *load*, ou seja, inicia-se a inserção de dados no *cluster* Cassandra. Ao término da inserção de todos os dados, os registros gerados no *Wattsup Meter* são armazenados e para cada amostra é gerada uma tabela com um arquivo texto. A partir dessas informações, pode-se computar o consumo de energia da amostra. Em seguida, registra-se as métricas obtidas no YCSB. Por último, realiza-se a análise de resultado fornecendo suporte para as conclusões dos experimentos. Este procedimento foi repetido por pelo menos 30 vezes.

As métricas de desempenho analisadas foram: tempo de execução para inserir todos os registros (segundos), a vazão do sistema (operações por segundo) e o tempo de resposta (milissegundos), que são fornecidas pelo YCSB. O consumo de energia (em *Joules*) foi obtido ao final do tempo de execução de cada experimento. O experimento foi conduzido para cada combinação dos níveis com os fatores (12 combinações), sendo coletadas 33 amostras em cada uma. As 3 primeiras amostras descartadas em todos os experimentos, pois foi observado através do tempo de execução que elas atingiam valores bem acima das demais, sendo consideradas como período de *warm-up* do Cassandra. O intervalo de confiança para média, considerando 95% de confiança, foi calculado e, na maioria dos casos, os intervalos não se sobrepõem, mostrando que os valores das médias são estatisticamente diferentes. Os intervalos de confiança não serão mostrados nos gráficos, mas nos casos em que não foi possível definir a diferença estatística serão pontuados no texto.

4.2 Análise de Desempenho

Foram calculados valores estatísticos como a média, desvio padrão e intervalo de confiança para cada cenário. Além disso, em algumas situações foram realizados o Teste-t com duas amostras independentes com o objetivo de verificar se as amostras são estatisticamente equivalentes. O apêndice B descreve o Teste-t com duas amostras independentes. Cada métrica será abordada separadamente.

Tempo de Execução. O tempo de execução consiste no tempo total para realizar um conjunto de operações de inserção. O gráfico da Figura 4.4 mostra o tempo de execução de acordo com a quantidade de registros inseridos. Para 10K operações de inserção, é possível observar que o tempo permaneceu constante mesmo com o aumento da quantidade de nós. Já para 100K, é possível observar que o tempo de execução para 1 e 2 nós são estatisticamente iguais. O teste-t emparelhado, cujo resultado é mostrado na Tabela 4.1, mostra que não podemos rejeitar a hipótese nula, que indica a igualdade dos tempos de execução com 1 e 2 nós. Contudo, a partir de 4 nós, é possível observar uma queda no tempo de execução. A Tabela 4.1 não apresenta os resultados para 10K, pois este cenário apresentou comportamento uniforme mesmo com o aumento da quantidade de nós.

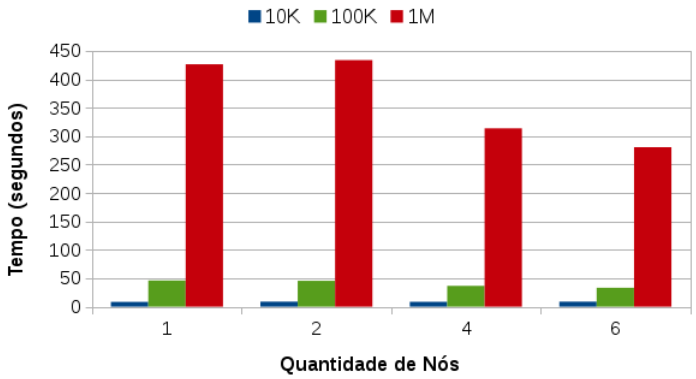


Figura 4.4: Tempos de execução de acordo com o número de registros inseridos.

Tabela 4.1: Teste-T no tempo de execução entre *clusters* com 1 e 2 nós.

Qntd Operações	Valor-p	α	Diferença entre as médias
100K	0,1397	0,05	0,54 seg
1M	6,147E-008	0,05	7,35 segs

Já para 1M de operações, o tempo de execução para 2 nós é um pouco maior do que para 1 nó. Também foi realizado teste-t (Tabela 4.1) nesta situação, que indicou que os valores são diferentes, visto que o valor-p é menor que o α. Este comportamento pode ser justificado pelo compartilhamento de recursos entre as VMs, que por estarem hospedadas no mesmo computador, compartilham recursos como a placa de rede e disco. Desta forma, o aumento da carga de trabalho leva a um tráfego mais intenso entre as VMs e um número maior de escritas em disco, levando a uma maior disputa por estes recursos e o consequente aumento do tempo de resposta.

Vazão. A Figura 4.5 mostra o comportamento da vazão em relação à quantidade de operações de inserção no *cluster*. É possível observar que para 10K operações, a vazão apresentou comportamento uniforme, no entanto para as demais situações (100K e 1M) observou-se crescimento considerável a partir de 4 nós.

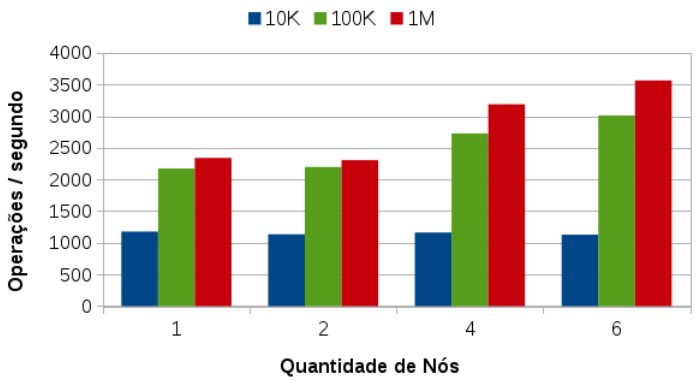


Figura 4.5: Comportamento da Vazão.

Em todos os cenários, a vazão permaneceu similar para 1 e 2 nós. Para as inserções de 100K e 1M, houve um aumento significativo, principalmente para 1 milhão de registros, este último apresentou um crescimento de mais de 50% em relação ao experimento com um único nó, como mostra a Figura 4.6, que apresenta a proporção da vazão para 2, 4 e 6 nós em relação a 1 nó. A partir das Figuras 4.5 e 4.6, pode-se concluir que o aumento do *cluster* Cassandra de 1 para 2 nós não apresentou aumento significativo na vazão em nenhuma das situações analisadas, a despeito do aumento de recursos computacionais.

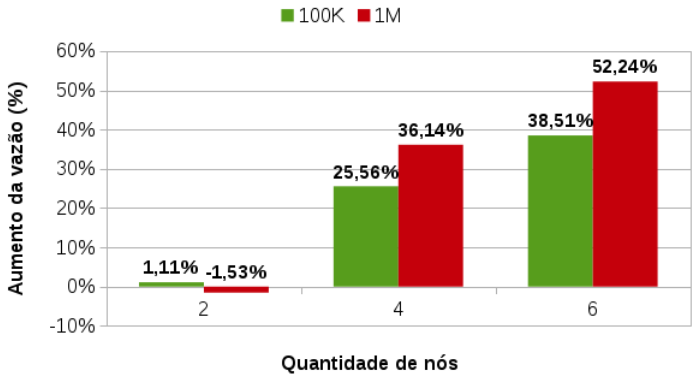


Figura 4.6: Comparação da vazão em relação a 1 nó para 100K e 1M de operações.

Tempo de Resposta. O gráfico da Figura 4.7 ilustra o comportamento do tempo de resposta. É possível observar que para 10K operações de inserção, o tempo de resposta apresentou um comportamento uniforme, em que os tempos tinham diferença média de 2%. No entanto, em um *cluster* com 2 nós o tempo de resposta teve um aumento de 5% em relação a 1 nó. Para 100K e 1M operações de inserção, observa-se uma queda a partir de 4 nós. Esta queda representou uma diminuição de aproximadamente 26,8% quando o *cluster* era composto com 4 nós e 35% para 6 nós, em relação a 1 nó. É possível observar que para as duas maiores quantidades de operações de inserção (100K e 1M), os valores médios quantificados ficaram sobrepostos, indicando que, para os cenários analisados, um *cluster* Cassandra com 4 nós oferece a melhor razão entre tempo de resposta e uso de recursos.

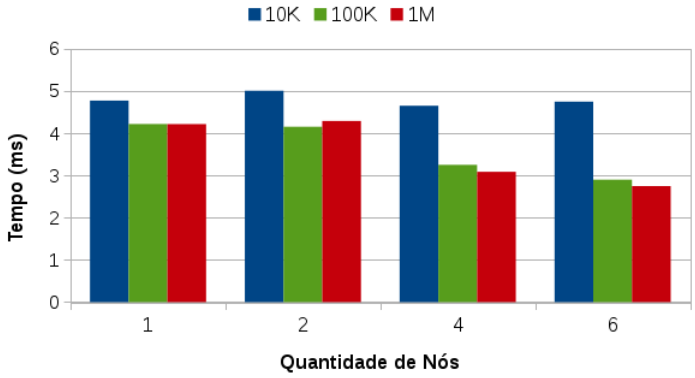
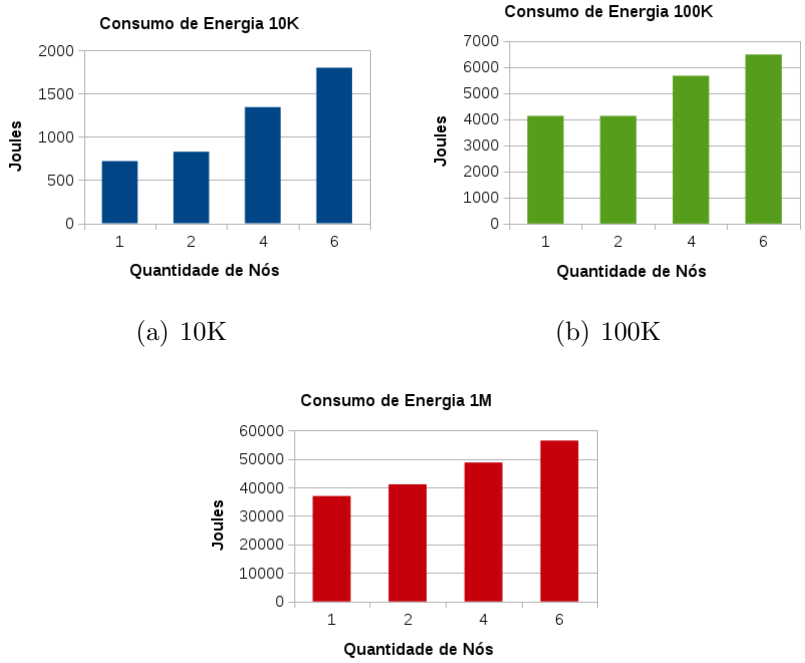


Figura 4.7: Análise Tempo de Resposta.

