Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Replicação de Dados em Workflows de Bioinformática usando os Bancos de Dados NoSQL

Iasmini Virgínia O. Lima

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Brasília
2016

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Replicação de Dados em Workflows de Bioinformática usando os Bancos de Dados NoSQL

Iasmini Virgínia O. Lima

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof.a Dr.a Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.a Dr.a Maria Emília Walter
CIC/UnB

Dr. Sérgio Lifschitz
PUC/Rio

Prof.a Dr.a Célia Ghedini Ralha
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 02 de dezembro de 2016

Dedicatória

Dedico este trabalho aos meus pais Elaine e José, e à minha irmã Bruna, pois sem eles jamais teria conseguido alcançar meus objetivos.

Agradecimentos

Depois desses dois anos de muito estudo, empenho e esforço, gostaria de agradecer a algumas pessoas que me acompanharam e foram essenciais para a realização de mais esta importante etapa concluída. Não foi fácil chegar até aqui. Desde o processo seletivo, passando pela aprovação até a conclusão do mestrado, foi um longo caminho percorrido. Nada foi fácil, nem tampouco tranquilo, porém sempre tive Deus ao meu lado, e pessoas que sempre confiaram em mim e na minha capacidade.

Agradeço, portanto, antes de tudo à Deus. Por ter me dado o dom da vida, sabedoria, paciência e perseverança para correr atrás dos meus objetivos. Agradeço por me abençoar com tantos presentes divinos, me dando talvez além do que mereço.

Agradeço à minha mãe Elaine e ao meu pai José, pois sei que não foi fácil me ver partir para outra cidade. E mesmo de longe, estiveram sempre presente na minha vida, me ajudando em tudo. Agradeço também por serem um exemplo de vida para mim e por me dar apoio em todas as decisões.

À minha irmã Bruna, porque com a certeza de que um dia estaríamos morando juntas novamente, sempre me mantive firme e forte na minha caminhada. E por ser uma das pessoas mais importantes na minha vida.

Às minhas avós, Maruca e Jovelina, por serem as melhores avós do mundo, e nunca mediram esforços para de ajudar em qualquer coisa.

Agradeço ao meu namorado, que torna os dias que passamos juntos melhores e mais felizes. Obrigada por estar comigo em todos os momentos de desespero e de alegrias e vitórias. Obrigada também pelas inúmeras vezes que me acompanhou à todos os lugares, inclusive nas idas à UnB, até mesmo nos fins de semana, muitas vezes apenas para "ligar computadores e fazer execuções". Obrigada também por me acalmar quando senti medo e vontade de desistir e por ter sido a minha base desde que se mudou para Brasília.

Gostaria de agradecer também a todos que conheci nesse período, de forma muito especial à minha amiga Gizele, que foi a primeira pessoa que conheci quando me mudei, e a partir de então, se tornou uma grande amiga e companheira para a vida.

Às minhas amigas da graduação, Pricila, Leidiane, Kemilly e Mariana, que mesmo depois de cada uma seguir seu caminho, nos mantivemos sempre em contato e sempre nos

ajudando em todos os momentos.

Agradeço também à minha orientadora professora Doutora Maristela, por todo apoio e auxilio, por toda paciência e por ser uma excelente professora, ótima profissional e acima tudo uma amiga. Também agradeço à professora Doutora Maria Emília, por todo apoio dado durante todo o mestrado, principalmente por tudo que fez por mim durante o *X-meeting + BSB* 2015, evento no qual tive o prazer de participar ao seu lado.

Por fim agradeço à todos que de alguma forma contribuíram para o sucesso deste trabalho.

Resumo

Projetos em Bioinformática são geralmente executados como *workflows* científicos. Os biólogos frequentemente podem executar o mesmo *workflow* diversas vezes com diferentes parâmetros. Estas execuções podem gerar um grande volume de arquivos com diferentes formatos, que precisam ser armazenados para futuras execuções. Dentre diversos sistemas de armazenamento de dados apresentados na literatura, novos modelos de bancos de dados têm sido definidos para armazenar grandes volumes de dados, os bancos de dados NoSQL (*Not Only SQL*), que surgiram da necessidade de uma alta escalabilidade e de um desempenho superior quando o volume de dados é muito grande. Nesse contexto, é proposta a análise do impacto da replicação de dados quando se utiliza bancos de dados NoSQL para o armazenamento dos diferentes formatos de arquivos resultantes da execução de *workflows* de Bioinformática. Foram executados três *workflows* de Bioinformática envolvendo diferentes formatos de dados, os quais foram utilizados para os testes nos bancos de dados NoSQL Cassandra e MongoDB, e em seguida é apresentada uma análise dessas execuções e testes. Os resultados obtidos mostraram que os tempos de inserção e extração dos dados são mínimos comparados aos tempos de execução dos *workflows* e portanto, os bancos de dados NoSQL mostraram ser uma boa opção para armazenamento e gerenciamento de dados biológicos.

Palavras-chave: Bancos de Dados NoSQL, *Workflows* em Bioinformática, Cassandra, MongoDB

Abstract

Projects in bioinformatics are generally executed as scientific workflows. The biologists often perform the same workflow several times with different parameters. These executions can generate a large volume of files with different formats, which need to be stored for future executions. Among several data storage systems presented in the literature, new database models have been defined to store large volumes of data, the NoSQL databases (Not Only SQL), which arose from the need for high scalability and superior performance when the data volume is too large. In this context, it is proposed to analyze the impact of data replication when using NoSQL databases for the storage of the different file formats resulting from the execution of bioinformatics workflows. Three bioinformatics workflows involving different data formats were executed, which were used for testing the NoSQL Cassandra and MongoDB databases, and then an analysis of those executions and tests is presented. The results show that the data insertion and extraction times are minimal compared to the execution times of the workflows, and therefore, the NoSQL databases have proved to be a good option for the storage and management of biological data.

Keywords: NoSQL Databases, Bioinformatics Workflows, Cassandra, MongoDB

Sumário

1 Introdução **1**
1.1 Objetivo 2
1.2 Estrutura do Documento 2

2 Referencial Teórico **4**
2.1 *Workflows* em Bioinformática 4
2.1.1 Programas Utilizados nos *Workflows* em Bioinformática 6
2.1.2 Formatos de Arquivos 8
2.2 Bancos de Dados NoSQL 10
2.2.1 Modelos de Dados 13
2.2.2 Cassandra 14
2.2.3 MongoDB 19
2.3 Replicação de Dados 21
2.3.1 Tipos e Estratégias de Replicação de Dados 22
2.4 Trabalhos Relacionados 24

3 Ambiente de Execução **27**
3.1 *Workflows* Executados 27
3.2 Preparação dos Bancos de Dados 31
3.2.1 Preparação do Cassandra 31
3.2.2 Preparação do MongoDB 32

4 Análise de Resultados **35**
4.1 Resultados das Execuções dos *Workflows* 35
4.1.1 Teste de Tolerância a Falhas 38
4.2 Análise dos Dados 38
4.2.1 *Workflow* 1 38
4.2.2 *Workflow* 2 40
4.2.3 *Workflow* 3 44
4.2.4 Comparação entre os NoSQLs 49

5 Conclusão e Trabalhos Futuros **53**

Referências **55**

Lista de Figuras

2.1 Fases de um *workflow* de Bioinformática - filtragem, mapeamento/montagem e análise/anotação. 5
2.2 Exemplo do processo de mapeamento. 6
2.3 Exemplo do processo de montagem. 6
2.4 Exemplo de um arquivo no formato FASTA. 9
2.5 Exemplo de um arquivo no formato FASTQ. 9
2.6 Exemplo de um arquivo no formato SAM. 10
2.7 Exemplo de um arquivo no formato BED. 10
2.8 Possibilidades do Teorema CAP. Adaptado de [Han et al., 2011]. 13
2.9 SGBDs mais utilizados atualmente [DB-Engines, 2016c]. 15
2.10 Bancos de Dados NoSQLs baseado em colunas mais utilizados atualmente [DB-Engines, 2016b]. 15
2.11 Bancos de Dados NoSQLs orientado a documentos mais utilizados atualmente [DB-Engines, 2016a]. 16
2.12 Exemplo de coluna do Cassandra. 18
2.13 Estrutura da família de colunas do Cassandra. 18
2.14 Exemplo de família de colunas do Cassandra. 19
2.15 Estrutura do MongoDB. 20
2.16 Exemplo de uma coleção de documentos composto por documentos com campos distintos. 21
2.17 Ambiente com 4 máquinas e fator de replicação 3. 22
2.18 Estratégias de replicação. 24

3.1 *Workflow* 1, com a fase de mapeamento. 28
3.2 *Workflow* 2, com as fases de mapeamento / montagem e anotação. 29
3.3 *Workflow* 3 com as fases de filtragem, mapeamento / montagem e anotação. 30
3.4 Modelo de dados do Cassandra. 32
3.5 Entidades de um *sharded cluster*. Adaptado de [Harrison, 2015]. 33

4.1 Teste quente X Teste frio em cada NoSQL. 36

4.2 Inserção e extração dos arquivos de entrada FASTA em cada NoSQL (*Workflow* 1). 39
4.3 Inserção e extração do arquivo SAM da fase de mapeamento em cada NoSQL (*Workflow* 1). 40
4.4 Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 1). 40
4.5 Inserção e extração de todos os arquivos do *Workflow* 1 em cada NoSQL. . 41
4.6 Inserção e extração dos arquivos de entrada FASTQ em cada NoSQL (*Workflow* 2). 41
4.7 Inserção e extração dos arquivos BAM da fase de mapeamento em cada NoSQL (*Workflow* 2). 42
4.8 Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 2). 42
4.9 Inserção e extração dos arquivos FASTA da fase de montagem em cada NoSQL (*Workflow* 2). 43
4.10 Inserção e extração dos arquivos TXT da fase de anotação em cada NoSQL (*Workflow* 2). 43
4.11 Inserção e extração de todos arquivos do *workflow* 2 em cada NoSQL. . . . 44
4.12 Inserção e extração dos arquivos de entrada FASTQ em cada NoSQL (*Workflow* 3). 45
4.13 Inserção e extração dos arquivos FASTQ da fase de filtragem em cada NoSQL (*Workflow* 3). 46
4.14 Inserção e extração dos arquivos FASTA da fase de filtragem em cada NoSQL (*Workflow* 3). 46
4.15 Inserção e extração dos arquivos BAM da fase de mapeamento em cada NoSQL (*Workflow* 3). 47
4.16 Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 3). 47
4.17 Inserção e extração dos arquivos FASTA da fase de montagem em cada NoSQL (*Workflow* 3). 48
4.18 Inserção e extração dos arquivos TXT da fase de anotação em cada NoSQL (*Workflow* 3). 48
4.19 Inserção e extração de todos os arquivos em cada NoSQL (*Workflow* 3). . . 49
4.20 Comparação entre o tempo de execução, inserção e extração dos *workflows* 1, 2 e 3 no *cluster* com 2 máquinas e FR1. 51
4.21 Comparação entre o tempo de execução, inserção e extração dos *workflows* 1, 2 e 3 no *cluster* com 4 máquinas e FR3. 52

Lista de Tabelas

2.1 Ferramentas utilizadas na execução de *workflows* em Bioinformática. 7
2.2 Trabalhos Relacionados. 26

3.1 Informações dos *workflows*. 30

4.1 Inserção de todos arquivos no Cassandra. 37
4.2 Inserção de todos arquivos no MongoDB. 37
4.3 Extração de todos os arquivos no Cassandra. 37
4.4 Extração de todos os arquivos no MongoDB. 38
4.5 Relação percentual entre o tempo execução, inserção e extração dos arquivos de cada *workflow*. 50

Lista de Abreviaturas e Siglas

ACID Atomicidade, Consistência, Isolamento e Durabilidade.

API *Application Programming Interface.*

BAM *Binary SAM.*

BASE *Basically Available, Soft state, Eventualy consistent.*

BED *Browser Extensible Data.*

BSON *Binary JSON.*

BTOP *Blast trace-back operations.*

CAP *Consistency, Availability, Partition Tolerance.*

EHR *Electronic Health Record.*

JSON *JavaScript Object Notation.*

NoSQL *Not Only SQL.*

QPNs *Queueing Petri Nets.*

SGWfC Sistemas Gerenciadores de *Workflows* Científicos.

SQL *Structured Query Language.*

UCSC *University of California Santa Cruz.*

XML *eXtensible Markup Language.*

YCSB *Yahoo! Cloud Serving Benchmark.*

Capítulo 1

Introdução

Workflows científicos são compostos por uma série de programas de forma que cada programa possua um conjunto de parâmetros e dados de entradas, combinados para resolver um problema específico [Mattoso et al., 2014]. Esses *workflows* podem ser gerenciados através de Sistemas Gerenciadores de *Workflows* Científicos (SGWfC). Os SGWfC auxiliam os cientistas na descrição e gerenciamento das análises de processos, além de permitir a criação e reutilização de tarefas. Outro benefício desses sistemas para os cientistas é auxiliar no gerenciamento dos dados utilizados e gerados a cada passo, e sistematicamente gravar as informações de proveniência para mais tarde serem usadas [Davidson and Freire, 2008].

Os projetos realizados na Bioinformática podem ser executados como *workflows* científicos, classificados como projetos genômicos ou transcritômicos, dependendo do tipo de sequência utilizada. Os biólogos podem executar o mesmo *workflow* científico de Bioinformática diversas vezes com diferentes parâmetros, a fim de comparar os resultados obtidos em cada execução e refinar as análises dos dados.

Cada execução de um *workflow* científico de Bioinformática pode gerar um grande volume de dados, o qual deve ser armazenado para futuras execuções ou análises dos resultados. Dentre diversos sistemas de armazenamento de dados apresentados na literatura, novos modelos de bancos de dados têm sido definidos, os bancos de dados não relacionais, também conhecidos como bancos de dados *Not Only SQL* (NoSQL) [Cattell, 2011].

Os sistemas gerenciadores de bancos de dados NoSQL surgiram da necessidade de uma alta escalabilidade e desempenho quando o volume de dados é muito grande, sendo aplicados em armazenamento distribuído de dados e projetados para suportar o acesso a dados em grande escala [Stonebraker, 2010]. Dentre as principais características dos NoSQL estão: a desnormalização das informações, a capacidade de serem escalados horizontalmente, esquema flexível, baixo tempo de resposta e grande disponibilidade de acesso

[Mohamed et al., 2014]. Além disso, oferecem mecanismos de replicação de dados [Osman and Piazzolla, 2014], o qual garante alta disponibilidade dos dados e maior escalabilidade.

De acordo com [Corbellini et al., 2017], os NoSQL são classificados em três categorias: bancos de dados baseados em documentos, chave/valor e baseado em colunas. No entanto, outros autores [Pourabbas, 2014], [Pankhudi Swaroop, 2016], incluem uma quarta categoria nessa classificação, os bancos de dados baseados em grafos.

Nesse contexto, foi analisada a replicação de dados de *workflows* de Bioinformática em bancos de dados NoSQL em um ambiente distribuído composto por máquinas com poucos recursos computacionais. Nesse tipo de ambiente distribuído, é comum o surgimento de falhas, e por isso é importante considerar a replicação dos dados para que o ambiente consiga responder mesmo com possíveis falhas.

Os bancos de dados NoSQL utilizados nesta dissertação foram o Cassandra e o MongoDB, pois segundo [Holt et al., 2015], [Guo et al., 2016] e [DB-Engines, 2016c], estes NoSQL estão entre os sistemas gerenciadores de bancos de dados mais utilizados no mundo. Logo, é analisado o tempo de escrita e leitura de diferentes formatos de dados biológicos em cada NoSQL, quando se utiliza replicação dos dados. Essa avaliação é feita considerando o número de nós (máquinas) do *cluster* e o fator de replicação.

1.1 Objetivo

O objetivo geral desta dissertação é analisar o impacto da replicação de dados de *workflows* de Bioinformática, usando projetos transcritômicos compostos por arquivos com diferentes formatos, quando se utiliza bancos de dados NoSQL em um ambiente computacional distribuído.

Para o desenvolvimento do objetivo geral, foram definidos os seguintes objetivos específicos:

- Especificação do ambiente distribuído que suporte replicação de dados;
- Realizar as operações de escrita e leitura nos bancos de dados NoSQL utilizando um ambiente computacional distribuído;
- Definir os processos de leitura e escrita em diferentes famílias de NoSQL.
- Analisar e comparar os resultados obtidos.

1.2 Estrutura do Documento

Este documento está estruturado da seguinte maneira:

- O Capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento dessa pesquisa, tais como os conceitos de *workflow* de Bioinformática, bancos de dados NoSQL, replicação de dados e alguns trabalhos relacionados ao projeto proposto;
- O Capítulo 3 apresenta o ambiente computacional distribuído utilizado, as informações de cada *workflow* executado e como foi realizado a preparação de cada NoSQL utilizado;
- O Capítulo 4 apresenta os resultados obtidos (tempos de inserção e extração dos arquivos) e a análise dos resultados para cada *workflow* e seus respectivos formatos de arquivos;
- O Capítulo 5 apresenta a conclusão do trabalho e os trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo são apresentados os conceitos fundamentais para o desenvolvimento desde trabalho, sendo dividido nas seguintes seções: a Seção 2.1 trata sobre os conceitos de *workflows* científicos de Bioinformática, os programas utilizados nas execuções de cada fase dos *workflows* e os principais formatos de dados; a Seção 2.2 apresenta os conceitos relacionados aos bancos de dados NoSQL, destacando os dois sistemas gerenciadores de bancos de dados NoSQL utilizados nesta dissertação, o Cassandra e o MongoDB; a Seção 2.3 apresenta os principais conceitos da replicação de dados; e por fim, a Seção 2.4 mostra um resumo dos trabalhos relacionados a esta dissertação.

2.1 *Workflows* em Bioinformática

Workflow científico é um conjunto de tarefas a serem executadas para alcançar determinado objetivo [Mattoso et al., 2014]. É composto de um conjunto de dados de entrada e saída, fases e ferramentas, os quais são ordenados na melhor forma de atingir o objetivo almejado [de Paula et al., 2013].