4.3 Análise do Consumo de Energia

A Figura 4.8 ilustra o consumo de energia de acordo com o aumento do número de nós no *cluster* durante o experimento com 10K, 100K e 1M operações de inserção. É possível observar que para 10K (Figura 4.8(a)), houve um aumento significativo do consumo de energia para 4 e 6 nós. Para 100K (Figura 4.8(b)), o consumo de energia para 1 e 2 nós foi similar. Mesmo utilizando 2 nós hospedados em um único computador, o consumo de energia não sofreu alterações. Contudo para 4 e 6 nós houve um aumento significativo, visto que a quantidade de computadores aumentou para 2 e 3 respectivamente.



(a) 10K

(b) 100K

(c) 1M

Figura 4.8: Consumo de Energia para 10K (a), 100K (b) e 1M (c)

A Figura 4.9 mostra o aumento do consumo de energia em relação a 1 nó. Observa-se um aumento significativo de 87% e 150% no consumo de energia para realizar 10K operações de inserção em 4 e 6 nós, respectivamente, devido a adição de mais computadores no ambiente (2 computadores para 4 nós e 3 computadores para 6 nós). Já para 100K e 1M, os aumentos foram próximos. Para 4 nós o aumento foi 37% e 32% para 100K e 1M, respectivamente. Enquanto que para 6 nós o aumento foi de 57% e 53% nas mesmas condições.

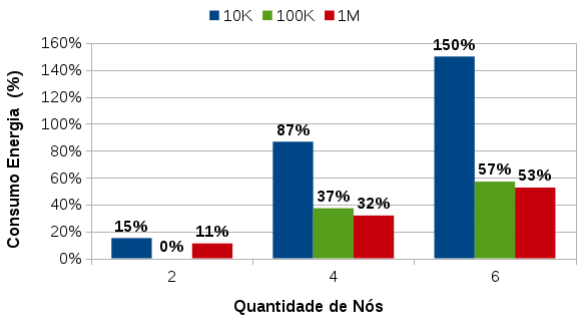


Figura 4.9: Aumento do consumo de energia em relação a 1 nó.

4.4 Discussão

Analisando de forma conjunta o desempenho e o consumo de energia dos cenários propostos é possível observar que o aumento de 1 para 2 nós no *cluster* não apresentou melhorias significativas para o desempenho. Um dos fatores para este comportamento é o compartilhamento de recursos, como placa de rede e disco, já que as 2 máquinas virtuais estavam hospedadas em um único computador. Observa-se que a introdução de novos nós virtuais sob um mesmo computador físico, pode não apresentar bons resultados. Por outro lado, quando utilizavam-se 2 nós, o aumento do consumo de energia em relação a 1 nó foi de 15% para realizar 10K operações de inserção e 11% para inserir 1M de registros.

Quando o *cluster* possui 4 ou 6 nós, constata-se significativa melhoria no desempenho em relação a 1 nó, principalmente para realizar inserção dos dois maiores conjuntos de dados, 100K e 1M. Entre as métricas de desempenho, pode-se destacar a vazão, que obteve um aumento de mais de 50% quando inseridos 1M de registros em um *cluster* composto por 6 nós. Entretanto, houve um aumento de 150% no consumo de energia quando inseridos 10K registros. Além disso, ocorreu um aumento, em relação a 1 nó, de 57% e 53% no consumo de energia para inserir, respectivamente, 100K e 1M de registros. A Tabela 4.2 mostra uma análise na eficiência do desempenho e do consumo de energia para 100K e 1M em um *cluster* composto por 4 e 6 nós em relação a 1 nó. A tabela não apresenta resultados para 10K operações pois este caso não apresenta alterações significativas nas métricas de desempenho, entretanto na métrica de consumo de energia esse caso é considerado. Os valores negativos indicam uma redução em relação ao valor encontrado em 1 nó.

Tabela 4.2: Análise das métricas para 4 e 6 nós em relação a 1 nó.

Métricas	**Qtd Operações**	**4 nós**	**6 nós**
Tempo de Execução	100K	-20,42%	-27,88%
	1M	-26,42%	-34,20%
Vazão	100K	25,56%	38,51%
	1M	36,14%	52,24%
Tempo de Resposta	100K	-22,95%	-31,24%
	1M	-26,80%	-34,81%
Consumo de Energia	10K	86,67%	150,00%
	100K	37,21%	56,98%
	1M	31,77%	52,56%

Baseados nos valores da Tabela 4.2, é possível indicar a quantidade ideal de nós de acordo com o número de operações de inserção realizadas. Para manipulação de 10K registros pode-se utilizar a instalação do Cassandra em apenas um nó, já que o aumento do número de nós no *cluster* não melhora o desempenho neste caso simples e aumenta o consumo de energia.

Já para 100K e 1M, o número de nós ideal foi 4, pois, nesta configuração, há um aumento considerável no desempenho com um menor consumo de energia do que um *cluster* com 6 nós. Note-se que obteve-se o melhor desempenho com 6 nós, contudo, nesta configuração, houve um aumento significativo do consumo de energia. Em todos os casos o aumento foi superior a 50%. Então, se o consumo energético não for uma restrição, um *cluster* com 6 nós proverá o melhor desempenho. No entanto, considerando consumo de energia e desempenho de forma integrada, a configuração com 4 nós provê um balanço adequado entre os fatores analisados.

4.5 Considerações Finais

Este capítulo apresentou um conjunto de experimentos que avaliou o desempenho e o consumo de energia do *cluster* do Cassandra para operações de inserção. Foram analisadas diferentes configurações do *cluster* com diferentes quantidades de operações de inserção. Analisando as métricas de desempenho, pode-se observar um comportamento linear quando aumenta-se o número de máquinas físicas (cenários com 4 e 6 nós). Contudo, para os cenários de 1 para 2 nós, é observado um comportamento uniforme nas métricas de desempenho. Isto pode ter ocorrido devido ao fato de utilizar somente uma máquina física.

O comportamento linear das métricas de desempenho sugere que o Cassandra apresenta uma escalabilidade linear. Embora o número de nós utilizados neste trabalho seja limitado devido ao número de máquinas físicas, o comportamento linear também é observado em avaliações de desempenho de outros trabalhos (COCKCROFT; SHEAHAN, 2011; RABL et al., 2012; POINT, 2015). Dessa forma, a próxima etapa deste trabalho é focar na análise interna do comportamento de um nó do Cassandra durante o processo de inserção, visto que todo os nós do Cassandra realizam as mesmas tarefas.

Portanto, o entendimento do *cluster* do Cassandra serviu como base para desenvolver o modelo de desempenho como também foi importante para entender as ferramentas para obtenção das métricas de desempenho. Além disso, esta primeira etapa do trabalho foi publicada no *Workshop de Computação em Clouds e Aplicações - XV* (WCGA) - 2017 (BARROS; CALLOU; GONÇALVES, 2017).

Capítulo 5

Modelagem de Desempenho do Cassandra

Este capítulo apresenta a modelagem de desempenho do Cassandra. A Seção 5.1 apresenta o método de modelagem de desempenho utilizado neste trabalho, detalhando o processo de obtenção de informações necessárias para a criação do modelo. Para o desenvolvimento do modelo, foi utilizado o formalismo de redes de Petri estocásticas (SPN). Este modelo permite realizar a análise de desempenho do processo de inserção de dados no Cassandra. Finalmente, os elementos e os parâmetros do modelo são detalhados (Seção 5.2).

5.1 Método de Modelagem

A modelagem de desempenho de um banco de dados consiste em desenvolver um modelo formal que represente determinado ambiente em que o banco de dados está submetido. O modelo é desenvolvido considerando os principais elementos do sistema que impactam no desempenho, ou seja, são representados a aplicação cliente (geradora da carga de trabalho), componentes de rede e os processos internos do banco de dados. Também verificam-se, através do modelo, possíveis situações de *deadlock* como também gargalos que podem interferir durante o processo de simulação ou análise estacionária.

Inicialmente, o comportamento do Cassandra foi analisado através de técnicas de medição. Vários bancos de dados informam com precisão o que está acontecendo durante a execução de determinado processo interno através dos *logs*. O Cassandra, por exemplo, indica através dos *logs* o momento exato do processo de *flush*, além de informar a quantidade de dados que é descarregada da memória para o disco rígido como também o tempo de execução das compactações e a quantidade de *bytes* compactados. Com o auxílio de *scripts* é possível extrair os momentos exatos e a duração desses processos. Esses valores são utilizados para realizar cálculos estatísticos, correlacionando com o tempo de resposta no momento desses processos. Dessa forma, é possível determinar se tal processo impacta no tempo de resposta das requisições.

Neste trabalho foram realizadas inserções de dados por longos períodos com o objetivo de obter informações suficiente dos *logs* fornecidos pelo YCSB e pelo Cassandra. Para fins de validação do modelo de desempenho, foi criado um ambiente de medição com o objetivo de medir os tempos dos processos internos do Cassandra assim como as métricas de desempenho. A Figura 5.1 apresenta os componentes básicos do ambiente de medição para análise de desempenho de banco de dados realizada neste trabalho. Ao final da inserção, esses *logs* eram capturados. A cada 10 segundos foram coletadas as amostras e o tempo de resposta computado pelo YCSB. O mínimo de 30 amostras foram coletadas para se fornecer a confiabilidade desejada nos dados quantificados durante a medição.

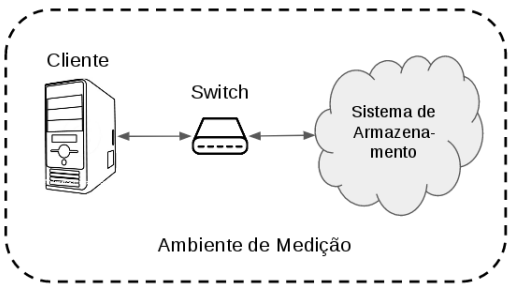


Figura 5.1: Visão Geral dos componentes do ambiente de medição.