Experimentos da Bioinformática geralmente são modelados como *workflows* científicos. *Workflows* científicos em Bioinformática são relacionados ao processamento de dados biológicos, especialmente com sequenciamento de DNA e/ou RNA. Esses *workflows*, quando usados em projetos de genomas ou transcritomas, podem ser compostos de diversas fases [Leipzig, 2016]. Os *workflows* utilizados nesta dissertação possuem três fases [de Paula et al., 2013], [Huacarpuma et al., 2011]: filtragem, mapeamento/montagem e análise/anotação, como pode ser visto na Figura 2.1.

A fase de filtragem utiliza como entrada as amostras de DNA ou RNA, chamadas *reads*, geradas por sequenciadores de alto desempenho [Huacarpuma et al., 2011]. Alguns exemplos de formato de saída dessa fase são FASTA e FASTQ. O objetivo geral da fase de filtragem é a remoção de sequências que possuem baixa qualidade, as quais podem

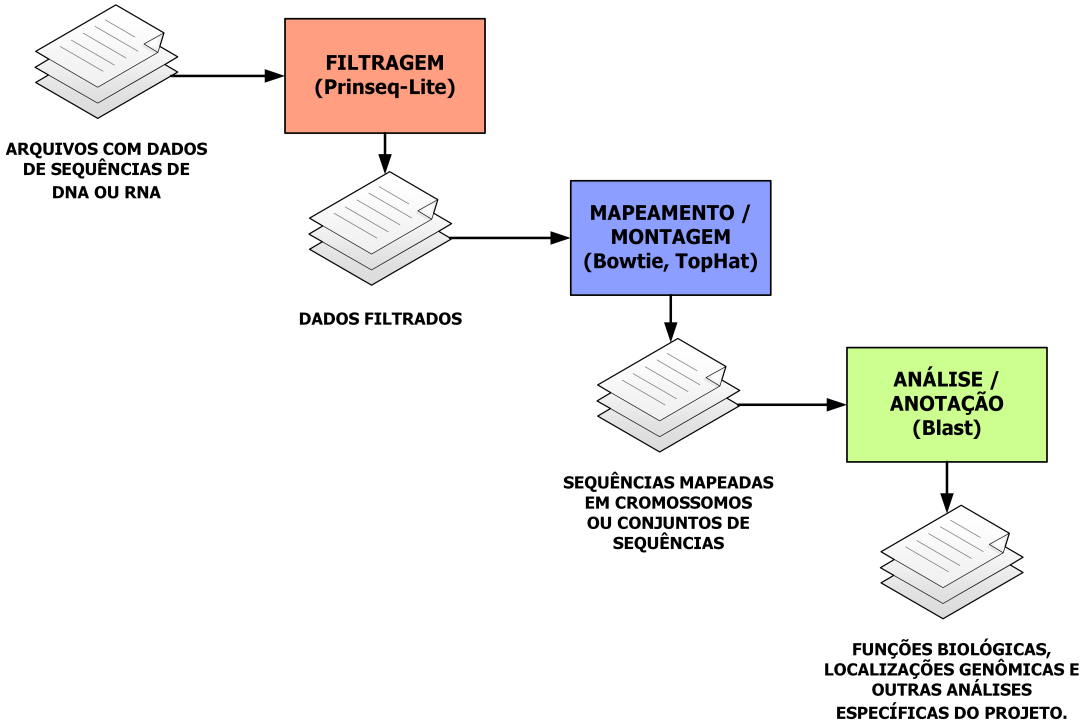

Figura 2.1: Fases de um *workflow* de Bioinformática - filtragem, mapeamento/montagem e análise/anotação.

dificultar ou afetar de forma negativa os resultados. Exemplos de ferramentas utilizadas nessa fase são o pacote FASTX-Toolkit [Gordon and Hannon, 2010] e a ferramenta Prinseq [Schmieder and Edwards, 2011].

Após a fase de filtragem, inicia-se o mapeamento/montagem. O mapeamento é a fase na qual as *reads* filtradas e padronizadas na fase anterior são mapeadas em um genoma de referência. O objetivo geral dessa fase é fazer o alinhamento de *reads* utilizando o genoma de referência e agrupá-las para serem analisadas na próxima fase [Pinheiro et al., 2014]. Quando não se tem um genoma de referência, é feito uma montagem *de novo* com o alinhamento das *reads* para gerar sequências maiores, as quais são chamadas de *contigs*. Alguns exemplos de programas utilizados nessa fase são o TopHat [Trapnell et al., 2009], Bowtie [Langmead et al., 2009b], Segemehl [Hoffmann et al., 2015] no mapeamento, e Trinity [Grabherr et al., 2011] na montagem *de novo*. A Figura 2.2 mostra um alinhamento de *reads*, que consiste em descobrir as suas posições em relação a um genoma de referência conhecido. Como pode ser observado, a última linha representa o genoma de referência e as *reads* apresentadas foram alinhadas nesse genoma. Na Figura 2.3, não há um genoma de referência, e portanto as *reads* são alinhadas gerando um *consenso*, ilustrada na última linha da figura. Como pode ser visto na Figura 2.3, o conjunto de *reads* e *consenso* é denominado *contig*.

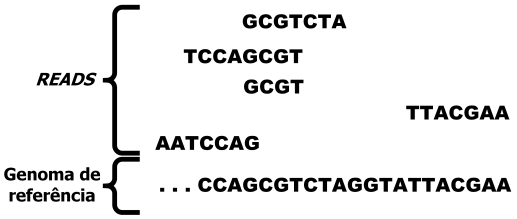

Figura 2.2: Exemplo do processo de mapeamento.

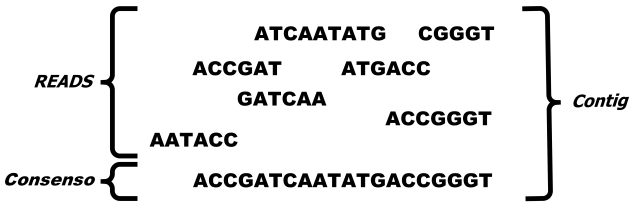

Figura 2.3: Exemplo do processo de montagem.

A última fase é a análise/anotação, em que as porções de DNA ou RNA mapeadas/montadas são analisadas de acordo com o objetivo do projeto que está sendo executado. A fase de análise é um processo com a finalidade de obter informações biológicas. Alguns programas utilizados nesta fase são BLAST [Camacho et al., 2013] e o R [Team, 2014].

2.1.1 Programas Utilizados nos *Workflows* em Bioinformática

Um dos desafios para a elaboração dessa dissertação foi a definição de quais programas analisar, uma vez que a Bioinformática possui diversas ferramentas que podem ser utilizadas para a execução de seus *workflows* científicos. Para a escolha dessas ferramentas, usou-se como base os artigos [Mossucca et al., 2012], [Fonseca et al., 2012] e [Haas et al., 2013] que descrevem algumas das aplicações mais utilizadas nas execuções dos *workflows* científicos de Bioinformática.

A Tabela 2.1 apresenta as ferramentas utilizadas neste trabalho, assim como o formato de dados de entrada e saída de cada uma. Como pode ser observado, os formatos de entrada FASTA e FASTQ são utilizados por todas as ferramentas analisadas. O formato SAM é utilizado como saída do Bowtie e o BAM ou BED como saída do TopHat.

Cada uma dessas ferramentas são descritas a seguir.

Prinseq

Prinseq [Schmieder and Edwards, 2011] é uma ferramenta que gera estatísticas de sequências e qualidade de dados. É utilizada para filtrar, formatar e remover dados de sequências

Tabela 2.1: Ferramentas utilizadas na execução de *workflows* em Bioinformática.

Ferramenta	**Fase**	**Entrada**	**Saída**
Prinseq [Schmieder and Edwards, 2011]	Filtragem	FASTA ou FASTQ	FASTA ou FASTQ
Trinity [Grabherr et al., 2011]	Montagem	FASTA ou FASTQ	FASTA
TopHat [Trapnell et al., 2009]	Mapeamento	FASTA ou FASTQ	BAM ou BED
Bowtie [Langmead et al., 2009b]	Mapeamento	FASTA ou FASTQ	SAM
Blast [Camacho et al., 2013]	Anotação	FASTA	Personalizado

biológicas.

Essa ferramenta é usada na fase de filtragem e tem como entrada e saída de dados arquivos tanto no formato FASTA ou FASTQ.

Trinity

Trinity [Grabherr et al., 2011] é um método eficiente e robusto para reconstrução de transcritoma *de novo*, usado para o processamento de grandes volumes de *reads* de RNA-Seq.

A ferramenta Trinity é utilizada na fase de montagem do *workflow*, tendo como entrada arquivos no formato FASTQ ou FASTA e como saída arquivos no formato FASTA.

TopHat

TopHat [Trapnell et al., 2009] é um programa de alinhamento de *reads* geradas por RNA-Seq que utiliza internamente o mapeamento de *short reads* do Bowtie [Langmead et al., 2009b], descrito na próxima seção.

O TopHat é utilizado na fase de mapeamento e tem como entrada de dados arquivos no formato FASTQ ou FASTA. O TopHat gera quatro arquivos de saída, sendo um no formato *Binary SAM* (BAM) e os demais no formato *Browser Extensible Data* (BED).

Bowtie

Bowtie [Langmead et al., 2009b] é um programa para alinhar conjuntos de *reads*, sendo usado na fase de mapeamento. Ele constitui a base de uma série de outras ferramentas como o TopHat [Trapnell et al., 2009], Cufflinks [Trapnell et al., 2012], Crossbow [Langmead et al., 2009a] e Myrna [Langmead et al., 2010]. O Cufflinks é uma ferramenta que faz a montagem do transcritoma. O Crossbow é uma ferramenta de computação em nuvem para re-sequenciamento de grande volume de dados. Myrna também é uma ferramenta de computação em nuvem que realiza o cálculo de expressão gênica diferencial de grandes conjuntos de dados de RNA-Seq [Langmead et al., 2009b].

O Bowtie aceita como entrada de dados arquivos no formato FASTQ e FASTA. Como saída padrão, o Bowtie gera um alinhamento por linha e cada linha possui oito campos separados por tabulações. O formato padrão de saída desse programa é um arquivo SAM.

Blast

É a ferramenta de busca de similaridade utilizados em projetos de Bioinformática, que possui diversas funcionalidades, as quais são divididas pelo tipo de dado na sequência de consulta e nas sequências do banco de dados [Camacho et al., 2013]. Alguns exemplos são:

- Blastp: pesquisa um banco de dados de proteínas usando como consulta uma proteína;
- Blastx: pesquisa um banco de dados de proteínas usando como consulta um nucleotídeo traduzido;
- tblastn: pesquisa um banco de dados de nucleotídeos traduzidos usando como consulta uma proteína;
- Blastn: pesquisa um banco de dados de nucleotídeos usando como consulta um nucleotídeo.

O Blast é usado na fase de análise/anotação e aceita como entrada alguns formatos de dados, como FASTA, identificadores de sequências ou apenas as linhas de dados da sequência. Já os formatos de saída dessa ferramenta podem ser personalizados de acordo com a preferência do usuário. Alguns exemplos de formatos de saída são: tabular, TXT e *Blast trace-back operations* (BTOP).

2.1.2 Formatos de Arquivos

Como apresentado na seção anterior, existem diversos programas que podem ser usados em cada fase do processo de sequenciamento, assim como também existem diferentes formatos de arquivos envolvidos. O tipo de arquivo depende do programa utilizado e da necessidade de cada fase. A seguir são discutidas algumas questões relacionadas aos principais tipos de arquivos utilizados nesse trabalho.

Os arquivos FASTA possuem um formato de texto simples e por isso é a forma mais comum para representar sequências biológicas [Lipman and Pearson, 1985]. No geral, o arquivo FASTA contém um conjunto de cabeçalhos seguidos pelo texto da sequência. Cada cabeçalho é definido por uma linha que se inicia com o caractere “>”. Pelo padrão, linhas que iniciam com o caractere “;” são consideradas comentários, mas algumas aplicações utilizam o caractere “#” para essa função.

Na Figura 2.4 é mostrado uma parte de um arquivo no formato FASTA. Nesse exemplo, tem-se três sequências, identificadas por SRR836437.1, SRR836437.2 e SRR836437.3. Após cada identificador, tem-se um comentário sobre a sequência, incluindo o tamanho dela. Como pode ser observado, por exemplo, a sequência SRR836437.1 tem o comprimento (*length*) 49.

```
>SRR836437.1 FCD0W37ACXX:2:1101:1477:1912 length=49
NAAAAAACACCACCACAAAAATCACAATGGCCTCCTCTCGTGTCTTTGC
>SRR836437.2 FCD0W37ACXX:2:1101:1605:1916 length=49
NAATCCTTCATGTTGTGTCTCAGCGCCTTATAGGGCCTTCTGTCGGTCT
>SRR836437.3 FCD0W37ACXX:2:1101:2974:1958 length=49
NACAAGTACAGAGGGGCTACTGTTGAAGATCTGGATCCTCCTCCTGCAC
```

Figura 2.4: Exemplo de um arquivo no formato FASTA.

O arquivo FASTA pode vir acompanhado de outro arquivo com a extensão ".qual", chamado de formato QUAL neste trabalho, onde são informados os valores de qualidade de cada base que compõe a sequência.

O formato FASTQ [Cock et al., 2010] foi criado para reduzir o volume de dados utilizados pelo formato FASTA e QUAL. Nesses arquivos as qualidades das bases são codificadas com caracteres ASCII. Nesse formato, ilustrado na Figura 2.5, o cabeçalho é definido pelo caractere "@", seguido pelo identificador da sequência e comentários. Na linha seguinte tem-se a sequência e logo abaixo o caracter "+" que pode ser seguido pelo identificador da sequência novamente. Na última linha tem-se os valores de qualidade representados em ASCII.

```
@SRR836437.1 FCD0W37ACXX:2:1101:1477:1912 length=49
NAAAAAACACCACCACAAAAATCACAATGGCCTCCTCTCGTGTCTTTGC
+SRR836437.1 FCD0W37ACXX:2:1101:1477:1912 length=49
%4BDFFFFHHHHHIIIIIIIIIIIIIIIIIIIIHIIIIIIIIIIIIIIH
@SRR836437.2 FCD0W37ACXX:2:1101:1605:1916 length=49
NAATCCTTCATGTTGTGTCTCAGCGCCTTATAGGGCCTTCTGTCGGTCT
+SRR836437.2 FCD0W37ACXX:2:1101:1605:1916 length=49
%1=DDDEFHGHHHIIIIIIIIIIIIIIIIIGHIIIIIIIIIIIIIIIHII
```

Figura 2.5: Exemplo de um arquivo no formato FASTQ.

O formato SAM [Li et al., 2009], é um arquivo de texto criado para representar o resultado do alinhamento das *reads* em um genoma de referência. A versão binária do formato SAM é chamada de BAM, e contém exatamente as mesmas informações, mas em formato binário para economizar espaço em disco e na memória. Tanto o formato SAM quanto o BAM, armazenam os dados das *reads* originais e dos alinhamentos feitos.

As primeiras linhas do arquivo SAM são o cabeçalho. Depois do cabeçalho tem-se os alinhamentos propriamente ditos, um por linha, com campos separados por tabula-

ções. Um exemplo deste formato é mostrado na Figura 2.6. Como pode ser observado, a primeira linha apresenta a sequência seq_67_1 com suas respectivas informações de alinhamento.

```
seq_67_1      -    chr1    569647  CATCGTCACAATTCTAATTCTACTGACTATCTTAG     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      3:C>T,30:T>G
seq_43_1      -    chr1    150318171       CTGTAGGTCTGTACATTGTCAAGAACAATGAGATT     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      24:T>G
seq_166_1     -    chr1    568478  GTACTAATCTGCAACTCCTACATACTTCCCCCATT     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      24:T>G,33:G>T
seq_323_1     +    chr1    17692605        ATAGGTTAGGAGTTTGAGACCAGCCTGGCCAAAAT     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      1:G>T,32:C>A
seq_348_1     -    chr1    569656  AATTATAATTCTACTGACTATCCTAGAAATCGCTC     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      0:G>C,30:C>A
seq_110_1     +    chr1    115123825       GCAGGCAATGGTTAAAACTGGAATTTAAAGCTCTC     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0
seq_370_1     +    chr1    149856653       CTAAGGATACATTTTGAGGAAGTAGCTCCAACAAA     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      31:G>C,34:C>A
seq_375_1     -    chr1    569918  AACAATGATGGCGCGATGTAACACGAGAAGGCACA     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII     0      5:A>G,33:C>A
```

Figura 2.6: Exemplo de um arquivo no formato SAM.

O formato BED, ilustrado na Figura 2.7, foi criado pela *University of California Santa Cruz* (UCSC) [Kent et al., 2002] para descrever anotações em um genoma. É um formato texto, com colunas separadas por tabulações que representam intervalos no genoma associados à anotação e possui diversos campos que descrevem a representação gráfica da anotação pelo visualizador [Aken et al., 2016]. Além das colunas, um arquivo BED pode ter um cabeçalho que começa com a palavra *track* e descreve outros atributos utilizados na representação gráfica das regiões. Como pode ser visto na Figura 2.7, o cabeçalho do arquivo além da palavra *track*, contém os atributos *name* e *description*.

```
track name=deletions description="TopHat deletions"
I       13232   13233   -       1
I       14258   14259   -       1
I       14598   14599   -       1
I       44702   44703   -       3
I       46767   46768   -       1
I       49814   49815   -       1
I       136984  136985  -       1
```

Figura 2.7: Exemplo de um arquivo no formato BED.

2.2 Bancos de Dados NoSQL

O aumento do volume de dados no âmbito da Computação acarretou no surgimento de um novo conceito, *Big Data*. Esse conceito diz respeito a uma coleção de dados complexa e volumosa, a qual é difícil de manipular, gerenciar e realizar operações simples como remoção, deleção e inserção de dados [Vieira et al., 2012]. Para solucionar esse tipo de problema, surgiu então uma nova arquitetura de armazenamento de grandes volumes de dados, os chamados NoSQL [Stonebraker, 2010].

O termo NoSQL foi usado pela primeira vez em 1998 por Carlo Strozzi como o nome do banco de dados baseado em arquivo que ele estava desenvolvendo [Mohamed et al., 2014].

Desde aquela data, o termo tem sido utilizado para os bancos de dados não relacionais que omitem o uso de *Structured Query Language* (SQL) [Mohamed et al., 2014]. Sistemas de gerenciamento de banco de dados NoSQL são usados principalmente quando se lida com um grande volume de dados onde os atributos de dados são mais difíceis de ser expressos e modelados nos modelos relacionais tradicionais [Pankhudi Swaroop, 2016].

NoSQL são bancos de dados que permitem processamento de um grande volume de dados de maneira rápida e consistente, mantendo foco em performance. Possuem uma arquitetura distribuída tolerante a falhas, o que é de suma importância para evitar a perda de informações, uma vez que NoSQL é um modelo não estruturado que não segue os padrões dos bancos de dados relacionais [Mohamed et al., 2014], [Stonebraker, 2010].

Algumas características dos bancos de dados NoSQL são [Leavitt, 2010]: desnormalização das informações e alto desempenho no armazenamento paralelo de dados em grande escala. Além disso, eles apresentam a capacidade da escalabilidade horizontal, possui esquema flexível, baixo tempo de resposta e grande disponibilidade de acesso.

Quanto ao tipo de controle de consistência, os bancos de dados NoSQL implementam o modelo conhecido como *Basically Available, Soft state, Eventualy consistent* (BASE), que se opõe ao Atomicidade, Consistência, Isolamento e Durabilidade (ACID) no modelo relacional, e tolera inconsistências temporárias para priorizar a disponibilidade dos dados [Brewer, 2012]. Seus itens são descritos da seguinte forma [de Souza and dos Santos, 2015], [Swaminathan, 2015]:

- Basicamente disponível: os bancos de dados NoSQL apresentam disponibilidade de dados até na presença de falhas. Isso significa que o sistema deve estar disponível na maior parte do tempo, mas subsistemas podem estar temporariamente inoperantes;

- Estado leve: os bancos de dados que implementam o modelo BASE abandonam requerimentos de consistência que o modelo ACID implementa. Um dos conceitos básicos por trás do modelo BASE é a consistência dos dados, e o estado leve define que os desenvolvedores de bancos de dados devem criar mecanismos para essa consistência;

- Consistência eventual: a única exigência que os bancos de dados NoSQL possuem em relação à consistência é que os dados irão convergir para um estado consistente, porém não existe garantias ou um tempo definido para quando isso deve ocorrer. Isso é uma oposição à necessidade de consistência imediata que é requerida pelo ACID, que proíbe uma transação de acontecer até que o banco de dados tenha convergido para um estado consistente.

De maneira geral, os bancos de dados NoSQL utilizam os princípios de sistemas distribuídos definidos por [Brewer, 2000], conhecido como Teorema *Consistency, Availability, Partition Tolerance* (CAP). Este teorema é baseado nas seguintes propriedades:

- Consistência: um sistema é dito consistente se uma consulta ao banco de dados por parte de vários usuários resulta em uma mesma visão após uma atualização feita por um outro usuário que possui permissão para escrever no sistema;
- Disponibilidade: propriedade que garante a disponibilidade do serviço a qualquer momento, ou seja, qualquer usuário terá sua requisição atendida a todo momento, mesmo se uma máquina em um *cluster* estiver fora do ar;
- Tolerância à partição: propriedade em que um sistema pode executar corretamente, mantendo todas as suas características independente da divisão física da rede, de perda de dados ou de falhas no sistema. Um sistema com alta tolerância ao particionamento é um sistema altamente escalável, pois possui a habilidade de lidar com a adição e remoção dinâmica de nós e de recursos.

O Teorema CAP preconiza que, dentre as três propriedades, só é possível utilizar duas dentro de um sistema distribuído. A impossibilidade de usar mais de duas propriedades vem do fato de que, ao escolher duas delas, a outra é inviabilizada. Em outras palavras, pode-se dizer que, caso exista Consistência e Disponibilidade ao se dar liberdade para o usuário de ler e escrever a qualquer momento, sendo que todos os usuários estarão vendo uma versão consistente do sistema, não é possível garantir a propriedade de Particionamento, isto é a rede não poderá ter uma divisão física, pois a divisão física quebraria o princípio da Consistência [Brewer, 2012].

Seguindo o raciocínio, se há Disponibilidade e Tolerância à partição, a consistência não é alcançada, pois um usuário poderia escrever e ler a qualquer momento, porém ao encontrar um evento de partição seria necessário esperar até que todo o dado voltasse a ser consistente e durante esse tempo o serviço não estaria disponível.

Na última possibilidade, caso um sistema possua Consistência e Tolerância à partição, não é possível garantir para o usuário a escrita e leitura a todo momento. Ao se dar essa liberdade de leitura e escrita o sistema distribuído não tem como se manter estável para todos tenham acesso ao banco de dados.

A Figura 2.8 mostra quais são as possíveis relações entre as propriedades do Teorema CAP e as classifica em três categorias: Consistência e Disponibilidade (CA), Consistência e Tolerância à partição (CP) e Disponibilidade e Tolerância à partição (AP).

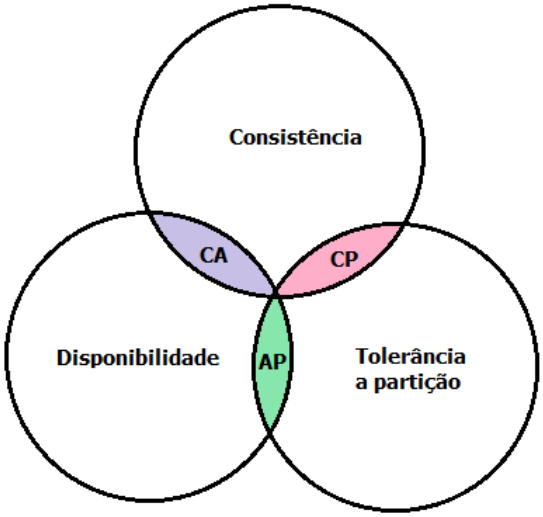

Figura 2.8: Possibilidades do Teorema CAP. Adaptado de [Han et al., 2011].

2.2.1 Modelos de Dados

Atualmente é possível encontrar muitos sistemas gerenciadores de bancos de dados NoSQL, com diferentes modelos de dados. De maneira geral, é possível classificar esses NoSQL de acordo com suas estruturas organizacionais [Pourabbas, 2014], [Pankhudi Swaroop, 2016], [Corbellini et al., 2017]:

- Modelo chave/valor: armazena os valores indexados por chaves. Nesse modelo de dados, os valores armazenados são divididos em duas partes: (a) chave, que é um identificador exclusivo para um dado de entrada particular. Não pode conter valores repetidos; (b) valor, que é um tipo de dado apontado por uma chave. Os dados são isolados e independentes entre si, e por esse motivo as relações devem ser tratadas pela lógica de aplicação. É um modelo que possui alta velocidade em consultas. Exemplo: Amazon Dynamo [DeCandia et al., 2007];

- Modelo orientado a documentos: armazena e organiza os dados como uma coleção de documentos. Estes documentos são codificados em um formato de dados padrão, como *eXtensible Markup Language* (XML), *JavaScript Object Notation* (JSON) ou *Binary JSON* (BSON). Os documentos são reconhecidos por um conjunto único de chaves e valores. Exemplos de modelos orientados a documentos são: CouchDB [Anderson et al., 2010] e MongoDB [Chodorow, 2013];

- Modelo baseado em colunas: é também conhecido como modelo tabular. Ele armazena os registros que podem ser particionados verticalmente e horizontalmente entre os nós. Nesse modelo, pode-se adicionar novas colunas de acordo com a necessidade

do usuário e estas são organizadas em famílias de colunas para facilitar a organização e particionamento do banco de dados. Exemplos: Cassandra [Lakshman and Malik, 2010], BigTable [Chang et al., 2008] e HBase [George, 2011];

- Modelo orientado a grafo: os objetos e os relacionamentos são armazenados em vértices (nós) e arestas de um grafo. Esse modelo é mais indicado para aplicações onde existe dados com muitas relações, devido à facilidade de realizar buscas com muitas conexões. Exemplo: Neo4j [Neubauer, 2010].

O primeiro passo para a definição dos sistemas gerenciadores de bancos de dados a serem analisados, foi a escolha do modelo de dados mais adequado para armazenar os dados de *workflows* científicos de Bioinformática. Nesse contexto foram selecionados os modelos baseado em documentos e baseado em colunas. Nesses modelos é possível armazenar os dados brutos e também os metadados, que são úteis para a proveniência de dados.

Dentre os diversos sistemas gerenciadores de bancos de dados NoSQL baseados em documentos e em colunas existentes, escolheu-se o MongoDB e o Cassandra, respectivamente, para o estudo de caso desta dissertação. A escolha desses bancos de dados foi baseada em uma pesquisa realizada, onde foi feita uma pesquisa no Google Acadêmico de trabalhos que utilizaram algum NoSQL, entre os anos de 2014 e 2016. Dentre os 40 primeiros trabalhos encontrados, 60% utilizaram o MongoDB e 38% utilizaram o Cassandra. Além disso, segundo [Holt et al., 2015] e [Guo et al., 2016], os bancos de dados NoSQL Cassandra e MongoDB estão entre os sistemas gerenciadores de bancos de dados mais utilizados atualmente.

A Figura 2.9, ilustra um gráfico representando a popularidade dos SGBD nos últimos anos. Como pode ser observado na figura, o MongoDB ocupa a 5ª posição dentre os SGBDs mais utilizados. O Cassandra ocupa o 7º lugar.

As Figuras 2.10 e 2.11 mostram os gráficos que representam quais bancos de dados são mais utilizados, de acordo com o seu modelo de dados. Como pode ser observado nas figuras, o Cassandra e o MongoDB são os NoSQL orientado a colunas e a documentos, respectivamente, mais populares.

2.2.2 Cassandra

O sistema gerenciador de bancos de dados NoSQL Cassandra surgiu em 2008, quando a equipe do Facebook desenvolveu uma arquitetura que combinava a abordagem do modelo de dados apresentada pelo sistema BigTable com a abordagem da infraestrutura do sistema Dynamo [Krishnan, 2013]. Em 2009, o Facebook doou o código do sistema à fundação Apache e após isso se tornou um projeto de fonte aberta [Lerner, 2010].

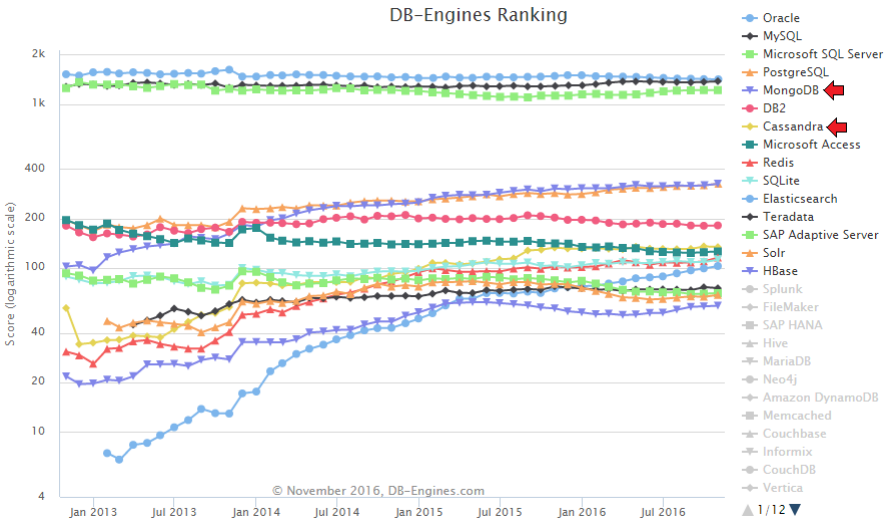

Figura 2.9: SGBDs mais utilizados atualmente [DB-Engines, 2016c].

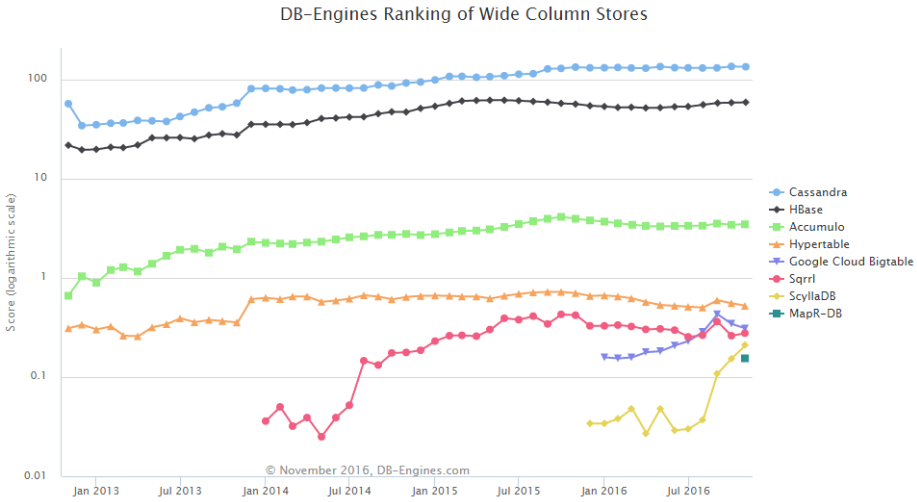

Figura 2.10: Bancos de Dados NoSQLs baseado em colunas mais utilizados atualmente [DB-Engines, 2016b].

Cassandra [Lakshman and Malik, 2010] é um sistema gerenciador de bancos de dados baseado no modelo orientado a colunas. Nele, os dados são identificados por uma chave. Por ser um sistema distribuído, o Cassandra é descentralizado, sendo capaz de executar em múltiplas máquinas, o qual aumenta o desempenho devido às operações executadas com programação paralela e melhora a segurança por causa da redundância dos dados. É escalável, e por isso seu alto desempenho é mantido mesmo se aumentar o número de requisições.

Dizer que o NoSQL Cassandra é um sistema capaz de prover alta disponibilidade e é

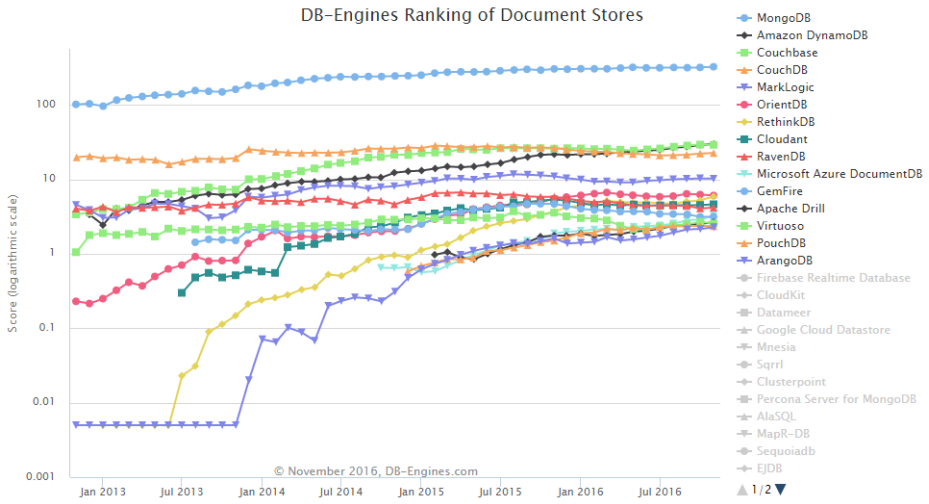

Figura 2.11: Bancos de Dados NoSQLs orientado a documentos mais utilizados atualmente [DB-Engines, 2016a].

tolerante a falhas significa que, mesmo que ocorra algum erro de hardware ou algum dado corrompido, o sistema possui a capacidade de completar as requisições. Isso é possível pois, quando apresenta falhas, o sistema faz o uso de redundância e realocação de recursos para dar continuidade ao procedimento [Lakshman and Malik, 2010].

No entanto, a disponibilidade do Cassandra pode ser reduzida quando o sistema garante a consistência dos dados. Porém, ele é um banco de dados de consistência ajustável, o que significa que o usuário tem permissão de decidir qual é o nível de consistência desejada para a aplicação [Lakshman and Malik, 2010].

As principais características do Cassandra são [Hewitt, 2010]:

- Distribuído: o Cassandra é capaz de executar em diversas máquinas, o que o torna distribuído, e por mais que ele esteja executando em diversas máquinas, para a visão usuário é como se ele estivesse executando em apenas uma;

- Descentralizado: o Cassandra não possui um único ponto de falha. Todos os nós do *cluster* funcionam exatamente do mesmo modo. A tal fato da-se o nome de simetria do servidor, ou seja, pode-se descrever como uma arquitetura *peer-to-peer* (uma topologia de anel, ponto-a-ponto);

- Escalável: escalabilidade é "uma característica arquitetural de um sistema que pode continuar servindo um grande número de requisição com uma pequena degradação na performance" (tradução livre) [Hewitt, 2010]. Desse modo, uma abordagem de escalabilidade vertical é simplesmente aumentar o poder computacional, melhorar o hardware. Já uma abordagem de escalabilidade horizontal significa aumentar o

número de máquinas e dividir os dados entre elas. A escalabilidade elástica refere-se a uma relação com a escalabilidade horizontal, no qual é possível aumentar e diminuir o número de máquinas dinamicamente. Dessa forma pode-se dizer que o *cluster* do Cassandra estará sempre apto a receber ou remover máquinas e fazer a distribuição das cópias dos dados entre elas para atender às novas requisições, sem que seja necessário reiniciar o sistema, processo, ou até mesmo editar algum comando;

- Alta disponibilidade e tolerância a falhas: a disponibilidade de um sistema é medida de acordo com sua capacidade de completar requisições. Porém todo e qualquer computador pode sofrer de algum tipo de falha, que envolvem desde falhas nos componentes de hardware, erros de dados corrompidos e até mesmo erros com a rede. O Cassandra busca reduzir as chances da ocorrência desses erros fazendo uso de redundâncias e realocação de recursos quando apresentam chances de falha;
- Consistência: significa que uma operação de leitura sempre retornará o valor do dado correspondente em sua versão mais recente. O Cassandra é um banco de dados de consistência ajustável, significando que é dado ao usuário a permissão de decidir qual é o nível de consistência desejada para a aplicação;
- Alta performance: o banco de dados Cassandra foi especificamente desenvolvido para tirar vantagem do uso de multiprocessamento, seja em máquinas que possuam *multicores* ou até mesmo em data-centers com diversas máquinas executando em conjunto.