5.1.1 Análise do Tempo de Resposta no Ambiente de Medição

No Cassandra, foram capturados os momentos e durações dos processos de *flush* e compactações. A partir desses valores, realizou-se uma análise categorizada do tempo de res-

posta, ou seja, analisou-se o tempo de resposta das operações de escrita em categorias que variam de acordo com os processos internos que estavam ocorrendo no Cassandra. Para tanto, a duração e o tempo em que ocorreram estes processos internos foram utilizados para detectar quais operações de escrita ocorreram concomitantemente a estes processos. Desta forma, pretende-se quantificar o impacto dos processos internos do Cassandra no tempo de resposta das operações de escrita. A Tabela 5.1 ilustra as categorias em que as operações de escrita foram classificadas. Para cada amostra é atribuída apenas uma das possíveis categorias. Por exemplo, na categoria #1 "Nenhum Processo", significa que durante operações de inserção, o banco de dados do Cassandra não estava realizando nem *flush* e nem Compactação. Essa categorização é essencial, uma vez que operações de *flush* e compactação impactam no tempo de inserção dos dados no banco.

Tabela 5.1: Categorias para análise de impacto dos processos internos do Cassandra no tempo de resposta das operações de escrita.

#	**Processo Cassandra**	**Flush**	**Compactação**
1	Nenhum Processo	Não	Não
2	Apenas Flush	Sim	Não
3	Apenas Compactação	Não	Sim
4	Flush e Compactação	Sim	Sim

Para obtenção de resultados do ambiente de medição foram realizadas um conjunto de atividades sequenciadas, conforme ilustra a Figura 5.2. A primeira atividade consiste em preparar o Cassandra, removendo os logs e *cache*, em seguida executa o YCSB configurado para computar as métricas a cada 10 segundos. Esse intervalo de tempo representa o tempo de resposta médio de todas as inserções que ocorreram no período de 10 segundos. Ao final da medição, foram obtidos os *logs* do YCSB e do Cassandra, os quais foram submetidos a análise categorizada do tempo de respostas baseados nas possíveis combinações (ver Tabela 5.1). Em seguida, os resultados foram analisados com o objetivo de identificar possíveis impactos dos processos do Cassandra no tempo de resposta da inserção dos dados no banco. Após obtenção dos resultados da medição, realiza-se uma análise dos resultados obtidos e que serão utilizados como parâmetros de entrada do modelo de desempenho (ver Seção 5.2).

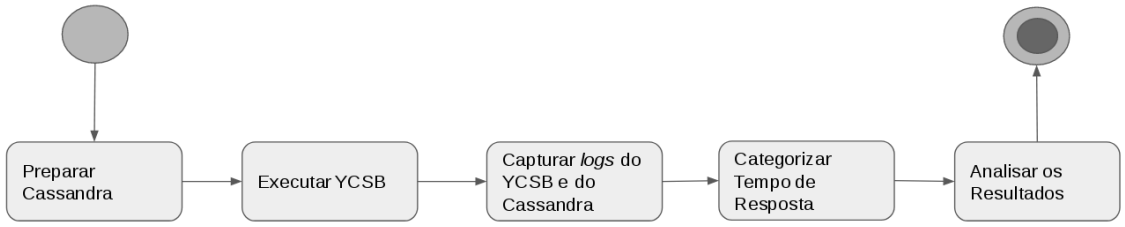


Figura 5.2: Atividades realizadas para obtenção dos resultados no ambiente de medição.

5.2 Modelo de Desempenho

A Figura 5.3 ilustra o modelo de desempenho proposto por esse trabalho. O modelo de desempenho permite a análise do tempo de resposta de operações de escrita no banco de dados Cassandra em diferentes configurações. O modelo segue o processo descrito na Seção 2.2.2. Quando uma operação de escrita de um registro é submetida ao Cassandra, o registro é inserido primeiramente na *Memtable* e a instrução é escrita no *Commit log*. Logo em seguida, uma resposta já é enviada ao Cliente indicando que a operação ocorreu com sucesso. Quando a *Memtable* atinge um tamanho limite preconfigurado, ocorre o processo de *flush*. Durante o processo de *flush*, novas requisições de escrita submetidas ao banco ficam em espera. Periodicamente ocorre a compactação, porém este processo não bloqueia a chegada de novas requisições.

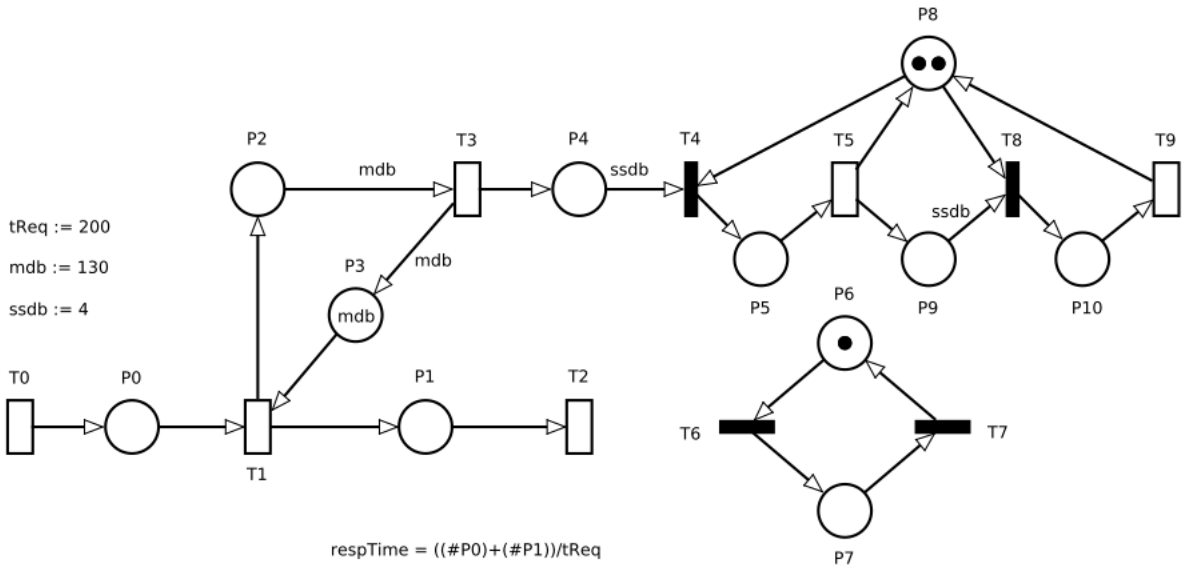


Figura 5.3: Modelo de Desempenho Proposto.

A transição *T0* representa a chegada de requisições de inserção no Cassandra. Ao ser disparada, é gerado um *token* em *P0*. O lugar *P0* representa a fila de operações de escrita que serão atendidas pelo Cassandra.

A transição temporizada *T1* representa o tempo de escrita do registro na *Memtable*. Quando *T1* é disparada, são gerados dois tokens: um em *P1* e outro em *P2*. O lugar *P1* representa a fila de envio de respostas para aplicação cliente. Enquanto *P2* representa os registros que foram escritos na *Memtable*.

A transição *T2* refere-se ao tempo de envio da resposta para o cliente. Quando disparada, um *token* é removido do lugar *P1*. Este fluxo representa o caminho completo do fluxo, desde da chegada da requisição no Cassandra até o envio da resposta.

O lugar *P3* é utilizado para representar a quantidade máxima de registros que a *Memtable* pode armazenar antes de executar um *flush*. O número máximo de registros suportados na *Memtable* é um parâmetro configurável no Cassandra que é definido no modelo pelo parâmetro *mdb*. Quando *T1* é disparada *mdb* vezes, o lugar *P3* fica sem nenhum *token*, impossibilitando o disparo de *T1* e representando o bloqueio de escrita de novos registros no Cassandra até a finalização do *flush*.

No momento em que o lugar *P3* fica sem *tokens*, o lugar *P2* tem quantidade suficiente de *tokens* para habilitar a transição temporizada *T3*, conforme o peso do arco que sai de *P2* para *T3* que é igual ao parâmetro *mdb*. Esta transição representa a duração do processo de *flush* do Cassandra. Quando *T3* for disparada, será gerado um *token* em *P4* e *mdb tokens* em P3, habilitando T1 novamente para disparar *mdb* vezes.

O *token* em *P4* representa que uma *SSTable* foi gerada a partir do *flush*. Quando este lugar atingir a quantidade de *SSTable* definidas no parâmetro *ssdb*, a transição imediata *T4* será habilitada. O parâmetro *ssdb* indica o número mínimo de *SSTables* necessárias para iniciar o processo de compactação.

Após o disparo de *T4*, que é imediato, o lugar *P5* estará com um *token* e a transição temporizada *T5* estará habilitada. Esta transição representa a compactação das *SSTable* geradas a partir do *flush*. Uma vez disparada, será gerado um *token* em *P9* que representa uma *SSTable* compactada. Quando atingir o valor definido por *ssdb*, a transição imediata

T8 ficará habilitada para disparo. Após o disparo de *T8*, o lugar *P10* receberá um *token* e o disparo na transição temporizada *T9* estará habilitado. Esta transição representa uma nova compactação das *SSTable* já compactadas.

Deve-se destacar no modelo o lugar *P8* que possui 2 *tokens* iniciais. Este lugar indica a quantidade máxima de compactações simultâneas que o Cassandra pode fazer. Em análise no ambiente de medição, foi constatado que o banco de dados realiza até duas compactações simultâneas.

Por último, o modelo possui um *clock* representado pelos lugares *P6* e *P7* e pelas transições imediatas *T6* e *T7*. Este *clock* indica o momento que o Cassandra está realizando pelo menos uma compactação. O *clock* auxilia a computação do tempo de resposta simplificando a parametrização do modelo. A Tabela 5.2 mostra as funções de habilitação inseridas nas transições imediatas do *clock*. A transição *T6* dispara quando há pelo menos um *token* em *P5* ou em *P10*. Um *token* no lugar *P7* indica que as operações de inserção de dados no Cassandra ocorrem durante a compactação. Para o *token* retornar a *P6*, a condição em *T7* indica que quando não houver *token* em *P5* e nem em *P10*. Dessa forma, quando há um *token* em *P6* o *clock* indica que naquele momento o banco não está realizando nenhuma compactação.

Tabela 5.2: Funções de Habilitação do *Clock*.