Algumas empresas que utilizam o Cassandra são [Jain, 2017]: Apple, Twitter, Netflix, Facebook, Digg, Reddit e OpenX.

Modelo de dados do Cassandra

Para se entender os modelos de dados, deve-se primeiro entender duas estruturas básicas do Cassandra. A primeira estrutura básica são as colunas, um par chave-valor, similar ao modelo chave-valor, A Figura 2.12 exemplifica três pares chave-valor, no qual as chaves 1, 2 e 3 estão relacionadas com os valores 1, 2 e 3 respectivamente.

As colunas desse banco de dados possuem um nome, um valor e um *timestamp*, onde os dois últimos são fornecidos pela aplicação cliente no momento em que o dado é inserido na tabela. Além das colunas, existem também as chamadas super colunas, as quais possuem as mesmas características das anteriores, com a diferença de que, em vez de possuírem objetos como valores, possuem outras colunas. No entanto, em versões mais recentes do Cassandra, as super colunas foram extintas, pois nelas existiam algumas limitações como perda de desempenho em operações de leituras [Hewitt, 2010].

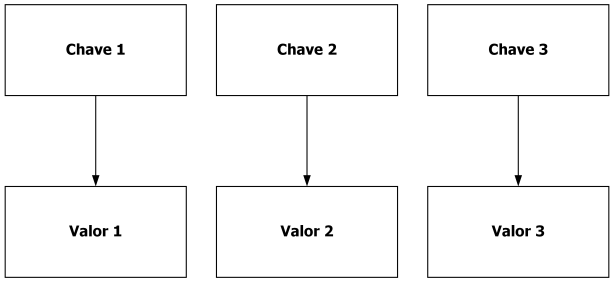

Figura 2.12: Exemplo de coluna do Cassandra.

A outra estrutura básica é a família de colunas, que é definida como uma divisão lógica que associam dados similares [Hewitt, 2010], em outras palavras, é um contêiner para linhas que possuem colunas similares, mas não necessariamente as mesmas.

Além das colunas e famílias de colunas, existe outro elemento que compõe o modelo de dados do Cassandra [Lakshman and Malik, 2010]: os *keyspaces*. Um *keyspace* é o agrupamento de dados que pode ser comparado a um esquema em um banco de dados relacional, porém seu maior diferencial é possuir informações sobre como os dados serão replicados ao longo de cada computador no *cluster*, tais como o fator de replicação e a estratégia de replicação. O *keyspace* possui a maior granularidade dentre os modelos de dados, sendo que uma família de colunas deve estar contida nele.

A Figura 2.13 representa a estrutura do Cassandra, onde tem-se um *keyspace* contendo duas famílias de colunas. Na figura, tomando como exemplo a *row key* 1, pode-se dizer que ela é a chave que representa a família de colunas representadas pelo contêiner com as colunas 1, 2 e 3.

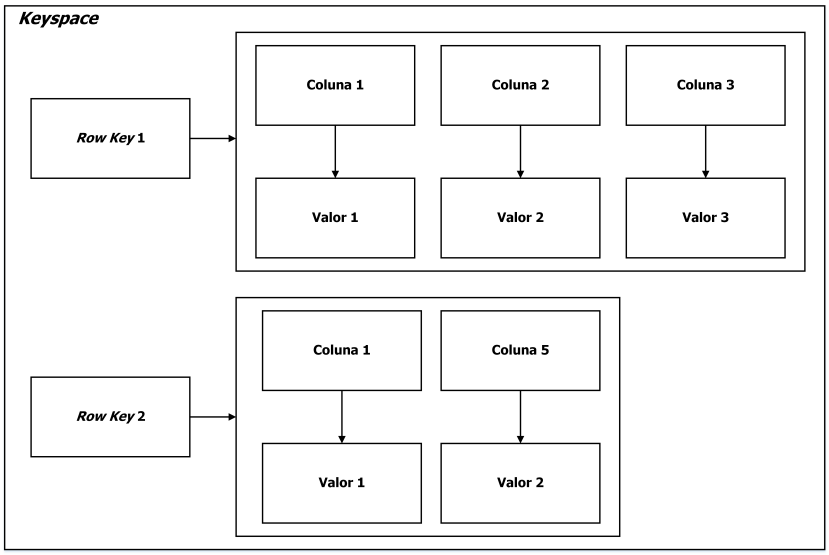

Figura 2.13: Estrutura da família de colunas do Cassandra.

A Figura 2.14 ilustra um exemplo de um *keyspace* no Cassandra, denominado *KeyspaceComercio*. Nele, existem duas famílias de colunas, identificadas pelos *id* 009 e 123, respectivamente. A família de colunas de *id* 009, refere-se a um restaurante e é composta por 4 colunas. A segunda família de colunas refere-se a uma farmácia e possui apenas 3 colunas.

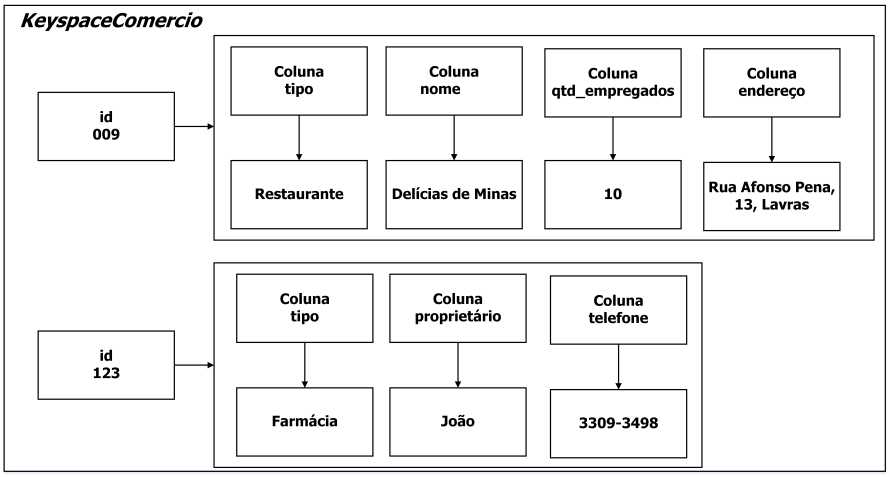

Figura 2.14: Exemplo de família de colunas do Cassandra.

2.2.3 MongoDB

MongoDB [Chodorow, 2013], é um sistema gerenciador de bancos de dados NoSQL com um mecanismo de consulta baseado em documentos. Foi criado em 2007 por 10gen (atualmente mantido pela MongoDB Inc.). O MongoDB é um banco de dados escalável, utiliza o formato JSON e foi desenvolvido em C++ como código aberto.

Esse sistema de banco de dados possui um foco em flexibilidade, facilidade de uso e velocidade de armazenamento. Além disso, possui suporte para servidores replicados e *drivers* para diversas linguagens de programação. A linguagem de consulta usada pelo MongoDB é baseada em chamadas de *Application Programming Interface* (API) e JavaScript, entre outras [Boicea et al., 2012].

O MongoDB possui outras duas características relevantes [Abramova and Bernardino, 2013]: a forma como gerencia o mecanismo de concorrência e os processos que utiliza para garantir a durabilidade dos dados. A durabilidade dos dados é garantida pela criação de réplicas. O NoSQL MongoDB utiliza um mecanismo de replicação mestre-escravo. Assim o utilizador define um nó mestre, e um ou vários nós escravos. O nó mestre pode ler ou escrever arquivos. Os nós escravos são limitados a servir como reservas, portanto, é

permitido que esses nós realizem apenas as operações de leitura. Se o nó mestre falhar, o nó escravo que possuir a réplica de dados mais recente é definido como o novo mestre.

Algumas empresas que utilizando o MongoDB são [Corbellini et al., 2017]: Squarespace, Forbes, New York Times.

Modelo de dados do MongoDB

O MongoDB utiliza uma estrutura onde tem-se uma base de dados composta por coleções de documentos, como é ilustrado na Figura 2.15. Os documentos possuem formato BSON, uma versão dos documentos JSON binário. A vantagem de se utilizar o formato BSON é a eficiência no espaço e no tempo de processamento ou computação, por ser um formato binário. Os documentos armazenados no MongoDB possuem tamanho máximo de 16 MB e não necessitam de ter o mesmo esquema [de Queiroz et al., 2013], [Guimaraes et al., 2015]. Cada documento armazenado é identificado por uma chave, a qual é uma sequência de caracteres e não pode ser duplicada. É possível armazenar e organizar os dados como uma coleção de documentos com campos e valores padrão para cada registro.

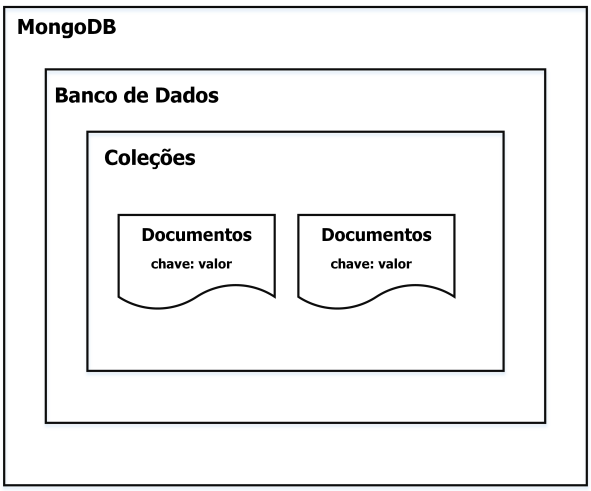

Figura 2.15: Estrutura do MongoDB.

A Figura 2.16, ilustra uma coleção de documentos, onde os documentos contém campos diferentes uns dos outros. Como pode ser visto, o primeiro documento refere-se a um tipo de restaurante com o nome "Delícias de Minas" e é identificado pelo _id 009. O segundo documento refere-se a uma farmácia do proprietário "João", e é identificada pelo _id 123.

Para armazenar aquivos maiores que 16 MB, o MongoDB fornece uma API chamada GridFS[8], que divide automaticamente os dados em pedaços menores de 256 KB e mantém metadados para todos os pedaços. Essa API utiliza duas coleções: uma contendo

[8]https://docs.mongodb.com/manual/core/gridfs/

```
{
        "_id" : "009",
        "tipo" : "restaurante",
        "nome": "Delícias de Minas",
qtd_empregados : "10"
    "endereço" : {
        "logradouro" : "Rua Afonso Pena",
            "numero" : 13
              "cidade" : "Lavras" }
},
{
        "_id" : "123",
        "tipo" : "farmácia",
    "proprietário" : "João",
        "telefone" : "3309-3498"
}
```

Figura 2.16: Exemplo de uma coleção de documentos composto por documentos com campos distintos.

os metadados dos arquivos armazenados e outra com os blocos de dados particionados [Pankhudi Swaroop, 2016].

O modelo usado pelo MongoDB para consistência dos dados é baseado no conceito de BASE, o qual provê transações distribuídas, replicação em sistemas distribuídos e tolerância a falhas relacionadas à consistência. No entanto, diferentemente de alguns bancos de dados relacionais, ele não oferece recursos como, controle de revisão e integridade referencial [Boicea et al., 2012].

2.3 Replicação de Dados

As operações de leitura e escrita de dados em disco ocupam muito do tempo de computação e transmissão de arquivos na rede e podem aumentar consideravelmente, dependendo do tamanho dos arquivos. Com o objetivo de reduzir este tempo, a replicação de dados é utilizada para conferir maior disponibilidade dos dados [da Silva et al., 2016].

A replicação de dados melhora o tempo de acesso aos dados e permite tolerância a falhas através da manutenção de cópias de dados (arquivos, objetos, bancos de dados ou partes de bancos de dados) em diferentes locais. Se as réplicas dos objetos estão bem organizadas, cada tarefa computacional pode acessar a réplica mais próxima do objeto necessário, reduzindo assim seu tempo de execução. Como resultado, o rendimento de todo o sistema pode ser aumentada [Loukopoulos and Ahmad, 2004].

A replicação de dados tem sido estudada em muitas áreas, especialmente em sistemas distribuídos (para fins de tolerância a falhas) e em bancos de dados (também por razões

tolerância a falhas e desempenho). A replicação nos bancos de dados NoSQL diz respeito ao armazenamento de várias cópias dos mesmos arquivos em diferentes nós, para garantir confiabilidade e disponibilidade do sistema [Wiesmann et al., 2000].

O controle do número de cópias de um dado é realizado pelo fator de replicação, o qual informa ao sistema gerenciador de bancos de dados a quantidade de cópias de determinado dado deve ser mantida disponível. Quanto maior o fator de replicação utilizado, maior a disponibilidade do dado. No entanto, a memória disponível no sistema de arquivos é reduzido proporcionalmente para este fator, devido à quantidade de cópias que cada dado acaba tendo [da Silva et al., 2016], [Loukopoulos and Ahmad, 2004].

A Figura 2.17 ilustra um ambiente composto de 4 máquinas e fator de replicação 3. Como é possível observar o arquivo file.txt está presente em três das quatro máquinas que compõe o ambiente distribuído.

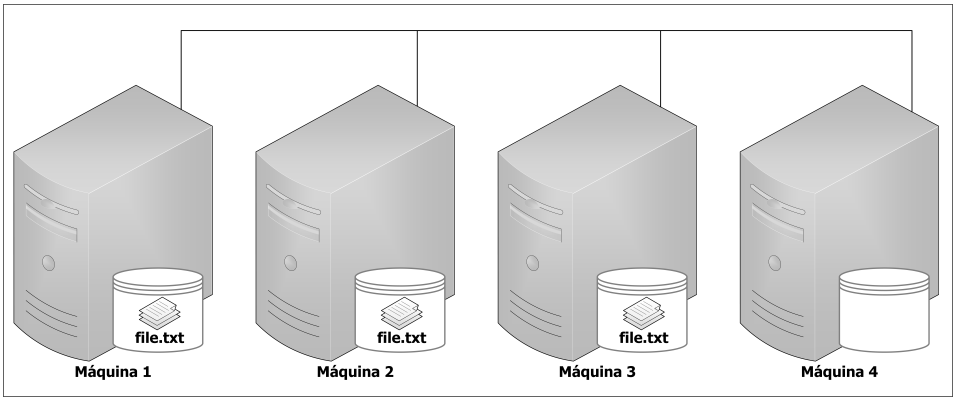

Figura 2.17: Ambiente com 4 máquinas e fator de replicação 3.

2.3.1 Tipos e Estratégias de Replicação de Dados

De acordo com a exigência de confiabilidade da aplicação, pode-se determinar o grau de replicação no banco de dados. Replicação de dados em um sistema de banco de dados distribuído pode ser classificada em três tipos [Elmasri, 2008]:

- Replicação completa: nesse tipo de replicação, todos arquivos armazenados são replicados em todos nós do sistema distribuído. Um banco de dados totalmente replicado irá fornecer disponibilidade em caso de falhas de nós ou partições de rede. No entanto, o desempenho do sistema em operações de escrita fica mais lento à medida que os arquivos são propagados para todas as cópias. Em sistema distribuído, a implementação de um banco de dados totalmente replicado tem como principal objetivo alcançar maior confiabilidade e disponibilidade dos dados;

- Replicação fragmentada: na replicação parcial, apenas fragmentos de dados necessários para a aplicação ou para o usuário é replicada;
- Sem replicação: um banco de dados distribuído, sem replicação, tem todos os dados armazenados exatamente em um local.

A replicação em sistemas de bancos de dados pode ser categorizada quando ocorre a propagação da atualização, denominadas replicação ansiosa (*eager raplication*) ou replicação preguiçosa (*lazy replication*) [Gray et al., 1996]. Em sistemas com replicação ansiosa as atulizações são propagadas dentro dos limites de uma transação, isto é, o usuário não recebe a notificação de confirmação até que cópias suficientes no sistema tenham sido atualizadas. Por outro lado, em sistemas com replicação preguiçosa, são feitas as atualizações, e somente após a confirmação, a propagação das alterações ocorre [Wiesmann et al., 2000].

Existem diversas opções de estratégias de replicação de dados, que devem ser escolhidas de acordo com o sistema a ser desenvolvido. Essas estratégias de replicação podem ser classificadas em centralizada e descentralizada [Swaminathan, 2015]:

- Replicação centralizada: é também conhecida como replicação mestre-escravo ou único mestre. Nesse tipo de replicação, as operações de escritas devem ser aplicadas inicialmente a apenas um local, chamado de local principal e posteriormente propagados às réplicas. A replicação centralizada é mais utilizada em sistemas que necessitam de processamento centralizado [Özsu and Valduriez, 2011], como os sistemas bancários. A grande desvantagem em ter um único mestre é que a disponibilidade do sistema pode ser afetada se esse mestre falhar;
- Replicação descentralizada: é também conhecida como mestre-mestre ou multi-mestre. Nesta, as operações de leitura e escrita são realizadas em uma cópia local do banco de dados e as alterações são então propagadas para outras máquinas. Os sistemas que implementam a replicação descentralizada são altamente disponíveis e têm recuperação rápida no caso de falha de um nó, no entanto possui baixa consistência.

Na Figura 2.18 são ilustradas as duas estratégias de replicação: mestre-escravo (replicação centralizada) e mestre-mestre (replicação descentralizada). Como pode ser observado, na Figura 2.18a o mestre é responsável pelo envio das réplicas de dados às máquinas escravos. Na Figura 2.18b os mestres dividem essa responsabilidade entre si.

Nos bancos de dados NoSQL, as estratégias de replicação de dados disponíveis variam de acordo com os requisitos necessários e com o modelo de dados escolhido. O Cassandra utiliza a estratégia de replicação descentralizada (*peer-to-peer*), pois cada um dos nós que

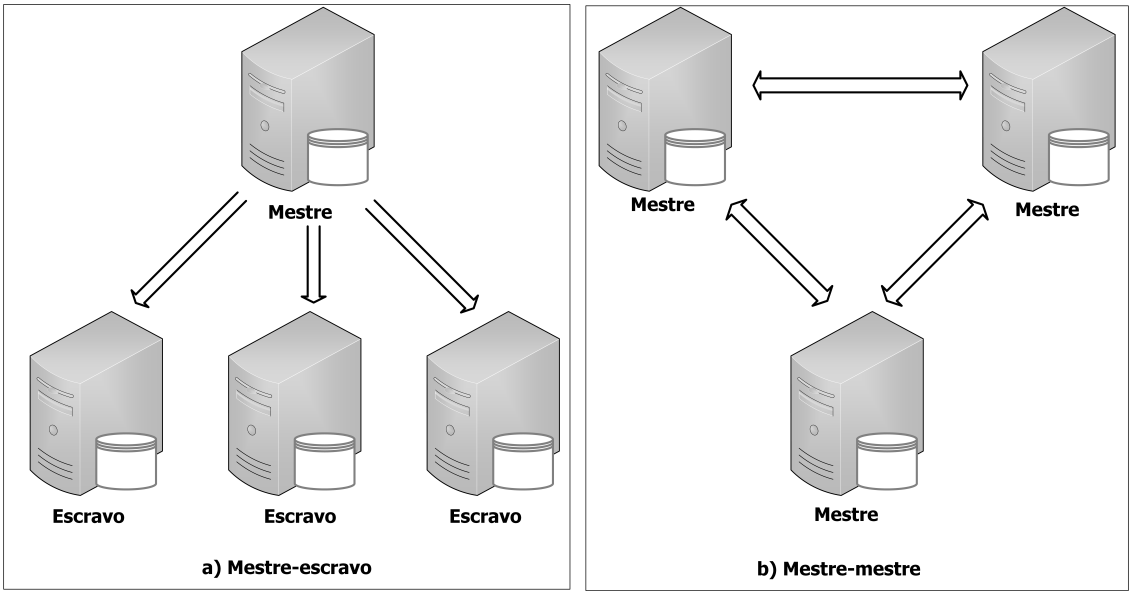

Figura 2.18: Estratégias de replicação.

compõe o *cluster* tem a mesma função. Quando algum nó é adicionado ou removido, todos os dados são distribuídos entre eles automaticamente. O fator de replicação é definido pelo usuário, que será o número total de réplicas. Por exemplo, o fator de replicação 1 significa que há apenas uma cópia de cada dado em um nó, enquanto o fator de replicação 2 representa que há duas cópias dos mesmos dados, cada um em nó diferente [Abramova and Bernardino, 2013].

O MongoDB utiliza uma variação da replicação centralizada, assim como os demais modelos baseados em documentos [Swaminathan, 2015]. Nele pode-se definir um mestre e um ou mais escravos. O mestre pode escrever ou ler os dados, enquanto no escravo somente operações de leitura são permitidas. Se o mestre falhar, o escravo com dados mais recentes é definido como o novo mestre [Abramova and Bernardino, 2013].

2.4 Trabalhos Relacionados

O armazenamento de dados biológicos não é um desafio recente, tendo em vista as dificuldades de gerenciamento de dados como relatado em [Bloom and Sharpe, 2004]. Em [Lifschitz, 2006] alguns tópicos de pesquisa no contexto de bancos de dados relacionais e da Bioinformática foram apresentados, destacando-se a necessidade de novas pesquisas para dar suporte aos projetos realizados com dados da Bioinformática.

Alguns trabalhos listados a seguir, realizaram estudos utilizando diversos sistemas de armazenamento para gerenciar dados biológicos. Em [Röhm and Blakeley, 2009] e [Huacarpuma et al., 2011] foram propostos modelos para armazenar dados genômicos

utilizando os sistemas de gerenciamento de banco de dados relacionais, SQL Server 2008 e PostgreSQL, respectivamente.

Em [Guimaraes et al., 2015] é realizado o armazenamento de dados de proveniência gerados durante a execução de um *workflow* científico em Bioinformática utilizando o sistema gerenciador de banco de dados NoSQL MongoDB. As operações de escrita e leitura dos dados foram realizadas com amostras de RNA do rim e fígado humanos, no formato FASTQ.

[Aniceto et al., 2015] realizaram um estudo sobre a persistência de dados genômicos e operações de entrada/saída no NoSQL Cassandra. Foram utilizados dados da fase de filtragem no formato FASTQ. Além disso, esses autores realizaram uma análise comparativa dos resultados com o banco de dados relacional PostgreSQL e NoSQL MongoDB.

[da Silva et al., 2016] propuseram uma política de replicação de dados, usando Hadoop, a fim de reduzir o tempo de processamento de dados biológicos. Assim como nesta dissertação, foram utilizados dados biológicos, no entanto não foi utilizado nenhum SGBD para o gerenciamento desses dados.

Outros artigos presentes na literatura, descrevem trabalhos que utilizaram a replicação de dados em bancos de dados NoSQL, com foco principalmente no desempenho, mas não utilizaram dados da área de Bioinformática. Em [Ye and Li, 2011], foram propostas melhorias para o banco de dados NoSQL Cassandra, a fim de garantir a consistência dos dados. Foi realizado também um balanceamento de armazenamento entre os nós dos *clusters* utilizados.

Em [Osman and Piazzolla, 2014] foi realizado um estudo sobre o desempenho do NoSQL Cassandra usando a plataforma de nuvem, Amazon EC2. Um modelo simples de *Queueing Petri Nets* (QPNs) foi usado para mostrar a escalabilidade do NoSQL, usando *clusters* de diferentes tamanhos e fatores de replicação.

Em [Klein et al., 2015], os bancos de dados MongoDB, Cassandra e Riak, foram avaliadas com foco no desempenho e escalabilidade de cada um. Para o estudo de caso, foram utilizados dados de um sistema *Electronic Health Record* (EHR), que fornece cuidados de saúde para mais de nove milhões de pacientes, em mais de 100 instalações em todo o mundo.

[Haughian et al., 2016] realizaram um estudo para avaliar o impacto no desempenho dos NoSQL Cassandra e MongoDB utilizando (ou não), a replicação de dados nas operações de leitura e escrita de diferentes cargas de trabalho sob diferentes níveis de consistência. Utilizaram o serviço *Yahoo! Cloud Serving Benchmark* (YCSB) como referência para realizar os experimentos em cada NoSQL.

Diferentemente dos trabalhos apresentados anteriormente, a contribuição desta dissertação é analisar o impacto da replicação de dados provenientes da execução de *workflows*

científicos de Bioinformática utilizando os sistemas gerenciadores de bancos de dados NoSQL Cassandra e MongoDB. Os trabalhos realizados anteriormente que, utilizaram dados biológicos, tiveram o foco em apenas alguns formatos de arquivos, especialmente o formato FASTQ. Outros focaram apenas da replicação de dados em bancos de dados NoSQL, utilizando outros formatos de arquivos. Nesta dissertação são consideradas as diferentes fases de *workflows* em Bioinformática, com diferentes formatos de arquivos. Além disso, foram considerados diferentes ambientes de computação distribuída, diversos tamanhos de *clusters* e mecanismos de replicação de dados.

Na Tabela 2.2 é apresentado um resumo de cada trabalho descrito anteriormente. Como pode ser observado, alguns trabalhos armazenam dados da Bioinformática em bancos de dados relacionais ou não relacionais, mas não utilizaram o mecanismo de replicação de dados. Além disso, esses trabalhos utilizaram poucos formatos de dados. Outros trabalhos, utilizam a replicação de dados, porém com arquivos de outras áreas. Como pode ser visto na tabela, esta dissertação faz o uso de dados da Bioinformática, utiliza o mecanismo de replicação nesses dados, além de analisar variados formatos de arquivos.

Tabela 2.2: Trabalhos Relacionados.

Autores	**Modelo BD**	**Bio**	**Replicação**	**Formato**
[Röhm and Blakeley, 2009]	Relacional	X		FASTQ
[Huacarpuma et al., 2011]	Relacional	X		FASTQ
[Guimaraes et al., 2015]	NoSQL (baseado em documentos)	X		FASTQ
[Aniceto et al., 2015]	NoSQL (baseado em colunas)	X		FASTQ
[da Silva et al., 2016]	-	X		FASTQ
[Ye and Li, 2011]	NoSQL (baseado em colunas)		X	Não informado
[Osman and Piazzolla, 2014]	NoSQL (baseado em colunas)		X	Não informado
[Klein et al., 2015]	NoSQL (baseado em colunas, documentos e chave/valor)		X	Não informado
[Haughian et al., 2016]	NoSQL (baseado em colunas e documentos)		X	Não informado
Esta dissertação	***NoSQL (baseado em colunas e documentos)***	***X***	***X***	***FASTQ, FASTA, BAM, BED, SAM, TXT***

Capítulo 3

Ambiente de Execução

Neste capítulo é apresentado o ambiente computacional utilizado para os testes realizados, os *workflows* que foram executados para esta dissertação, e como foi feita a preparação de cada NoSQL, sendo estruturado nas seguintes seções: na Seção 3.1 os três *workflows* executados e suas respectivas características são discutidos e na Seção 3.2 são apresentadas as aplicações desenvolvidas para cada NoSQL e os *clusters* utilizados.

3.1 *Workflows* Executados

Neste trabalho foi realizada uma avaliação para analisar impacto da replicação de dados nos bancos de dados Cassandra e MongoDB, ao inserir (*write*) e extrair (*read*) dados biológicos.

Os testes nos NoSQL e execuções dos *workflows* científicos de Bioinformática foram realizadas em um ambiente computacional distribuído com 7 máquinas, que utilizam o Sistema Operacional Linux (Ubuntu 14.04 64 bits). Cada máquina possui 8GB de RAM e 246,4 GB de disco, 4 CPUs e processador Intel ® Core™ i5.

Foram executados os seguintes *workflows*:

- *Workflow* 1: Identificação dos genes diferencialmente expressos em células cancerígenas humanas de rim e fígado [Marioni et al., 2008, Robinson and Oshlack, 2010].
- *Workflow* 2: Dados de experimentos de RNA-Seq do fungo *Schizosaccharomyces pombe* [Haas et al., 2013];
- *Workflow* 3: Dados de experimentos de RNA-Seq do fungo *Aspergillus fumigatus* [Latgé, 1999].

O *workflow* 1 é composto apenas pela fase de mapeamento. Como entrada de dados foi utilizado um arquivo no formato FASTA, e foram obtidos arquivos de saída no formato

SAM e BED, que somam 152,2 MB. As seguintes ferramentas foram usadas para executar o *workflow*: Bowtie [Langmead et al., 2009b], SAM2BED e genome2interval (*scripts* escritos em Perl), e coverageBed [Saldanha et al., 2012]. O tempo total para executar este *workflow* foi de 5 minutos e 31 segundos. A Figura 3.1 ilustra o *workflow* descrito. Como pode ser observado, os arquivos de entrada são *chr1.fa* e *s1.fa* e em azul é destacado sua única fase, o mapeamento.

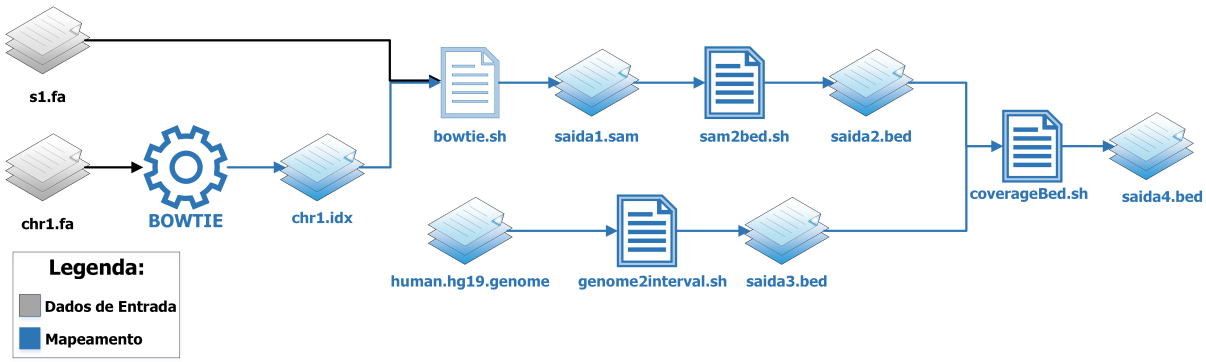

Figura 3.1: *Workflow* 1, com a fase de mapeamento.

O *workflow* 2, como ilustrado na Figura 3.2, é composto das fases: montagem / mapeamento e anotação. Os arquivos de entrada utilizados foram no formato FASTQ e como saída foram obtidos arquivos nos formatos FASTA, BAM, BED e TXT. Os arquivos obtidos em cada fase do *workflow*, somados aos arquivos de entrada, totalizam aproximadamente 2,39 GB de tamanho. O tempo total necessário para a execução completa foi de 16 horas, 50 minutos e 51 segundos. As ferramentas utilizadas foram: Trinity [Grabherr et al., 2011], Bowtie, TopHat [Trapnell et al., 2009] e Blast [Camacho et al., 2013]. Como pode ser visto na figura, os arquivos de entrada (representados pela cor cinza) foram agrupados, sendo que os arquivos *Sp.ds.left.fs*, *Sp.hs.left.fs*, *Sp.log.left.fs* e *Sp.plat.left.fs* foram concatenados em um único arquivo denominado *todas_reads_esquerda.fq*. Os arquivos *Sp.ds.right.fs*, *Sp.hs.right.fs*, *Sp.log.right.fs* e *Sp.plat.right.fs* também foram agrupados, originando o arquivo *todas_reads_direita.fq*. A fase de mapeamento é representada pela cor azul clara, a montagem pela cor vermelha e a anotação pela cor verde.

A execução do *workflow* 3 durou 23 horas, 28 minutos e 48 segundos. Como dados de entrada foram utilizados arquivos no formato FASTQ. Como saída, foram obtidos arquivos nos formatos FASTA, FASTQ, BAM, BED e TXT. Os arquivos de entrada e de saída somam aproximadamente 31,22 GB. Esse *workflow* possui as fases de filtragem, mapeamento/montagem e anotação. As ferramentas utilizadas para a execução foram Prinseq-Lite [Schmieder and Edwards, 2011], Bowtie, TopHat, Trinity e Blast. A Figura 3.3 ilustra o *workflow* 3. Nela, como pode ser observado tem-se 6 arquivos de entrada (cor cinza), os quais são utilizados na fase de filtragem, representada pela cor laranja.

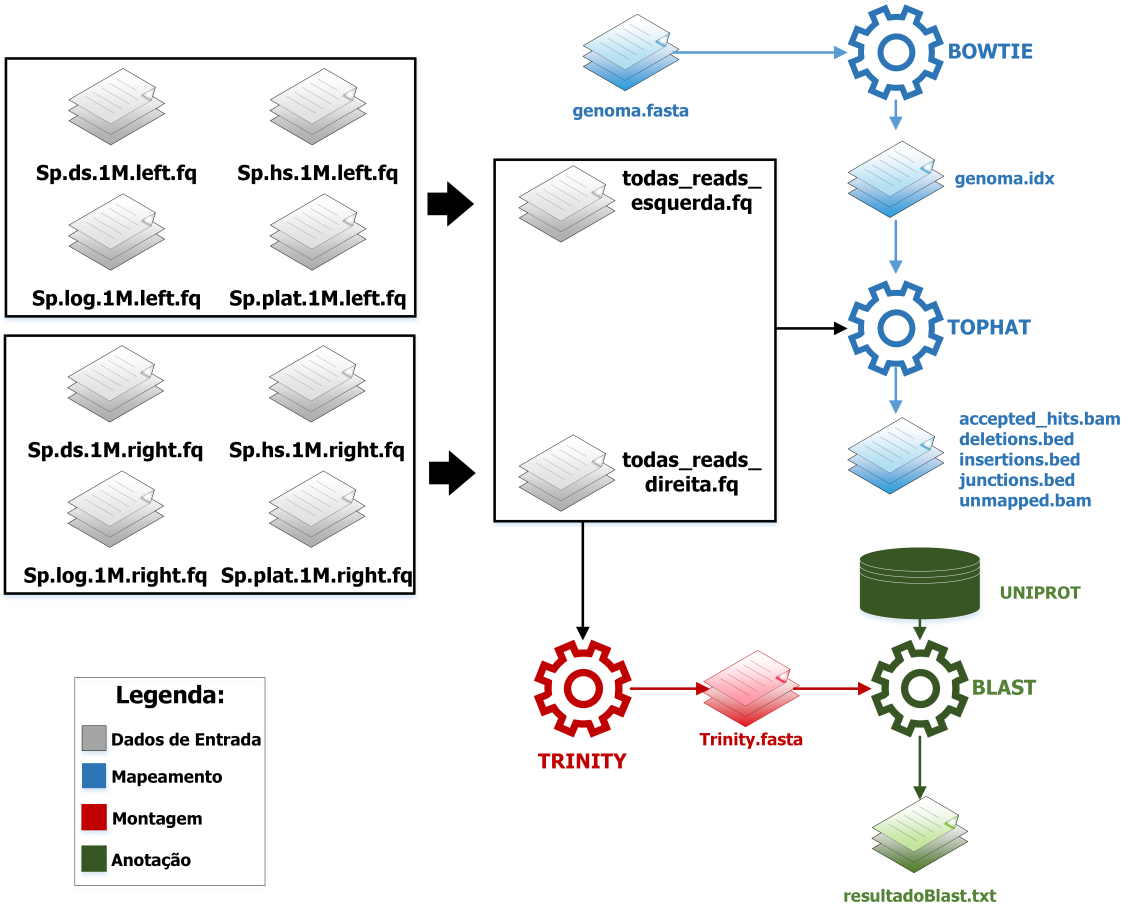

Figura 3.2: *Workflow* 2, com as fases de mapeamento / montagem e anotação.