#	**Transição**	**Função de Habilitação**
1	T6	$(\#P5 > 0)OR(\#P10 > 0)$
2	T7	$(\#P5 = 0)AND(\#P10 = 0)$

5.2.1 Parâmetros e Métrica do Modelo

O modelo possui três parâmetros que podem ser definidos pelo usuário. Em ambiente real, eles representam configurações do ambiente ao qual o Cassandra está submetido. Eles são importantes para realizar a análise do modelo de desempenho em situações não observadas no ambiente de medição através de combinações entre eles. O modelo também possui uma métrica responsável pelo cálculo do tempo de resposta.

O parâmetro *tReq* representa a taxa de chegada das requisições. No ambiente de medição,

esta taxa é definida pelo YCSB através dos parâmetros *target* e *threads* durante a execução da carga de trabalho. Este parâmetro é inserido no tempo da transição determinística *T0*. O valor inserido em *T0* é o inverso de *tReq*, que representa o tempo médio entre chegadas de cada requisição.

Para representar a quantidade máxima de registros que a *Memtable* armazena, foi criado o parâmetro *mdb*. Este valor foi calculado a partir de pequenas amostras no ambiente de medição. Para fins de validação, foi necessário realizar ajuste no tamanho da *Memtable*, pois a quantidade padrão de registros que a *Memtable* armazena pode ser muito grande, o que demandaria uma grande quantidade de *tokens* no modelo, para representar essa situação. Isto poderia implicar em problemas na simulação e na análise estacionária devido ao problema de *Largeness*(KORIEM; DABBOUS; EL-KILANI, 2004). Este problema consiste em que as SPN possuem um limite de alcançabilidade para gerar a Cadeia de Markov. Em outras palavras, existe um número máximo de *tokens* que podem ser gerados por uma SPN.

Já o parâmetro *ssdb* representa a quantidade mínima de *SSTable* geradas antes de iniciar o processo de compactação. A estratégia de compactação representada neste modelo é a *SizeTieredCompactionStrategy*, que é a estratégia padrão. Ela indica o início da compactação quando um número fixo de *SSTables* de tamanhos próximos estejam disponíveis para compactação. Por padrão, são necessárias quatro *SSTables* para iniciar a compactação. Contudo, este valor pode ser alterado durante a criação da tabela no Cassandra.

Por último, a métrica *respTime* do modelo é responsável por realizar o cálculo do tempo de resposta. Este tempo representa a duração entre o envio da requisição ao sistema até a chegada da resposta para aplicação cliente. Conforme visto na Seção 2.4.1, o método para calcular o tempo de resposta segue os conceitos da Lei de *Little*. A fórmula do tempo de resposta no modelo está na equação 5.1:

$$respTime = \frac{(E\{\#P0\}) + (E\{\#P1\})}{tReq} \tag{5.1}$$

Neste caso, $(E\{\#P0\})$ e $(E\{\#P1\})$ representam o número médio de *tokens* nos lugares *P0* e *P1*, respectivamente. Como a taxa de chegada das requisições é conhecida, o λ da fórmula encontrada na fórmula de *Little* (2.1) é a taxa de requisição do modelo *tReq*.

5.3 Considerações Finais

O presente capítulo apresentou a modelagem de desempenho do Cassandra para operações de inserção de dados. Foi abordado o método de modelagem do banco de dados, em que foi entendido o processo de desenvolvimento do modelo de desempenho e a estratégia utilizada para obter o tempo de resposta em diferentes situações. Em seguida, foi apresentado o modelo baseado em redes de Petri estocásticas que representa o processo de inserção do Cassandra, proposto por este trabalho. Foram detalhados os principais componentes do modelo e a sua relação com o sistema real, como as transições temporizadas, transições imediatas e os lugares. Também foram apresentados os parâmetros do modelo e o cálculo do tempo de resposta.

Capítulo 6

Estudos de Caso

Este capítulo apresenta quatro estudos de caso. O principal objetivo é avaliar se o modelo proposto neste trabalho pode ser aplicado para realizar a análise de desempenho do Cassandra durante o processo de inserção de dados. O primeiro estudo de caso (Seção 6.1) apresenta a validação do modelo de desempenho. Neste estudo foram utilizadas técnicas de medição para obtenção de resultados que foram comparados aos encontrados no modelo de desempenho. Já os demais estudos de caso (Seções 6.2, 6.3 e 6.4) utilizaram técnicas de modelagem e simulação para realizar a análise de desempenho do Cassandra em diferentes configurações. Estes estudos tem o objetivo de analisar a aplicabilidade do modelo proposto neste trabalho e podem auxiliar na escolha da melhor forma de utilizar o banco de dados quando o foco é desempenho em operações de inserção.

6.1 Estudo de Caso I - Validação do Modelo

Conforme visto na Seção 2.3.2, o desenvolvimento do modelo de desempenho passa por algumas etapas, entre elas a verificação e a validação (BUKH, 1992). A verificação consiste em analisar se o modelo está realizando o fluxo correto de operações, tal qual o sistema real analisado executa. Já a validação consiste em comparar os resultados obtidos pelo modelo, podendo ser através de simulação ou análise estacionária, com os resultados encontrados na medição. Neste trabalho foi montado um ambiente de medição para fins de validação.

O ambiente adotado para validação do modelo de desempenho é composto por um computador para execução do YCSB, que foi configurado para executar as cargas de trabalho e computar as métricas de desempenho, e outro computador hospedando uma máquina virtual (VM) com o Cassandra. Os computadores utilizados estão conectados através de uma rede local *Gigabit Ethernet* e possuem configurações iguais: processador Intel core i5, 8GB de memória RAM, 500GB de espaço em disco e com sistema operacional Ubuntu 14.04 LTS. Já a máquina virtual para execução do Cassandra possui 2GB de memória RAM, 30GB de espaço em disco e sistema operacional Ubuntu 14.04 LTS. A Figura 6.1 ilustra uma visão geral do ambiente de medição montado para validação do modelo.

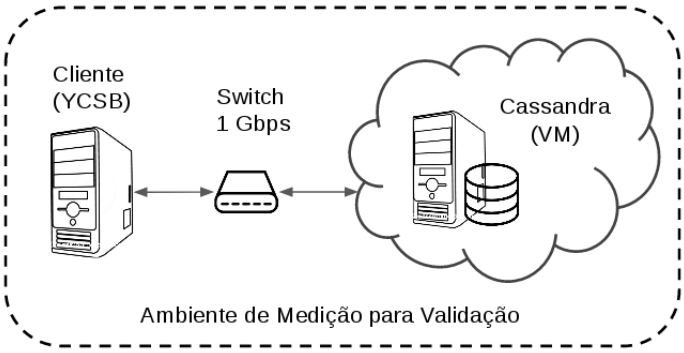


Figura 6.1: Visão geral do ambiente de medição para validação do modelo.

O YCSB foi configurado para realizar requisições de inserção de dados a uma taxa fixa. Os registros gerados pelo YCSB foram do tipo texto e inseridos em uma tabela com 10 colunas. Cada registro continha 1KB de informação. Nos cenários adotados, a ferramenta gerou requisições de 20, 40, 80, 120, 160 e 200 operações de inserção por segundo. A Tabela 6.1 apresenta o resultado da medição do tempo de resposta para cada cenário analisado.

Tabela 6.1: Resultados da Medição.

Tx. Req (Ops/s).	**Média μs**	**Intervalo de Confiança**
20	1698,16	[1665,16; 1731,16]
40	1734,11	[1672,11; 1796,11]
80	1839,43	[1795,43; 1883,43]
120	1889,46	[1816,46; 1962,46]
160	1976,12	[1918,12; 2034,12]
200	2124,49	[2004,49; 2244,49]

Ao final dos experimentos, foram realizadas análises do *log* do Cassandra. Dessa forma, os tempos dos eventos internos do modelos foram atribuídos de acordo com as informações obtidas a partir do *log* do Cassandra. A Figura 6.2 mostra os valores inseridos nas transições T0, T1, T2 e T3. Já os tempos atribuídos nas transições T4 e T5 serão detalhados a seguir, assim como as informações do valores atribuídos de cada transição separadamente.

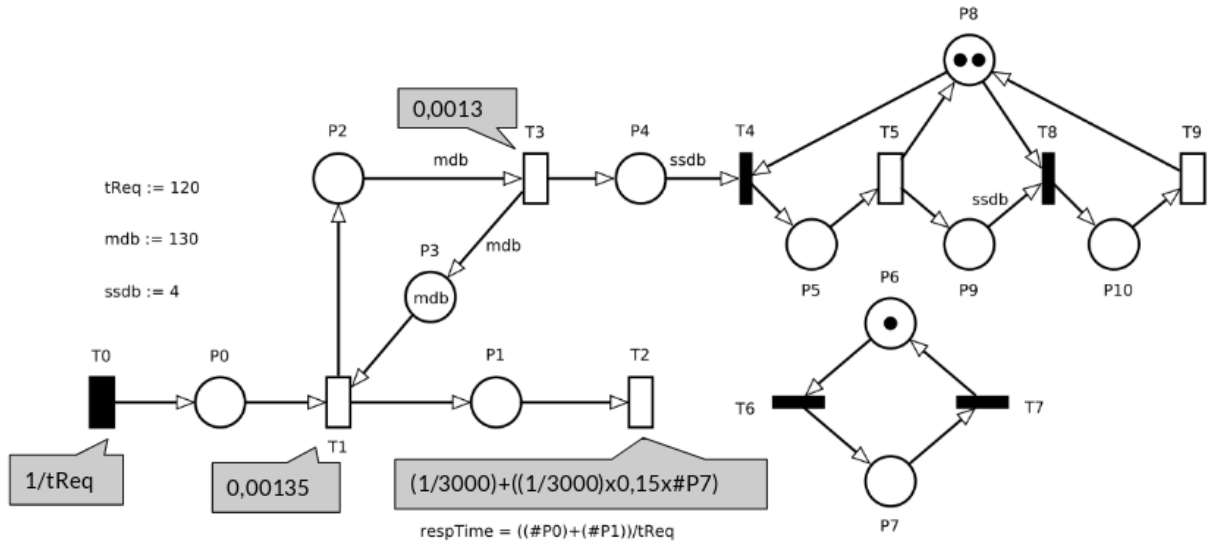


Figura 6.2: Valores atribuídos em T0, T1, T2 e T3.

Como a taxa de requisição é constante nos cenários avaliados, a transição exponencial do modelo proposto (Figura 5.3), foi alterada para uma transição determinística, representada pelo retângulo preto na Figura 6.2. Foram atribuídos seis valores analisados no ambiente de medição. O valor inserido é o inverso da taxa de requisição (representado pela variável *tReq* no modelo), que representa o intervalo entre chegadas de cada requisição. Portanto, foi inserido o valor $1/tReq$ na transição *T0*, conforme mostra a Tabela 6.2.

Tabela 6.2: Tempos da Transição Determinística *T0* ($1/tReq$).

***tReq* (Ops/s)**	**Tempo (s)**
20	0,05
40	0,025
80	0,0125
120	0,008333333
160	0,00625
200	0,005

Tempo de Escrita na *Memtable*. As medições para todos cenários analisados mostraram que o Cassandra escreve um registro na *Memtable* em 1,35 ms, em média. Esta escrita é representada por *T1*. Logo, foi atribuído o valor de 0,00135 na transição *T1* para obter o tempo de escrita na *Memtable* em segundos.

Tempo de Envio da Resposta. Para a resposta chegar na aplicação cliente, representado por *T2*, são necessários 0,33 ms (1/3000). Foi observado que o tempo de envio pode sofrer impacto caso o Cassandra esteja realizando pelo menos um processo de compactação. A Tabela 6.3 mostra o impacto para cada cenário analisado, observa-se que o impacto médio é de aproximadamente 15%. Então, o tempo de disparo em *T2* é dado pelo tempo base de 0,33 ms mais o impacto de 15% caso esteja ocorrendo uma compactação. O modelo SPN proposto identifica eventos de compactação quando o lugar *P7* possui um *token* (representado por #P7 na fórmula) quando a condição 1 da Tabela 5.2 é atendida. Portanto, foi atribuída a equação 6.1 para a transição *T2*.

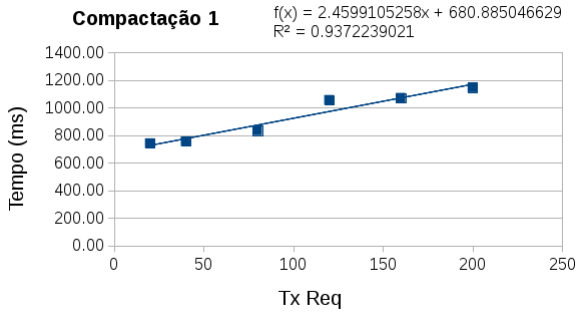
$$T2 = (1/3000) + ((1/3000) \times 0,15 \times \#P7) \tag{6.1}$$

Tabela 6.3: Impacto da Compactação no tempo de resposta.

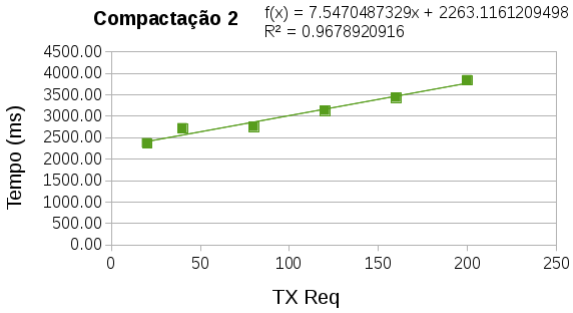
Tx. Req.(Ops/s)	**Tempo sem Comp.μs**	**Tempo com Comp.μs**	**Impacto(%)**
20	1590,94	1894,97	16,04
40	1599,58	1907,82	16,16
80	1754,62	2032,12	13,66
120	1809,03	2122,71	14,78
160	1871,28	2181,13	14,21
200	2019,71	2443,97	17,36
Média	**1774,19**	**2097,12**	**15,37**

Tempo de *Flush*. Durante as medições observou-se que o tempo de *flush* é o mesmo para todos os cenários analisados. O tempo encontrado foi de 13 ms. Portanto, a transição estocástica *T3* foi parametrizada com um atraso de 0,013 (segundos).

Tempo de Compactação. O Cassandra realiza duas compactações diferentes: uma compactação somente das *SSTables* geradas a partir do *flush* da *Memtable* (denominada neste trabalho **Compactação 1**) e a outra realizada com diversas *SSTables* compactadas (denominada **Compactação 2**). Foi observado que os tempos das compactações variam linearmente de acordo com a taxa de chegada de requisição, como mostram as Figuras 6.3(a) e 6.3(b).



(a) Compactação 1



(b) Compactação 2

Figura 6.3: Tempos das Compactações 1 (a) e 2 (b)

Para modelar esta correlação, o tempo da compactação das transições estocásticas *T5* e *T9* (correspondentes aos eventos de Compactação 1 e Compactação 2, respectivamente) foi parametrizado por fórmulas obtidas por meio da regressão linear entre o tempo de compactação e a taxa de requisição. As Figuras 6.3(a) e 6.3(b) mostram a linha de regressão e o valor do coeficiente de determinação em cada caso (R^2). Como se pode observar, os valores foram acima de 0,9 em ambos os casos, indicando a qualidade do modelo.

A Tabela 6.4 apresenta as fórmulas utilizadas na parametrização das transições *T5* e *T9*. O parâmetro *tReq*, conforme explicado anteriormente, é a taxa de requisição do sistema. Note-se que para adequação das unidades, as funções inseridas nas transições temporizadas

T5 e *T9* são dividas por 1000 para obter o tempo de compactação em segundos.

Tabela 6.4: Tempos das Compactações.

Transição	**Compactação**	**Tempo**
T5	1	$((2{,}46 \times tReq) + 680{,}88)/1000$
T9	2	$((7{,}55 \times tReq) + 2263{,}12)/1000$

Para realizar a validação do modelo, foram realizadas simulações para cada cenário analisado no ambiente de medição. O nível de confiança dos resultados é de 95% e o erro relativo da simulação é de 2%. A Figura 6.4 apresenta os resultados encontrados na modelagem e na medição. Para melhor observação do leitor, a Tabela 6.5 apresenta os valores da modelagem, da medição e os respectivos intervalos de confiança, que, como se pode notar, se sobrepõem em todos cenários. Indicando que o modelo SPN apresentado no capítulo anterior consegue representar o comportamento do sistema.

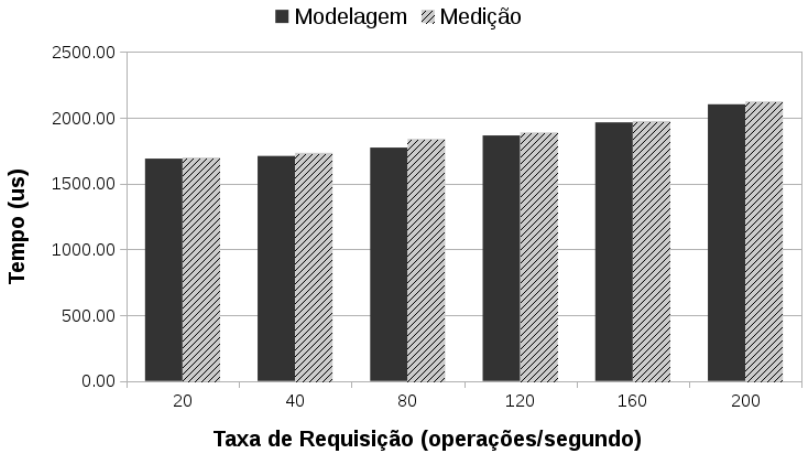


Figura 6.4: Resultado da Modelagem e da Medição de Desempenho.

Tabela 6.5: Validação do Tempo de Resposta com Resultados da Modelagem e Medição

	Modelagem		**Medição**	
Tx. Req.(Ops/s)	**Tempo(μs)**	**Int. Conf.**	**Tempo(μs)**	**Int. Conf.**
20	1689,79	[1666,89; 1712,69]	1698,16	[1665,16; 1731,16]
40	1710,07	[1680,89; 1740,68]	1734,11	[1672,11; 1796,11]
80	1773,87	[1746,46; 1818,56]	1839,43	[1795,43; 1883,43]
120	1865,56	[1832,01; 1899,11]	1889,46	[1816,46; 1962,46]
160	1964,98	[1913,57; 2016,39]	1976,12	[1918,12; 2034,12]
200	2102,43	[2017,45; 2187,41]	2124,49	[2004,49; 2244,49]

6.2 Estudo de Caso II

O estudo de caso apresentado nesta seção tem a finalidade de, usando o modelo proposto, avaliar o impacto do limite da *Memtable* no tempo de resposta observado. É possível realizar os ajustes deste parâmetro através de arquivos de configuração do Cassandra. Neste estudo de caso, ajustou-se o limite da *Memtable* no modelo para receber 10 vezes mais registros do que o cenário analisado na validação (o parâmetro *mdb* foi ajustado para 1300). Esse parâmetro governa o momento em que as operações de *flush* ocorrerão no banco. Ao aumentar o valor desse parâmetro, está-se permitindo que mais registros sejam mantidos em memória, antes de serem efetivamente gravados em disco.

O aumento do limite da *Memtable* tem impacto direto no tempo de resposta, pois quanto maior a quantidade de registros inseridos na memória, menor será a quantidade de operações de *flush* durante a inserção, e menor será o tempo de resposta, já que enquanto um *flush* está acontecendo, o Cassandra bloqueia a inserção de novos registros. Logo, quanto menos operações de *flush* ocorrer, um menor número de novas requisições será bloqueado. Por outro lado, este aumento exigirá um maior uso de memória RAM do computador/VM em que está instalado, sendo então fisicamente limitado pelas configurações de *hardware*.

O gráfico da Figura 6.5 ilustra a redução do tempo de resposta encontrado no modelo, com o incremento no limite da *Memtable* em relação aos resultados encontrados na modelagem (do estudo de caso I) para as mesmas taxas de requisição analisadas. O gráfico sugere uma redução gradativa do tempo de resposta a medida que a taxa de requisição aumenta.

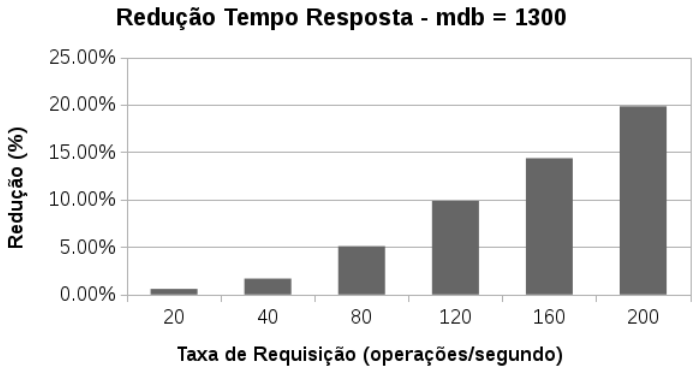


Figura 6.5: Redução do Tempo de Resposta com a *Memtable* 10 vezes maior.