Como nas figuras anteriores, o azul claro representa o mapeamento, vermelho representa a montagem e verde representa a anotação.

Após a execução dos *workflows*, foram selecionados quais arquivos deveriam ser salvos nos bancos de dados. A escolha desses arquivos foi tomada baseada em um pesquisa feitas com biólogos da área de Bioinformática. Os arquivos escolhidos permitem a reexecução do *workflow* a partir de qualquer fase, assim como também fazer análises dos arquivos.

A Tabela 3.1 mostra as informações de cada *workflow* descrito anteriormente, na qual TTA representa o tamanho total dos arquivos e TTE significa tempo de execução de cada *workflow*. Como pode ser observado na tabela, o *workflow* 1 possui apenas uma fase, o mapeamento, e apenas 2 formatos diferentes nos arquivos de saída desta fase. Além disso foi o *workflow* com menor tempo de execução, durando aproximadamente 5 minutos. O *workflow* 3 foi o mais demorado com diferentes formatos de arquivos.

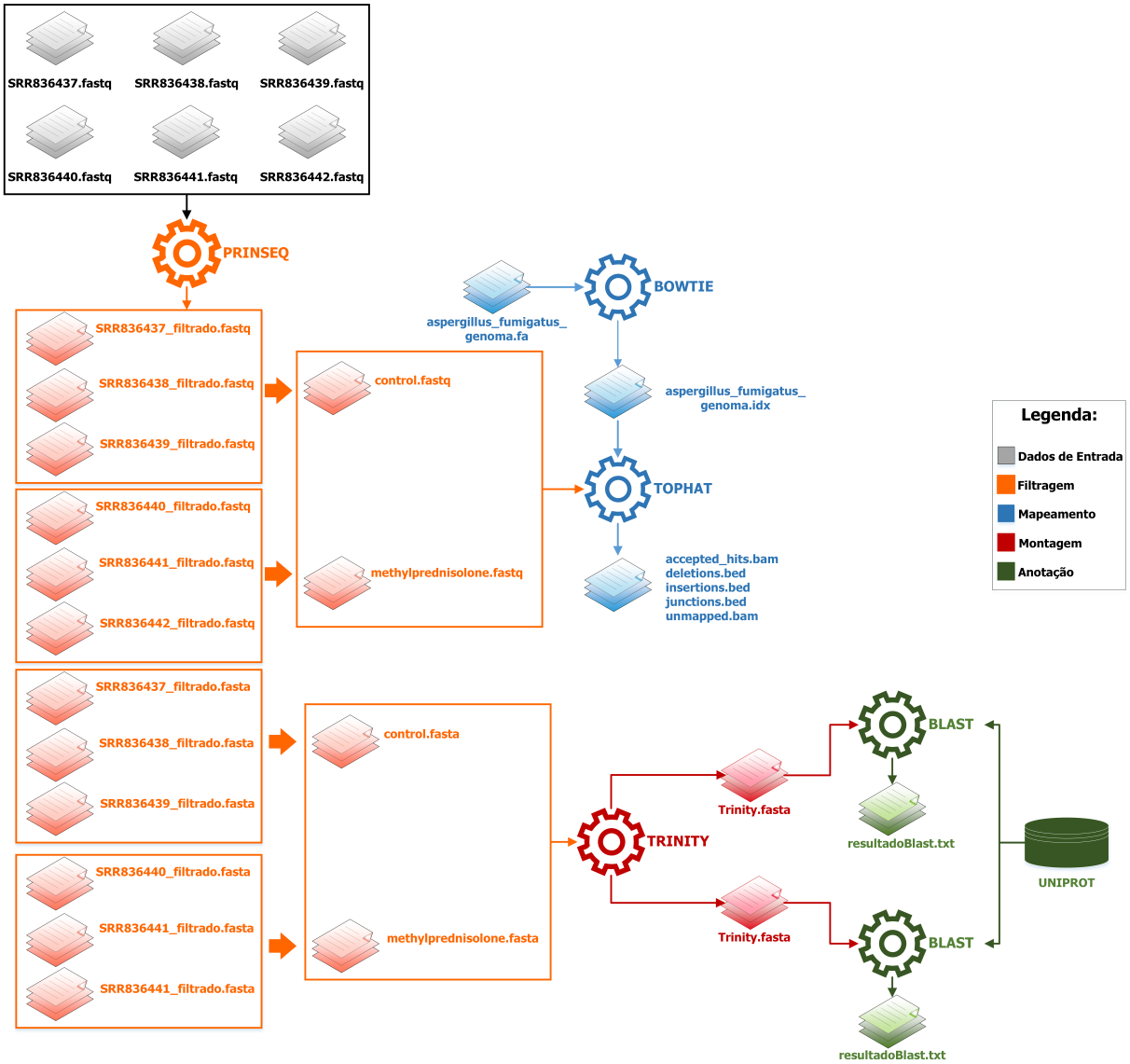

Figura 3.3: *Workflow* 3 com as fases de filtragem, mapeamento / montagem e anotação.

Tabela 3.1: Informações dos *workflows*.

Workflow	**Fases**	**Formatos dos Dados**		**TTA**	**TTE**
		Entrada	**Saída de cada fase**		
Workflow 1	Mapeamento	FASTA	BAM e BED	152,2 MB	5m 31s
Workflow 2	Montagem Mapeamento; Anotação	FASTQ	FASTA BAM e BED TXT	2,39 GB	16h 50m 51s
Workflow 3	Filtragem Montagem Mapeamento Anotação	FASTQ	FASTQ e FASTA FASTA BAM e BED TXT	31,22 GB	23h 28m 48s

3.2 Preparação dos Bancos de Dados

Dentre os diversos sistemas gerenciadores de bancos de dados NoSQL existentes, foram escolhidos dois para realização dos testes, o Cassandra e o MongoDB. Concomitantemente a execução dos *workflows* desenvolveu-se duas aplicações para realizar a importação e exportação dos dados em cada NoSQL. Estas aplicações foram desenvolvidas utilizando a linguagem Java, no ambiente de desenvolvimento Eclipse.

Em relação à replicação dos dados foi utilizado o tipo de replicação completa. Para cada NoSQL foi utilizada sua estratégia de replicação padrão: replicação centralizada no MongoDB e replicação descentralizada no Cassandra.

3.2.1 Preparação do Cassandra

Na aplicação desenvolvida para inserir e extrair os dados do Cassandra foi utilizado o Cassandra Java Driver[9], uma biblioteca disponibilizada pela Datastax[10]. A aplicação inicialmente tenta a conexão à uma instância do Cassandra. Após isso, ela cria um *keyspace* chamado *bioinformaticKeySpace* e duas tabelas, chamadas *files* e *meta*. A primeira tabela, *files*, armazena o arquivo, enquanto a tabela *meta* armazena os metadados do mesmo. Os metadados armazenados foram: o nome e a fase do *workflow*, o nome do arquivo, a data que ele foi inserido, o programa que o gerou, a versão do programa e o tamanho do arquivo. A Figura 3.4 mostra o modelo de dados utilizado, no qual dentro do *cluster* Cassandra está o *keyspace* criado. A *row key 1* identifica o arquivo e as colunas são todos os pedaços, em binário, dos arquivos. Cada coluna da tabela *files* possui um tamanho diferente, pois ela varia de acordo com o tamanho do arquivo.

Depois de criado o esquema do banco de dados, é feita a inserção dos arquivos na tabela *files*. Ao final da inserção de cada arquivo no Cassandra, a aplicação registra os seus metadados correspondentes na tabela *meta*, a fim de facilitar pesquisas, por exemplo, saber quantos arquivos existem registrados no banco de dados ou quais são os arquivos que pertencem a determinado *workflow*. Após a inserção, é feita a extração dos arquivos. Para garantir a integridade dos dados após a extração utilizou-se a ferramenta *diff*[11], nativa do sistema operacional Linux, que compara dois arquivos linha a linha e aponta se existe alguma diferença.

Existem duas estratégias de alocação das réplicas no Cassandra: estratégia de topologia em rede, que determina se elas serão distribuídas por uma rede de diferentes *clusters*, e estratégia simples, que determina se serão distribuídas em um único *cluster*. Foi escolhida a estratégia simples por ser mais adequada ao ambiente de testes utilizado.

[9] https://github.com/datastax/java-driver
[10] http://www.datastax.com/
[11] http://manpages.ubuntu.com/manpages/trusty/man1/diff.1.html

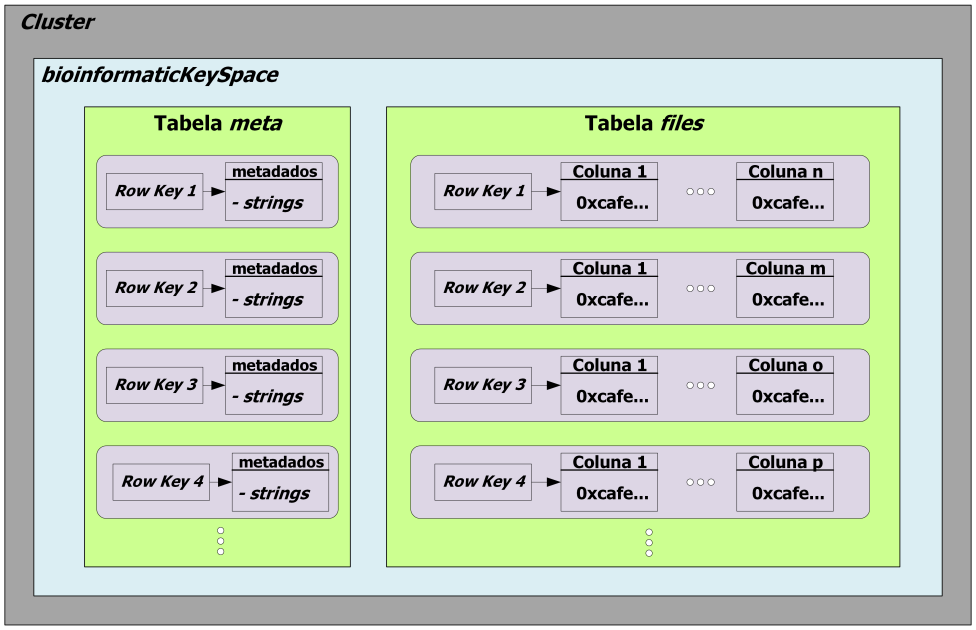

Figura 3.4: Modelo de dados do Cassandra.

A aplicação desenvolvida foi executada 10 vezes em cada *cluster*. Os ambientes computacionais distribuídos utilizados foram:

- Fator de replicação 1 em 3 *clusters* diferentes (2, 4 e 6 máquinas);
- Fator de replicação 2 em 3 *clusters* diferentes (2, 4 e 6 máquinas);
- Fator de replicação 3 em 1 *cluster* (4 máquinas).

3.2.2 Preparação do MongoDB

A aplicação desenvolvida para o MongoDB inicialmente faz a conexão com o banco de dados, insere todos os arquivos de cada *workflow* e em seguida faz a extração dos mesmos arquivos.

Os arquivos armazenamos são maiores que o tamanho máximo de um documento (16MB), portanto foi utilizada a API Java GridFS que quebra os arquivos em outros menores e os armazenam em duas coleções, *chunk* e *files*. A coleção *chunk* armazena os fragmentos dos arquivos quebrados pela API, enquanto a coleção *files* armazena os metadados de cada arquivo, como nome e fase do *workflow*, tamanho do arquivo, programa que gerou o arquivo e sua respectiva versão.

Um *cluster* montado para o MongoDB, é constituído dos seguintes componentes [Chodorow, 2013]:

- *Replica sets*: responsável pela replicação dos dados e prover a alta disponibilidade do banco de dados. É um conjunto de instâncias que mantém a mesma coleção de dados;

- *Shards*: responsável por garantir a escalabilidade horizontal. Um *sharded cluster* é formado por 3 entidades, como ilustrado na Figura 3.5 - um conjunto de *shards* que armazenam os subconjuntos de dados; um conjunto de servidores de configuração (chamado de *config server*) que armazenam os metadados e arquivos de configuração do *cluster*, e uma instância *mongos*, a qual funciona como interface entre o *cluster* e as aplicações do usuário. As principais vantagens do uso de *sharding* são: aumento da RAM disponível, aumento do espaço em disco, redução a carga de trabalho em um servidor e permitir *througoutput* maior que apenas uma única instância permite. À medida que novos *shards* são adicionados ao *cluster*, seu desempenho aumenta.

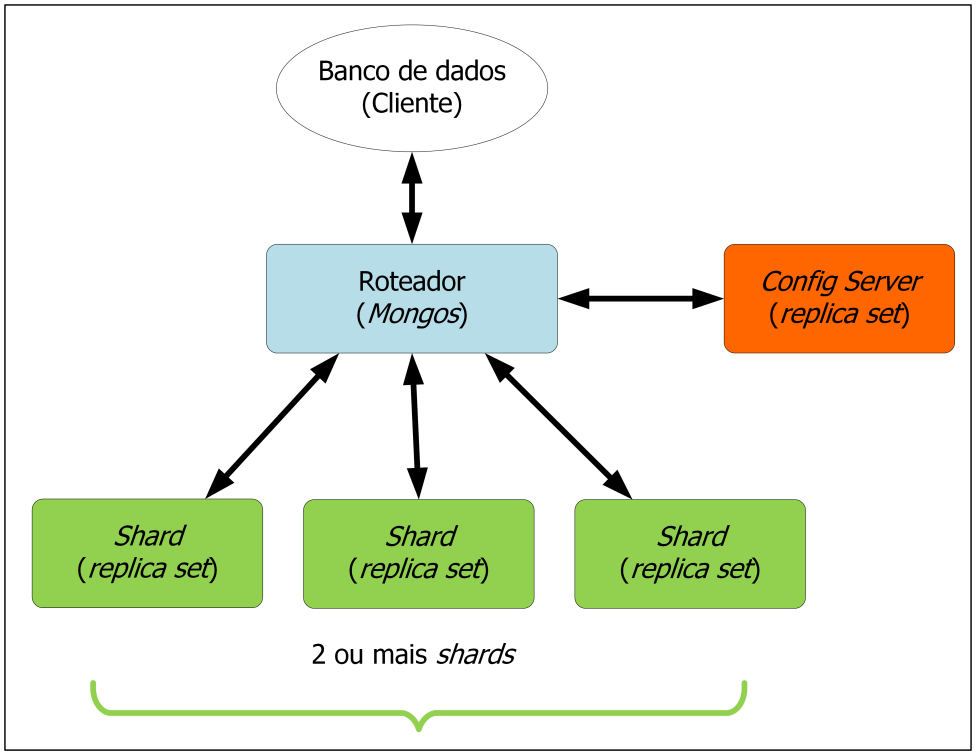

Figura 3.5: Entidades de um *sharded cluster*. Adaptado de [Harrison, 2015].

Os *clusters* utilizados para os testes de inserção e extração dos dados biológicos no MongoDB foram os mesmos utilizados para o Cassandra, com 2, 4 e 6 máquinas (*shards*) e fatores de replicação 1, 2 e 3 (*replica sets*). No entanto, foi necessário adicionar uma máquina em cada *cluster* do MongoDB, que foi utilizada para armazenar os metadados e arquivos de configuração do *cluster* (*config server*). Como no Cassandra, a aplicação

desenvolvida para o MongoDB foi executada 10 vezes em cada ambiente computacional distribuído definido.

Capítulo 4

Análise de Resultados

Neste capítulo são apresentados os resultados obtidos e a análise dos dados, sendo dividido em duas seções: a Seção 4.1 mostra os resultados dos testes de inserção e extração em cada NoSQL; e a Seção 4.2 mostra as análises realizadas a partir dos resultados obtidos.

4.1 Resultados das Execuções dos *Workflows*

Os testes das inserções e extrações dos arquivos nos NoSQL para cada ambiente foram executados de duas formas diferentes, denominadas de execuções frias e execuções quentes. As execuções frias foram realizadas logo após todas as máquinas serem ligadas e o *cluster* integrado pela primeira vez, sem que houvesse nenhuma interação com o banco de dados anteriormente. As execuções quentes foram realizadas quando o banco de dados já estava sendo utilizado.

Foram realizadas dez execuções frias e dez execuções quentes para cada *workflow* em cada um dos ambientes utilizando os NoSQL Cassandra e MongoDB. Além disso, a fim de evitar distorções, foram desconsiderados o maior e o menor tempo para o cálculo da média.

A Figura 4.1 mostra o tempo de inserção dos arquivos do *workflow* 3 para cada teste, sendo que a barra vermelha representa a execução quente e a barra azul representa a execução fria. Como pode ser visto nos gráficos não há uma grande variação entre as duas barras, porém os tempos da execução quente são sempre maiores que os da execução fria. Isso se dá, provavelmente, devido ao fato de que, durante as execuções quentes, a árvore de memória dos NoSQL está cheia e necessita realizar o envio dos arquivos para o disco um número maior de vezes do que na execução fria. As análises dos dados serão feitas com base nos resultados obtidos nos testes quentes, para cada *workflow*, explorando as fases que os compõe e os formatos dos arquivos de cada fase.