Analisando a Figura 6.5, observa-se que não houve redução significativa nos cenários com as menores taxa de requisição. Por exemplo, no cenário com a taxa de requisição de 20 operações/s, a redução foi em torno de 0,5%. Já nos cenários com as maiores taxas de requisição (200 operações/s), a redução atingiu valores próximo de 20%. É possível observar que esse valor representa o mesmo tempo de resposta encontrado no cenário de 80 operações/s com o limite da *Memtable* igual a 130 operações, já que os intervalos de confiança estão se sobrepondo, conforme a Tabela 6.6. Esta tabela apresenta os valores encontrados na modelagem para *mdb*=130 e os encontrados no modelo com a configuração analisada neste estudo de caso.

Tabela 6.6: Análise do Tempo de Resposta para limite da *Memtable* igual a 1300

	Modelagem, mdb=1300		**Modelagem, mdb=130**	
Tx. Req.(Ops/s)	**Tempo(μs)**	**Int. Conf.**	**Tempo(μs)**	**Int. Conf.**
20	1679,93	[1675,68; 1684,41]	1689,79	[1666,89; 1712,69]
40	1681,91	[1677,97; 1685,84]	1710,07	[1680,89; 1740,68]
80	1688,01	[1684,30; 1691,71]	1773,87	[1746,46; 1818,56]
120	1698,12	[1693,24; 1703,00]	1865,56	[1832,01; 1899,11]
160	1718,22	[1712,33; 1724,11]	1964,98	[1913,57; 2016,39]
200	1754,53	[1746,24; 1762,82]	2102,43	[2017,45; 2187,41]

6.3 Estudo de Caso III

O estudo de caso apresentado nesta seção avalia o impacto do tempo de resposta quando aumenta-se o número mínimo de *SSTables* necessárias para iniciar a compactação. Neste estudo de caso, o número mínimo de *SSTables* para iniciar a compactação foi dobrado, ou seja, passou de 4 para 8 *SSTables*. No modelo, este número é representado pelo parâmetro *ssdb*. Em ambiente real, esta configuração é definida no momento da criação de uma tabela no Cassandra.

O número mínimo de *SSTables* indica quantas *SSTables* devem ser geradas pelo processo de *flush* antes de iniciar o processo de Compactação 1 (compactação das *SSTables* geradas a partir do *flush*). Além disso, este parâmetro indica quantas *SSTables* compactadas devem ser geradas pela Compactação 1, antes de iniciar-se o processo de Compactação 2.

Conforme visto na validação do modelo (Seção 6.1) o tempo de resposta sofre um aumento de aproximadamente 15% durante o tempo de envio da resposta para aplicação cliente enquanto pelo menos uma compactação esteja ocorrendo durante a inserção de dados. Logo, quanto maior o número mínimo de *SSTables* para iniciar o processo de compactação, menor será o impacto no tempo de resposta médio do sistema. A Figura 6.6 ilustra a redução do tempo de resposta para cada taxa de requisição.

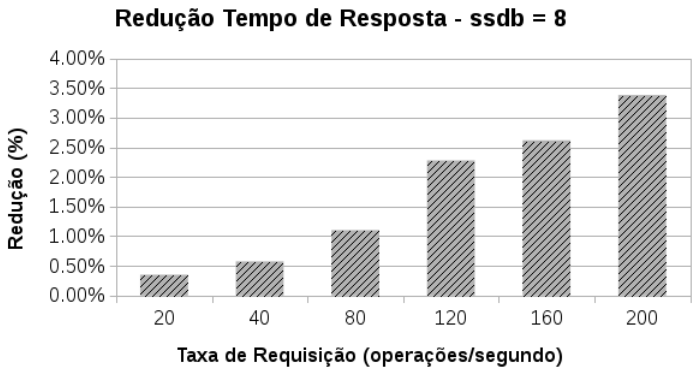


Figura 6.6: Redução do Tempo de Resposta com $ssdb = 8$.

Analisando o gráfico da Figura 6.6, é possível observar que o maior impacto foi para o cenário com 200 operações por segundo, chegando a uma redução de aproximadamente 4%. As reduções para os cenários de menores taxas de requisições (20 e 40 operações/seg) foram muito próximas, diferenciando aproximadamente de 0,2%. Em geral, os resultados demonstraram que a redução do tempo de resposta foi menor do que os cenários analisados no estudo de caso II. Esse fato sugere que a estratégia de aumentar o número mínimo de *SSTables* pode trazer resultados inferiores do que aumentar o limite da *Memtable*. A Figura 6.6 mostra um redução gradativa do tempo de resposta a medida que a taxa de requisição aumenta, comportamento similar ao encontrado no estudo de caso anterior.

6.4 Estudo de Caso IV

Este último estudo de caso realiza uma avaliação conjunta das configurações analisadas nos dois estudos de caso anteriores. Em outras palavras, este estudo avalia o impacto do tempo de resposta quando se aumenta o limite da *Memtable* para 1300 registros e o número mínimo de 8 *SSTables* necessárias para iniciar a compactação, para as mesmas taxas de requisição utilizadas no experimento base. Além disso, é realizada uma comparação entre os estudos II, III e IV a fim de identificar a melhor configuração (entre as analisadas).

O gráfico da Figura 6.7 ilustra a redução do tempo de resposta em relação ao estudo de caso I, para a configuração analisada variando a taxa de requisição. O gráfico segue comportamento similar ao encontrado no Estudo de Caso II, em que o limite da *Memtable* foi aumentado para suportar 10 vezes mais registros.

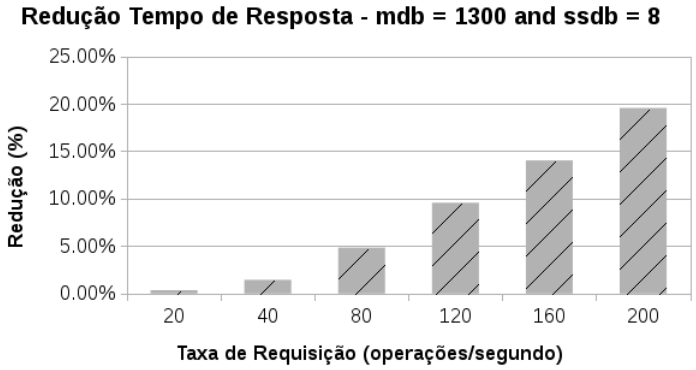


Figura 6.7: Redução do Tempo de Resposta com $mdb = 1300$ e $ssdb = 8$.

É possível observar também que quanto maior a taxa de requisição, maior a redução do tempo de resposta. Para o cenário com 200 operações/s, o tempo de resposta médio foi de 1758,63 μs, representando uma redução de 19,5%. Esta redução indica que o tempo de resposta é estatisticamente igual ao encontrado no cenário de 80 operações por segundo considerando as configurações padrão do ambiente de medição, visto que a média está dentro do intervalo de confiança (ver Tabela 6.6). Enquanto que para o menor cenário, 20 operações por segundo, a redução foi inferior a 1%.

Comparação entre os estudos II, III e IV. Analisando os Estudos de Caso II, III e IV, é possível concluir que a estratégia de aumentar o limite da *Memtable* traz melhores resultados quando comparada à estratégia de aumentar o número mínimo de *SSTables* antes de iniciar a compactação. Ao utilizar as duas configurações juntas (estudo de caso IV), os resultados obtidos foram estatisticamente similares aos encontrados no estudo de caso II, visto que os intervalos de confiança se sobrepõem (ver Tabela 6.7). O gráfico da Figura 6.8 apresenta uma comparação das reduções obtidas nas três configurações: Conf I, Conf II e Conf III, analisadas nos estudos de caso II, III e IV, respectivamente. É possível concluir que a Conf II não apresentou resultados significativos, mesmo quando esta configuração foi inserida junto com o aumento do limite da *Memtable*.

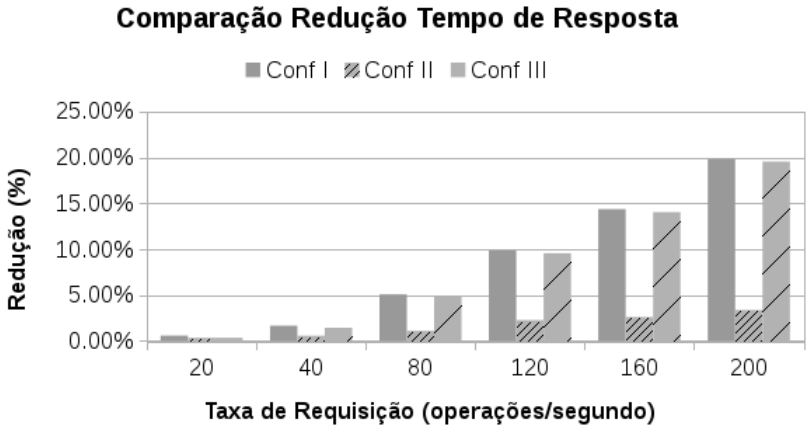


Figura 6.8: Comparação dos resultados entre as configurações analisadas.

Tabela 6.7: Comparação entre os estudos de caso II e IV

	Estudo Caso II		**Estudo Caso IV**	
Tx. Req.(Ops/s)	**Tempo(μs)**	**Int. Conf.**	**Tempo(μs)**	**Int. Conf.**
20	1679,93	[1675,68; 1684,41]	1683,83	[1680,57; 1687,10]
40	1681,91	[1677,97; 1685,84]	1685,77	[1682,64; 1688,91]
80	1688,01	[1684,30; 1691,71]	1691,99	[1688,83; 1695,15]
120	1698,12	[1693,24; 1703,00]	1702,74	[1698,74; 1706,75]
160	1718,22	[1712,33; 1724,11]	1723,16	[1717,78; 1728,53]
200	1754,53	[1746,24; 1762,82]	1758,63	[1752,47; 1764,48]

6.5 Considerações Finais

Este capítulo apresentou quatro estudos de caso. O primeiro estudo proporcionou a validação do modelo. A validação foi realizada utilizando técnicas de medição para obtenção das métricas em ambiente de medição e simulação para obtenção das métricas no modelo. Dessa forma, foram definidos os tempos das transições temporizadas para os ambientes analisados. Já os estudos II, III e IV proporcionaram uma avaliação do tempo de resposta em diferentes configurações do Cassandra utilizando o modelo desenvolvido. Foram variados o limite da *Memtable* e a quantidade mínima de *SSTables*. Por último, foi feita uma comparação entre os três últimos estudos a fim de obter a melhor configuração entre as analisadas quando o foco é desempenho no tempo de resposta durante o processo de inserção de dados. Foi observado que o aumento da *Memtable* (estudo II) traz melhores resultados em relação ao aumento do número mínimo de *SSTables* (estudo III) antes de iniciar a compactação. Por último, o Apêndice A apresenta tabelas que sumarizam a descrição dos elementos do modelo e os tempos atribuídos nas transições temporiza, e podem ser utilizadas para rápida referência.

Capítulo 7

Conclusão

Nos últimos anos tem se observado um grande aumento de dados gerados por diversas plataformas. A cada dia novos sistemas e dispositivos com acesso à *Internet* estão se comunicando e gerando informações. O aumento do número de dispositivos, das trocas de informação em redes sociais e do número de sensores está gerando cada vez mais dados trafegados na *Internet*. Estima-se que em 2020, o tráfego anual global vai ultrapassar 3,3 *zettabytes*(ZB) (CISCO, 2017), equivalente a 3,3E+9 *terabytes*(TB). Diante deste cenário, tecnologias emergentes de armazenamento estão surgindo como alternativa para gerenciar esta grande demanda de dados, entre elas encontram-se os bancos de dados NoSQL.

A avaliação de desempenho de banco de dados é uma área de pesquisa ativa que tem se intensificado devido ao aparecimento de modelos alternativos aos bancos de dados convencionais. Entretanto, a maioria dos trabalhos relacionados à avaliação de desempenho de bancos de dados utilizam-se de técnicas de medição, comparando os mais variados modelos de sistemas de armazenamentos. Contudo, existe a necessidade de avaliar os mais diversos sistemas de armazenamento de forma mais rápida e com menor custo.

Neste contexto, este trabalhou realizou uma análise de desempenho do banco de dados NoSQL Cassandra utilizando técnicas de medição e modelagem. Foi analisada a operação de inserção de dados no *cluster*. Para obtenção de métricas de desempenho durante as medições e geração de carga de trabalho utilizou-se a ferramenta de *benchmark Yahoo! Cloud Serving Benchmark*. Nesta etapa, foram analisados cenários simples, com apenas uma único nó, e

clusters com quantidade variadas de nós. Além disso, foi realizada uma análise de consumo de energia desses cenários a fim de encontrar a quantidade ideal de nós levando em consideração aspectos de desempenho e consumo de energia.

Também foi proposto um modelo de desempenho em SPN para representar o comportamento das operações realizadas pelo banco de dados Cassandra no momento da inserção de registros. Para fins de validação do modelo, foram feitas medições do tempo de resposta durante a inserção de dados em um único nó do Cassandra. No final desta etapa, foram extraídos os *logs* provenientes do YCSB e do Cassandra para realizar análise categorizada do tempo de resposta. Esta análise consistiu em avaliar o tempo de resposta de acordo com os processos internos do Cassandra durante a inserção de dados. Para isso, foi desenvolvida uma ferramenta para automatização do processo [1]. Após a obtenção dos tempos de resposta categorizados, foi analisado o tempo dos processos internos do Cassandra. Entre os processos estão o *flush*, as compactações, tempo de escrita do registro na *Memtable* e de envio de resposta para o cliente.

Com o modelo de desempenho validado, foram analisadas diferentes configurações do Cassandra nos estudos de caso II, III e IV. As configurações analisadas foram o aumento do limite da *Memtable*, aumento do número mínimo de *SSTables* necessário para iniciar a compactação e as duas configurações juntas. Os resultados mostraram que o aumento do limite da *Memtable* obteve maior redução do tempo de resposta comparando com o aumento do número mínimo de *SSTables*. O configuração proposta pelo estudo de caso II atingiu uma redução de aproximadamente 20% no tempo de resposta quando analisado o cenário com 200 operações/s. Já a configuração proposta pelo estudo de caso III atingiu uma redução máxima de 3,5%. Por último, a configuração proposta pelo estudo de caso IV atingiu valores estatisticamente similares ao estudo de caso II. No entanto, o aumento do limite da *Memtable* exigirá um maior uso de memória RAM do computador/VM em que está instalado, sendo então fisicamente limitado pelas configurações de *hardware*.

[1] https://github.com/jukabarros/calcLatencia

7.1 Contribuições

Apresentam-se as principais contribuições deste trabalho na listagem a seguir:

- Avaliação de desempenho e consumo de energia do Cassandra em cenários com uma única instância do banco de dados e em cenários de dados distribuídos em várias instâncias;
- Ferramenta para obtenção dos tempos de resposta categorizados baseados nos processos internos do Cassandra (*flush* e compactação). O código está disponibilizado no *github*[2];
- Modelo de desempenho para obtenção do tempo de resposta do Banco de Dados Cassandra durante o processo de inserção de dados. O modelo foi desenvolvido em rede de Petri estocástica;
- Publicação de Artigo na Revista de Informática Teórica e Aplicada:
  BARROS, J. R. A.; CALLOU, G. ; GONÇALVES, G. ; MEDEIROS, V. ; CASTELETTI, H. . Análise de desempenho de Banco de Dados Relacionais e Não Relacionais em dados genômicos. REVISTA DE INFORMÁTICA TEÓRICA E APLICADA: RITA, v. 24, p. 11-27, 2017.
- Publicação de Artigo no Workshop em Clouds e Aplicações (WCGA) 2017:
  BARROS, J. R. A.; CALLOU, G. ; GONÇALVES, G. . Análise Integrada de Desempenho e Consumo de Energia em Sistemas de Armazenamento de Dados Distribuídos. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Workshop em Clouds e Aplicações, 2017, Belém. Anais do WCGA. Porto Alegre: Sociedade Brasileira de Computação, 2017. v. XV. p. 87-100.

[2]https://github.com/jukabarros/calcLatencia

7.2 Trabalhos Futuros

Este trabalho apresenta resultados que indicam continuidade para avaliar o desempenho de banco de dados utilizando técnicas de modelagem e simulação. Além disso, o trabalho apresenta possibilidades de avaliar outros fatores, como por exemplo, o consumo de energia desses sistemas.

A utilização de técnicas de modelagem e simulação apresentou-se como uma alternativa para avaliar o desempenho de um sistema de armazenamento de dados o qual foi projetado para gerenciar grande quantidade de dados. Os resultados obtidos justificam a continuidade de estudos em ambientes distribuídos, como também a análise de outras métricas de desempenho como a vazão. Além disso, existe a alternativa de avaliar o consumo de energia a partir do desenvolvimento novos modelos formais.

Logo, como trabalhos futuros, pretende-se avaliar o desempenho através de técnicas de modelagem e simulação em cenários com dados distribuídos em várias instâncias do Cassandra. Além disso, existe a possibilidade de analisar outras métricas de desempenho, como vazão e tempo de execução tanto em cenários simples, com apenas uma instância, como também em cenários distribuídos. Avaliar o desempenho de outras operações, como por exemplo, operação de leitura tanto em cenário simples como distribuídos representa como uma opção para trabalhos futuros.

Devido a popularidade e a quantidade de extensões encontradas nas redes de Petri, pretende-se propor modelos de desempenho utilizando extensões de redes de Petri, como por exemplo, redes de Petri coloridas. Além disso, pretende-se realizar modelagem de outros fatores, como o consumo de energia desses ambientes, utilizando EFM.

Por último, pretende-se analisar outros bancos de dados NoSQL que apresentam modelos e estratégias diferentes do Cassandra. Além disso, avaliar o desempenho e consumo de energia de outros tipos de bancos de dados emergentes, entre eles os que fazem parte da classe *NewSQL*.

Referências Bibliográficas

ABRAMOVA, V.; BERNARDINO, J. Nosql databases: Mongodb vs cassandra. In: ACM. *Proceedings of the international C* conference on computer science and software engineering*. [S.l.], 2013. p. 14–22.

ABUBAKAR, Y.; ADEYI, T. S.; AUTA, I. G. Performance evaluation of NoSQL systems using YCSB in a resource austere environment. In: *Performance Evaluation*. [s.n.], 2014. v. 7, n. 8. Disponível em: ⟨https://pdfs.semanticscholar.org/7f04/ab70ba5bc2ba711915a7bb82a346d6a16aaa.pdf⟩.

ACADEMY, D. *A Brief Introduction to Apache Cassandra*. 2017. Disponível em `https://academy.datastax.com/resources/brief-introduction-apache-cassandra`. Acessado em 07/01/2018.

ANDRADE, E. C. d. *Modelagem e análise de mecanismos de tratamento de interrupções em infraestruturas computacionais dos sistemas distribuídos*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2014.

ANICETO, R. C.; XAVIER, R. F. *Um estudo sobre a utilização do banco de dados NoSQL cassandra em dados biológicos*. Tese (Monograph) — Universidade de Brasília, 2014. Disponível em: ⟨http://bdm.unb.br/handle/10483/7927⟩.

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

ARAÚJO, C. G. *Avaliação do consumo de energia em sistemas de gerenciamento de banco de dados NoSQL*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2016.

ASSIS, J. de O. et al. Performance evaluation of nosql data store for digital media. In: IEEE. *Information Systems and Technologies (CISTI), 2017 12th Iberian Conference on*. [S.l.], 2017. p. 1–6.

AZURE, M. Windows azure storage – 4 trillion objects and counting. Disponível em `https://azure.microsoft.com/en-us/blog/windows-azure-storage-4-trillion-objects-and-counting/`. Acessado em 06/01/2018. 2012.

BALBO, G. Introduction to stochastic petri nets. *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands, July 3-7, 2000: Revised Lectures*, Springer, 2001.

BARROS, J.; CALLOU, G.; GONÇALVES, G. Análise integrada de desempenho e consumo de energia em sistemas de armazenamento de dados distribuídos. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Workshop em Clouds e Aplicações, 2017, Belém. Anais do WCGA*. SBRC, 2017. p. 87–100. Disponível em: ⟨https://sbrc2017.ufpa.br/wp-content/uploads/2017/05/proceedingsWCGA2017.pdf⟩.

BARROS, J.; GONÇALVES, G.; MEDEIROS, V. *Análise de desempenho de bancos de dados Relacionais e Não Relacionais em dados genômicos*. Tese (Monograph) — Universidade Federal Rural de Pernambuco, 2015.