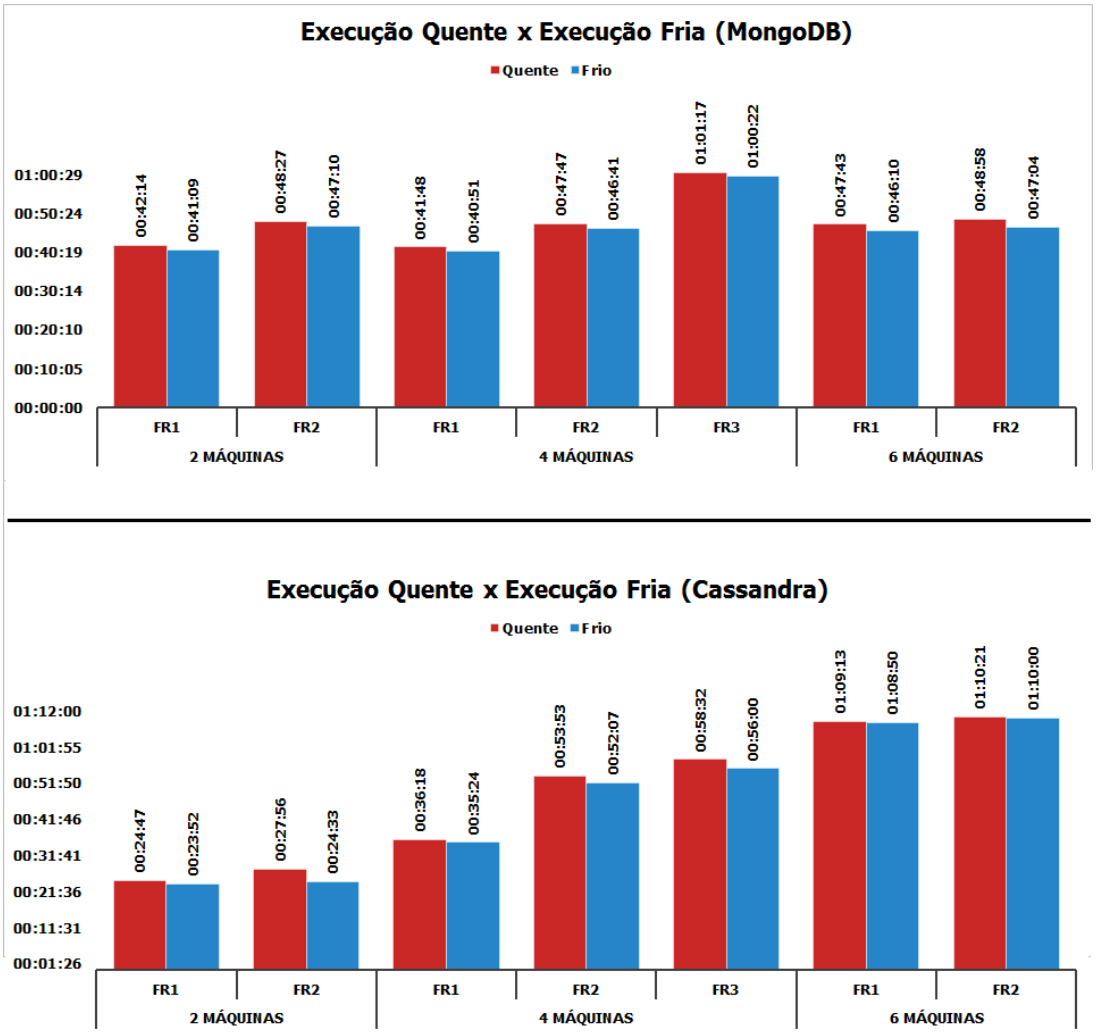

Figura 4.1: Teste quente X Teste frio em cada NoSQL.

Primeiramente foram realizadas as inserções dos arquivos dos *workflows* em cada NoSQL, iniciando-se pelos dados de entrada de cada *workflow*, também chamados de dados brutos e em seguida pelos arquivos de saída de cada uma das suas fases. Os dados brutos são essenciais para a reexecução de um *workflow*, sendo os arquivos iniciais utilizados em sua primeira fase. Os tempos de inserção de todos os arquivos que compõem cada *workflow* nos NoSQL Cassandra e MongoDB são apresentados nas Tabelas 4.1 e 4.2 respectivamente.

Nessas tabelas os tempos estão expressos em horas, minutos e segundos (h:mm:ss), sendo FR1, FR2, FR3 os fatores de replicação 1, 2 e 3, respectivamente. As execuções frias são representadas pela letra F e as quentes pela letra Q. O tempo apresentado representa a soma dos tempos para inserção de todos os arquivos no *workflow* (W) no Cassandra e no MongoDB. Pode-se observar na Tabela 4.1, por exemplo, que o tempo médio total para a inserção dos arquivos do *workflow* 1 no Cassandra, no teste quente,

em um *cluster* de 2 máquinas com fator de replicação 1 foi 10 segundos, enquanto no MongoDB foi 21 segundos.

Tabela 4.1: Inserção de todos arquivos no Cassandra.

		Inserção de todos os arquivos - Cassandra						
		2 máquinas		**4 máquinas**			**6 máquinas**	
Teste	**W**	**FR1**	**FR2**	**FR1**	**FR2**	**FR3**	**FR1**	**FR2**
Q	**W1**	00:00:10	00:00:25	00:00:13	00:00:16	00:00:31	00:00:22	00:00:28
	W2	00:02:47	00:03:28	00:04:31	00:06:29	00:07:00	00:06:16	00:08:01
	W3	00:24:47	00:27:56	00:36:18	00:53:53	00:58:32	01:09:13	01:10:21
F	**W1**	00:00:11	00:00:21	00:00:11	00:00:11	00:00:30	00:00:18	00:00:28
	W2	00:02:42	00:03:22	00:05:27	00:06:06	00:06:56	00:07:00	00:06:43
	W3	00:23:52	00:24:33	00:35:24	00:52:07	00:56:00	01:08:50	01:10:00

Tabela 4.2: Inserção de todos arquivos no MongoDB.

		Inserção de todos os arquivos - MongoDB						
		2 máquinas		**4 máquinas**			**6 máquinas**	
Teste	**W**	**FR1**	**FR2**	**FR1**	**FR2**	**FR3**	**FR1**	**FR2**
Q	**W1**	00:00:21	00:00:22	00:00:20	00:00:24	00:00:28	00:00:23	00:00:25
	W2	00:04:50	00:05:24	00:04:44	00:05:27	00:06:41	00:05:29	00:05:37
	W3	00:42:14	00:48:27	00:41:48	00:47:47	01:01:17	00:47:43	00:48:58
F	**W1**	00:00:23	00:00:24	00:00:21	00:00:25	00:00:30	00:00:23	00:00:24
	W2	00:05:05	00:05:39	00:04:59	00:05:38	00:06:46	00:05:30	00:05:44
	W3	00:44:09	00:49:50	00:43:51	00:50:41	00:59:22	00:48:10	00:50:04

Após os testes de inserções, foram realizados os testes de extrações (leitura) dos arquivos dos *workflows*. As Tabelas 4.3 e 4.4 referem-se ao somatório dos tempos, expresso em horas, minutos e segundos (h:mm:ss), para extrair os arquivos de entrada de cada *workflow* e os arquivos de saída de cada fase em cada *cluster*, sendo FR1, FR2, FR3 os fatores de replicação 1, 2 e 3, respectivamente. Pode-se ver na Tabela 4.3, por exemplo, que o tempo médio total de extração de todos os arquivos do *workflow* 1, em um teste quente, num *cluster* de 2 máquinas com fator de replicação 1 no Cassandra foi 11 segundos, enquanto no MongoDB (Tabela 4.4) foi 14 segundos.

Tabela 4.3: Extração de todos os arquivos no Cassandra.

		Extração de todos os arquivos - Cassandra						
		2 máquinas		**4 máquinas**			**6 máquinas**	
Teste	**W**	**FR1**	**FR2**	**FR1**	**FR2**	**FR3**	**FR1**	**FR2**
Q	**W1**	00:00:11	00:00:10	00:00:10	00:00:09	00:00:09	00:00:07	00:00:07
	W2	00:02:32	00:02:07	00:02:27	00:01:57	00:01:27	00:01:39	00:01:16
	W3	00:20:57	00:19:42	00:20:55	00:17:00	00:12:09	00:19:04	00:15:38
F	**W1**	00:00:11	00:00:10	00:00:11	00:00:08	00:00:08	00:00:06	00:00:10
	W2	00:02:54	00:02:04	00:02:11	00:01:46	00:01:28	00:01:47	00:01:12
	W3	00:20:51	00:19:25	00:21:14	00:18:50	00:11:52	00:19:11	00:14:57

Tabela 4.4: Extração de todos os arquivos no MongoDB.

		Extração de todos os arquivos - MongoDB						
		2 máquinas		**4 máquinas**			**6 máquinas**	
Teste	**W**	**FR1**	**FR2**	**FR1**	**FR2**	**FR3**	**FR1**	**FR2**
Q	**W1**	00:00:14	00:00:14	00:00:14	00:00:14	00:00:14	00:00:14	00:00:14
	W2	00:03:58	00:03:48	00:04:03	00:04:02	00:03:58	00:04:08	00:04:00
	W3	00:33:46	00:34:20	00:34:19	00:34:32	00:34:06	00:35:05	00:34:12
F	**W1**	00:00:14	00:00:15	00:00:14	00:00:14	00:00:14	00:00:15	00:00:14
	W2	00:04:00	00:04:03	00:04:07	00:04:08	00:04:01	00:04:05	00:03:58
	W3	00:34:42	00:36:15	00:35:20	00:35:21	00:34:26	00:34:49	00:34:58

4.1.1 Teste de Tolerância a Falhas

Além dos testes de inserção e extração dos arquivos para obter os tempos, foram realizados também testes de tolerância a falhas em cada NoSQL em estudo. Esses testes foram realizados para garantir que, mesmo se alguma máquina que integra o *cluster* for desligada por qualquer motivo, os arquivos permanecem disponíveis nos bancos de dados e podem ser extraídos.

Para realizar esses testes, foi inicialmente utilizado um dos *clusters* montados, o composto por 4 máquinas e fator de replicação 3. Os testes em ambos NoSQL seguiram os mesmos passos. O primeiro passo foi inserir os arquivos no banco de dados. Após, foram extraídos os arquivos com todas as máquinas ligadas. Como foi possível realizar a inserção e extração, desligou-se uma máquina por vez, e a cada máquina desligada, foi realizada a extração dos arquivos. Ao fim, apenas uma máquina estava ligada e mesmo assim ainda era possível extrair os arquivos armazenados nos bancos de dados.

4.2 Análise dos Dados

Esta seção está dividida em subseções onde cada *workflow* é apresentado em detalhe e em seguida uma análise comparativa.

4.2.1 *Workflow* 1

O primeiro *workflow* executado, possui arquivos de entrada no formato FASTA e é composto apenas da fase de mapeamento, a qual possui arquivos no formato SAM e BED. Os gráficos da Figura 4.2 ilustram as médias dos tempos de inserção e extração dos arquivos de entrada no Cassandra (barra azul) e no MongoDB (barra alaranjada). Esses arquivos FASTA possuem um tamanho total de 115,5 MB.

Como pode ser visto no gráfico, tanto para o MongoDB quanto para o Cassandra, os tempos de inserção aumentam quanto maior o número de máquinas do *cluster* e o fator

de replicação. Para o tempo de extração acontece o inverso, quanto maior o fator de replicação utilizado, menor o tempo de extração dos arquivos. Além disso, os tempos de extração do Cassandra, para o *workflow* 1, foram melhores que os tempos do MongoDB. A inserção no Cassandra no *cluster* com 6 máquinas e fator de replicação 1 gastou 18 segundos e a extração 3 segundos (barras destacadas no gráfico da Figura 4.2). Considerando essa mesma configuração no *cluster*, no MongoDB para a inserção foi gasto 17 segundos e para extração 10 segundos.

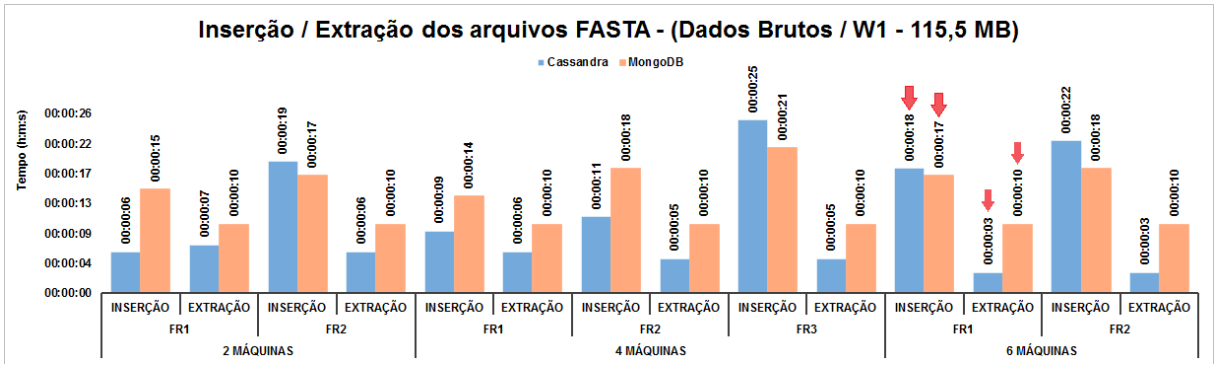

Figura 4.2: Inserção e extração dos arquivos de entrada FASTA em cada NoSQL (*Workflow* 1).

Os arquivos de saída da fase de mapeamento do *workflow* 1 possuem 36,7 MB de tamanho, sendo 13,1 MB o arquivo no formato SAM e 23,6 MB os arquivos no formato BED. Os gráficos apresentados nas Figuras 4.3 e 4.4, referem-se às médias dos tempos de inserção e extração dos arquivos SAM e BED em cada NoSQL. Como pode ser observado nos gráficos, as alterações nos tempos gastos com estes arquivos são mínimas, variando apenas 1 segundo, e por isso não é possível encontrar um padrão para os tempos de inserção e extração desses arquivos nos NoSQL. Isso ocorre por serem arquivos muito pequenos, e portanto, para armazenar arquivos pequenos pode-se aumentar o número de replicação e máquinas utilizadas, de forma a garantir uma confiabilidade dos dados maior, sem perder desempenho.

A soma dos tempos de inserção e extração de todos os arquivos, retrata o tempo total gasto para inserir e extrair todos os arquivos que compõe o *workflow* 1, como apresentados nos gráficos da Figura 4.5.

De acordo com a Figura 4.5, pode-se notar que no geral o Cassandra mostrou um desempenho melhor que o MongoDB, pois realizou as operações de inserção e extração em tempos menores. Para o MongoDB, os tempos de extração dos arquivos são constantes em 14 segundos, independente do número que máquinas do *cluster* e do fator de replicação. No Cassandra, esse tempo sofreu uma queda quando se aumentou o número de máquinas, sendo gastos 11 segundos para extrair em um *cluster* com 2 máquinas e fator de replicação

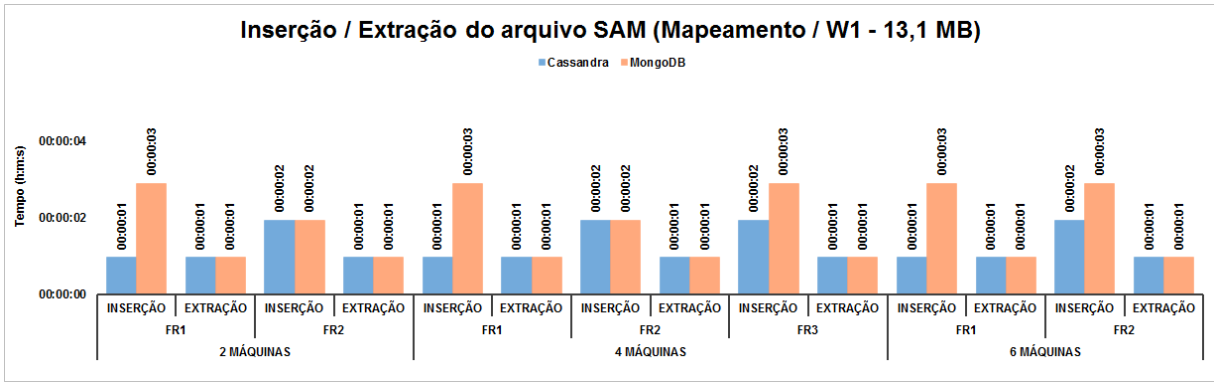

Figura 4.3: Inserção e extração do arquivo SAM da fase de mapeamento em cada NoSQL (*Workflow* 1).

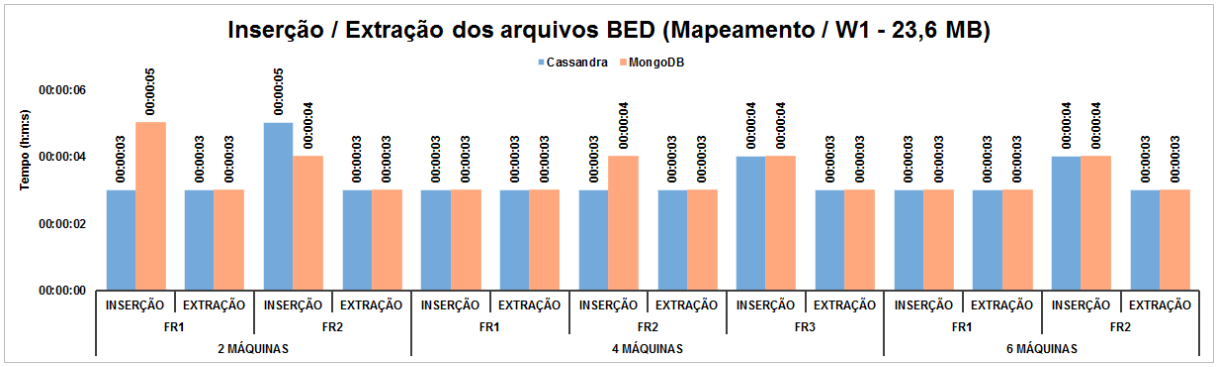

Figura 4.4: Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 1).

1 e apenas 7 segundos em um *cluster* com 6 máquinas e fator de replicação 2, ambos destacados na figura.

O tempo de inserção em ambos NoSQL, é maior à medida que se aumenta o número de máquinas do *cluster* e fator de replicação. No entanto, para o MongoDB, a variação dos tempos não é tão significante quanto no Cassandra. Como pode ser observado na Figura 4.5, destacados com a seta vermelha, o Cassandra gastou o tempo de 10 segundos para inserir os arquivos no *cluster* com 2 máquinas e fator de replicação 1 e 28 segundos para o *cluster* com 4 máquinas e fator de replicação 3.