BOLCH, G. et al. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. [S.l.]: John Wiley & Sons, 2006.

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

CALLOU, G. et al. An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost. *Energies*, Multidisciplinary Digital Publishing Institute, v. 7, n. 1, p. 238–277, 2014.

CALLOU, G. et al. Energy consumption and execution time estimation of embedded system applications. *Microprocessors and Microsystems*, Elsevier, v. 35, n. 4, p. 426–440, 2011.

CARNIEL, A. C. et al. Query processing over data warehouse using relational databases and nosql. p. 1–9, 2012.

CISCO. *The Zettabyte Era: Trends and Analysis*. [S.l.], 2017.

COCKCROFT, A.; SHEAHAN, D. *Benchmarking Cassandra Scalability on AWS–Over a million writes per second.* [S.l.], 2011.

COOPER, B. F. et al. Benchmarking cloud serving systems with YCSB. In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010. p. 143–154. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=1807152⟩.

DATASTAX. Configuring compaction. Disponível em `https://docs.datastax.com/en/cassandra/2.1/cassandra/operations/ops_configure_compaction_t.html`. Acessado em 08/01/2018. 2018.

DATASTAX. The write path to compaction. Disponível em `http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_write_path_c.html`. Acessado em 07/01/2018. 2018.

DIANA, M. D.; GEROSA, M. A. Nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. In: *IX Workshop de Teses e Dissertações em Banco de Dados*. [s.n.], 2010. v. 9. Disponível em: ⟨http://www.lbd.dcc.ufmg.br/colecoes/wtdbd/2010/sbbd_wtd_12.pdf⟩.

FOWLER, P. J. S. M. J. *Nosql distilled a brief guide to the emerging world of polyglot persistence*. [S.l.]: Addison-Wesley Professional, 2012.

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

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

GIRAULT, C.; VALK, R. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. [S.l.]: Springer Science & Business Media, 2013.

GOMES, C.; TAVARES, E.; JUNIOR, M. N. d. O. Energy consumption evaluation of NoSQL DBMSs. In: *WPerformance, 2016, Porto Alegre. Anais do XXXVI congresso da sociedade brasileira de computação*. CSBC, 2016. p. 2828–2838. Disponível em: ⟨http://www.lbd.dcc.ufmg.br/colecoes/wperformance/2016/001.pdf⟩.

JANOUŠEK, V. *Modelling Objects by Petri Nets*. Tese (Doutorado) — Brno University of Technology, Brno, Czech Republic, 1998.

JENSEN, K. Coloured petri nets: A high level language for system design and analysis. In: SPRINGER. *International Conference on Application and Theory of Petri Nets*. [S.l.], 1989. p. 342–416.

KAUR, K.; SACHDEVA, M. Performance evaluation of newsql databases. In: IEEE. *Inventive Systems and Control (ICISC), 2017 International Conference on*. [S.l.], 2017. p. 1–5.

KORIEM, S. M.; DABBOUS, T.; EL-KILANI, W. S. A new petri net modeling technique for the performance analysis of discrete event dynamic systems. *Journal of systems and software*, Elsevier, v. 72, n. 3, p. 335–348, 2004.

LABS, S. Date-tiered compaction in apache cassandra. Disponível em `https://labs.spotify.com/2014/12/18/date-tiered-compaction/`. Acessado em 08/01/2018. 2014.

LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, ACM, v. 44, n. 2, p. 35–40, 2010.

LI, X.; MEDINA, J. M.; CHAPA, S. V. Applying petri nets in active database systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 37, n. 4, p. 482–493, 2007.

LI, Y.; MANOHARAN, S. A performance comparison of SQL and NoSQL databases. In: *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*. IEEE, 2013. p. 15–19. Disponível em: ⟨http://ieeexplore.ieee.org/abstract/document/6625441/⟩.

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

LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. NoSQL no desenvolvimento de aplicações web colaborativas. In: *VIII SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, Paraty, RJ: SBC*. [s.n.], 2011. Disponível em: ⟨http://www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf⟩.

MACIEL, P. et al. Performance evaluation of sheepdog distributed storage system. In: IEEE. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 3370–3375.

MACIEL, P. et al. Performance and dependability in service computing: Concepts, techniques and research directions. In: _____. [S.l.]: Information Science Reference, 2011. cap. 3 - Dependability Modeling, p. 53–97.

MARSAN, M. et al. Modelling with Generalized Stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review*, ACM Press New York, NY, USA, v. 26, n. 2, 1998.

MARSAN, M. A. Stochastic petri nets: an elementary introduction. In: SPRINGER. *European Workshop on Applications and Theory in Petri Nets*. [S.l.], 1988. p. 1–29.

MELO, F. F.; GOUVEIA, R. M. M.; ALENCAR, A. L. d. *Análise de Desempenho de Banco de Dados Não Relacionais no Cenário de Dados Abertos Educacionais*. Tese (Monograph) — Universidade Federal Rural de Pernambuco, 2016.

MERLIN, P.; FARBER, D. Recoverability of communication protocols–implications of a theoretical study. *IEEE transactions on Communications*, IEEE, v. 24, n. 9, p. 1036–1043, 1976.

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

NETFLIX GIVES USERS EXACTLY WHAT THEY WANT – EVERY TIME. [S.l.], 2011.

NIEMANN, R. Evaluating the performance and energy consumption of distributed data management systems. In: IEEE. *Global Software Engineering Workshops (ICGSEW), 2015 IEEE 10th International Conference on*. [S.l.], 2015. p. 27–34.

NIEMANN, R. Towards the prediction of the performance and energy efficiency of distributed data management systems. In: ACM. *Companion Publication for ACM/SPEC on International Conference on Performance Engineering*. [S.l.], 2016. p. 23–28.

PETRI, C. A. *Kommunikation mit automaten*. Tese (Doutorado), 1962.

POINT, E. *Benchmarking Top NoSQL Databases: Apache Cassandra, Couchbase, HBase, and MongoDB*. [S.l.], 2015.

RABL, T. et al. Solving big data challenges for enterprise application performance management. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 5, n. 12, p. 1724–1735, 2012.

SILVA, B. *A framework for availability, performance and survivability evaluation of disaster tolerant cloud computing systems*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2016.

SILVA, B. et al. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: *Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN*. [S.l.: s.n.], 2015.

SOARES, B. E.; BOSCARIOLI, C. Modelo de Banco de Dados Colunar: Características, Aplicações e Exemplos de Sistemas. *Escola Regional de BDs, Camobiu. IX ERBD*, 2013. Disponível em: ⟨http://www.lbd.dcc.ufmg.br/colecoes/erbd/2013/007.pdf⟩.

SOUSA, E. *Modelagem de desempenho, dependabilidade e custo para o planejamento de infraestruturas de nuvens privadas*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2015.

STRAUCH, C.; SITES, U.-L. S.; KRIHA, W. NoSQL databases. *Lecture Notes, Stuttgart Media University*, 2011. Disponível em: ⟨http://webpages.uncc.edu/xwu/5160/nosqldbs.pdf⟩.

TORRES, E. B. et al. Performance and availability evaluation of storage services in private cloud. In: IEEE. *Information Systems and Technologies (CISTI), 2016 11th Iberian Conference on*. [S.l.], 2016. p. 1–6.

TRIVEDI, K. S. *Probability & statistics with reliability, queuing and computer science applications*. [S.l.]: John Wiley & Sons, 2008.

WORLD, C. Hybrid cloud adoption set for a big boost in 2015. Disponível em `https://www.computerworld.com/article/2860980/hybrid-cloud-adoption-set-for-a-big-boost-in-2015.html`. Acessado em 05/01/2018. 2014.

ZIMMERMANN, A. et al. Towards version 4.0 of timenet. In: VDE. *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2006 13th GI/ITG Conference*. [S.l.], 2006. p. 1–4.

Apêndice A

Descrição dos Elementos do Modelo

A.1 Lugares

Lugar	**N°** ***Token*** **Inicial**	**Descrição**
P0	0	Fila de Requisições
P1	0	Fila de Resposta
P2	0	Registro escrito na *Memtable*
P3	*mdb*	*Buffer* da *Memtable*
P4	0	Fila de *SSTables*
P5	0	Início da Compactação 1
P6	1	Estado do *Clock* sem compactações
P7	0	Estado do *Clock* com compactações
P8	2	Quantidade de compactações simultâneas
P9	0	Fila de *SSTables* compactadas
P10	0	Início da Compactação 2

A.2 Transições

Transição	Tipo	Tempo	Descrição
T0	Determinística	$1/tReq$	Chegada de 1 requisição
T1	Exponencial	0,00135	Escrita na *Memtable*
T2	Exponencial	$(1/3000)+(1/3000)\times 0{,}15\times \#P7$	Envio da resposta
T3	Exponencial	0,013	Flush
T4	Imediata	-	Início da Compactação 1
T5	Exponencial	$((2{,}459\times tReq)+680{,}88)/1000$	Compactação 1
T6	Imediata	-	Muda o *clock*
T7	Imediata	-	Muda o *clock*
T8	Imediata	-	Início da Compactação 2
T9	Exponencial	$((7{,}547\times tReq)+2263{,}12)/1000$	Compactação 2

A.3 Parâmetros

Parâmetro	Valor Padrão	Descrição
tReq	20; 40; 80; 120; 160; 200	Quantidade de requisições por segundo
mdb	130	Quantidade máxima de registro na *Memtable*
ssdb	4	Quantidade de *SSTables* para iniciar Compactação

Apêndice B

Teste T com Duas Amostras Independentes

O Teste T com duas amostras independentes é um teste de hipótese que indica se um determinado conjunto de amostras apresenta valores estatisticamente diferentes de outro conjunto de amostras baseado no nível de confiança dos resultados das amostras (representado por α). Por exemplo, considerando que A e B são dois conjuntos de amostras independentes, a fórmula para calcular o teste t é mostrada na equação B.1:

$$t = \frac{m_A - m_B}{\sqrt{\frac{S^2}{n_A} + \frac{S^2}{n_B}}} \tag{B.1}$$

onde m_A e m_B são as médias de A e B, respectivamente, e n_A e n_B são os tamanhos das amostras de A e B, respectivamente. Enquanto que S^2 é um valor estimado da variância comum das duas amostras, que é calculado através da equação B.2.

$$S^2 = \frac{\sum (x - m_A)^2 + \sum (x - m_B)^2}{n_A + n_B - 2} \tag{B.2}$$