4.2.2 *Workflow* 2

O *workflow* 2 possui arquivos de entrada no formato FASTQ que somados possuem um tamanho de 1,35 GB. A Figura 4.6 refere-se às médias dos tempos de inserção e extração desses arquivos em cada NoSQL. Nesta figura, é possível perceber que, quando se trata da inserção dos arquivos, tanto para o Cassandra quanto para o MongoDB, à medida que aumenta o número de máquinas e o fator de replicação, o tempo de resposta da operação

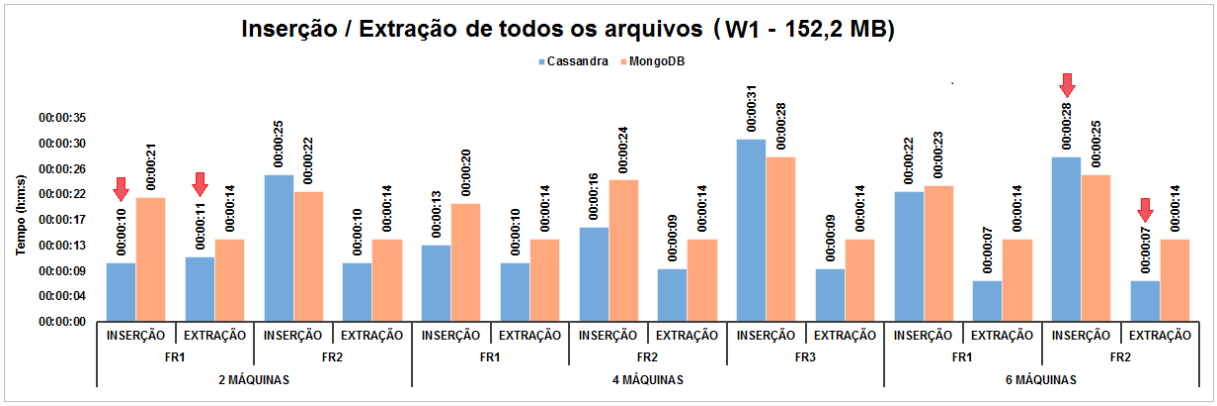

Figura 4.5: Inserção e extração de todos os arquivos do *Workflow* 1 em cada NoSQL.

também aumenta. No entanto, no Cassandra, a variação do tempo de inserção é grande nos diferentes *clusters*, com 1 minuto e 23 segundos no *cluster* com 2 máquinas e fator de replicação 1, e 4 minutos e 22 segundos no *cluster* com 6 máquinas e fator de replicação 2 (em destaque na figura). No MongoDB, a inserção nos *clusters* menores tem um tempo de resposta inferior ao Cassandra. Entretanto, o MongoDB tem um desempenho melhor que o Cassandra, quando se usa mais máquinas no *cluster*.

O tempo de extração dos arquivos de entrada FASTQ é menor quando se utiliza o Cassandra. Comparando o tempo de extração de ambos NoSQL no *cluster* com 6 máquinas e fator de replicação 2, pode-se perceber, em destaque na Figura 4.6, que para o MongoDB foram necessários 2 minutos e 14 segundos, enquanto no Cassandra foi apenas de 36 segundos.

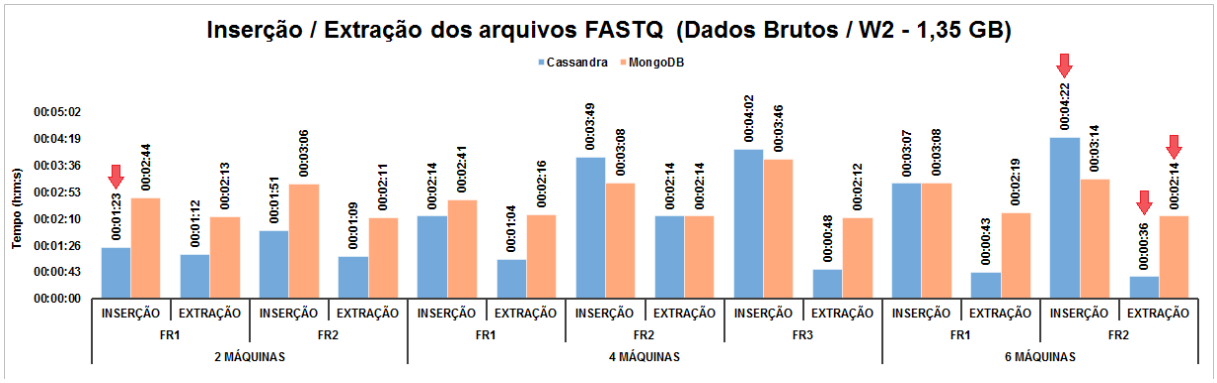

Figura 4.6: Inserção e extração dos arquivos de entrada FASTQ em cada NoSQL (*Workflow* 2).

Os demais formatos de arquivos desse *workflow* são provenientes da execução de suas fases. As fases que compõem esse *workflow* são mapeamento/montagem e anotação. No mapeamento, foram obtidos arquivos em dois formatos diferentes: BAM e BED, sendo seus tamanhos 422 MB e 324 MB respectivamente. Na montagem, há apenas um formato

de arquivo, FASTA, e é a fase com o menor arquivo obtido, sendo 11 MB. Por último, na fase de anotação, tem-se os arquivos no formato TXT, com 610 MB de tamanho.

Os tempos de inserção e extração dos arquivos BAM e BED do mapeamento, são ilustrados nas Figuras 4.7 e 4.8. A inserção dos arquivos BAM (Figura 4.7), em ambos os NoSQL, tem um tempo maior, quando são utilizados mais máquinas no *cluster* e maior fator de replicação. Já para a extração, acontece o inverso: mais máquinas e maior fator de replicação implica em um menor tempo de extração. No entanto, no Cassandra, a variação desse tempo é maior que no MongoDB. Enquanto no Cassandra o tempo variou entre 47 e 16 segundos (maior e menor tempo, em destaque na figura), no MongoDB permaneceu quase estável, oscilando entre 41 e 43 segundos. Então, a quantidade de máquinas e do fator de replicação no MongoDB, para este tamanho e formato de arquivo, não afeta o desempenho, enquanto no Cassandra aumenta. A inserção e extração dos arquivos BED se manteve constante em ambos NoSQL.

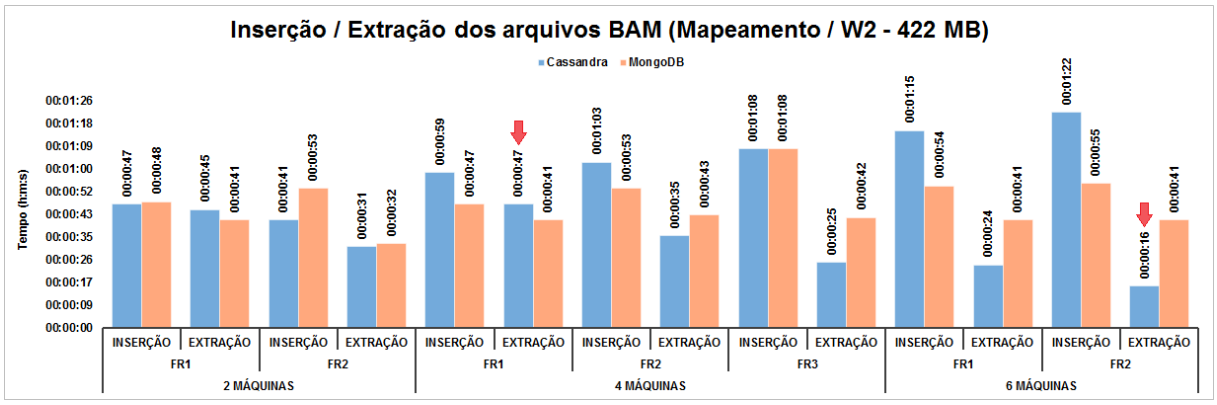

Figura 4.7: Inserção e extração dos arquivos BAM da fase de mapeamento em cada NoSQL (*Workflow* 2).

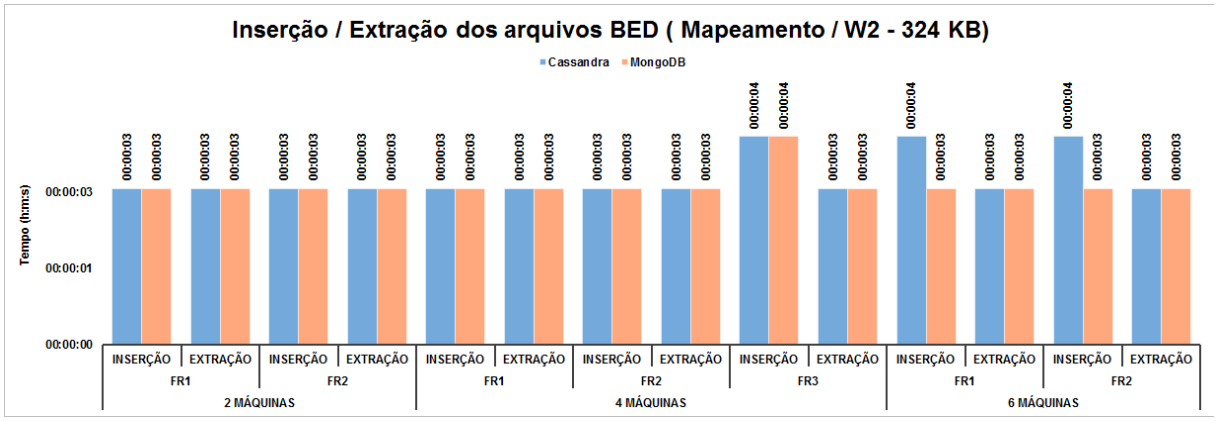

Figura 4.8: Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 2).

Na montagem, foi obtido o menor arquivo do *workflow*, no formato FASTA. Por esse motivo, os tempos de inserção e extração foram praticamente iguais, com uma pequena variação de apenas 1 segundo para a inserção no Cassandra. Os tempos desse arquivo podem ser vistos na Figura 4.9.

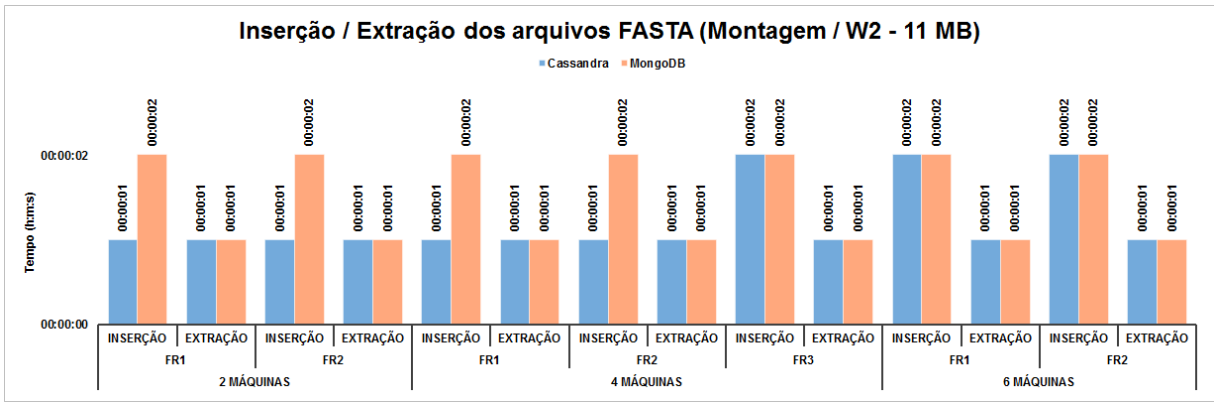

Figura 4.9: Inserção e extração dos arquivos FASTA da fase de montagem em cada NoSQL (*Workflow* 2).

O último arquivo obtido na execução do *worklfow* 2 foi no formato TXT, resultado da fase de anotação. A inserção desse arquivo no MongoDB, possui um tempo melhor que o Cassandra, quando é utilizado o *cluster* maior. No entanto, a diferença entre os tempos dos NoSQL são pequenas. Já a extração dos arquivos, possui uma larga variação entre os tempos de cada NoSQL, sendo que o Cassandra mostrou melhor desempenho para esta operação. Como pode ser visto na Figura 4.10, indicado por uma seta vermelha, no *cluster* com 6 máquinas e fator de replicação 2, MongoDB gastou 1 minuto e 1 segundo para extrair o arquivo, enquanto o Cassandra gastou apenas 20 segundos.

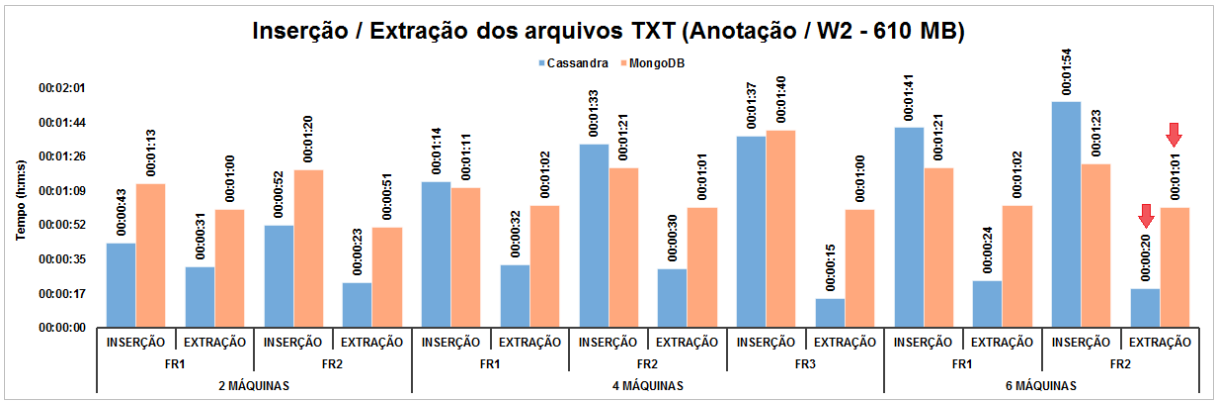

Figura 4.10: Inserção e extração dos arquivos TXT da fase de anotação em cada NoSQL (*Workflow* 2).

Após a análise de cada formato foram somados os tempos de cada operação de todos os arquivos, para obter o tempo total de inserção e extração dos arquivos do *workflow*

2. Na Figura 4.11 é ilustrado esse tempo. Como pode ser observado, na inserção dos arquivos quando se utiliza *cluster* com menos máquinas, o Cassandra gasta menos tempo para finalizar a operação. Entretanto, quando se aumenta o número de máquinas que compõem o *cluster*, o MongoDB finaliza a operação de inserção em menos tempo que o Cassandra. Para a extração dos arquivos do *workflow* 2, o Cassandra gastou menos tempo que o MongoDB. O tempo encontrado para o MongoDB na extração de todos os arquivos no *cluster* com 6 máquinas e fator de replicação 2, foi de 4 minutos, marcado na Figura 4.11, e no Cassandra, o tempo para essa mesma configuração no *cluster* é de 1 minuto e 16 segundos.

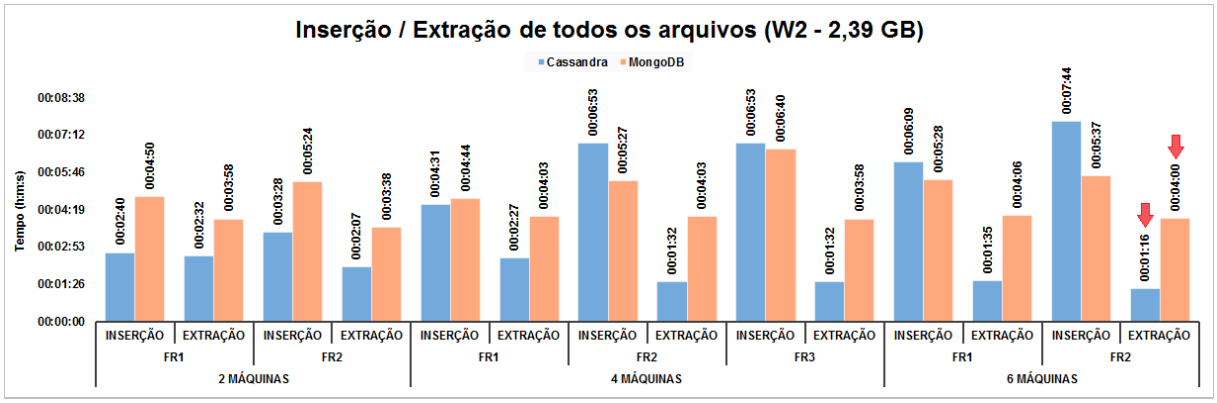

Figura 4.11: Inserção e extração de todos arquivos do *workflow* 2 em cada NoSQL.

4.2.3 *Workflow* 3

O *workflow* 3 é composto pelas fases de filtragem, mapeamento/montagem e anotação, sendo o *workflow* que envolve o maior número de fases e com maiores tamanhos dos arquivos, desta dissertação. Como dito anteriormente a execução total foi 23 horas, 28 minutos e 48 segundos e como resultados foram obtidos arquivos nos formatos FASTA, FASTQ, BAM, BED e TXT.

Os dados de entrada deste *workflow* totalizam 7,8 GB e estão no formato FASTQ. A Figura 4.12, refere-se aos gráficos com as médias dos tempos em cada NoSQL. Como pode ser visto na figura, a inserção dos dados em ambos os bancos de dados aumenta à medida que aumenta também o número de máquinas no *cluster*. No entanto, no MongoDB a variação entre os tempos é menor que no Cassandra, e por esse motivo, quando se tem um *cluster* com mais máquinas, a inserção no Cassandra é mais lenta.

Os tempos de extração dos dados de entrada diminui quando aumenta o número de máquinas e o fator de replicação. Esse tempo é menor em todos os ambientes computacionais, quando é utilizado o Cassandra. Como é mostrado na Figura 4.12, para extrair os arquivos de entrada do *workflow* 3 no *cluster* com 4 máquinas e fator de replicação 1

no MongoDB foi necessário 12 minutos e 27 segundos, enquanto no Cassandra apenas 4 minutos e 58 segundos.

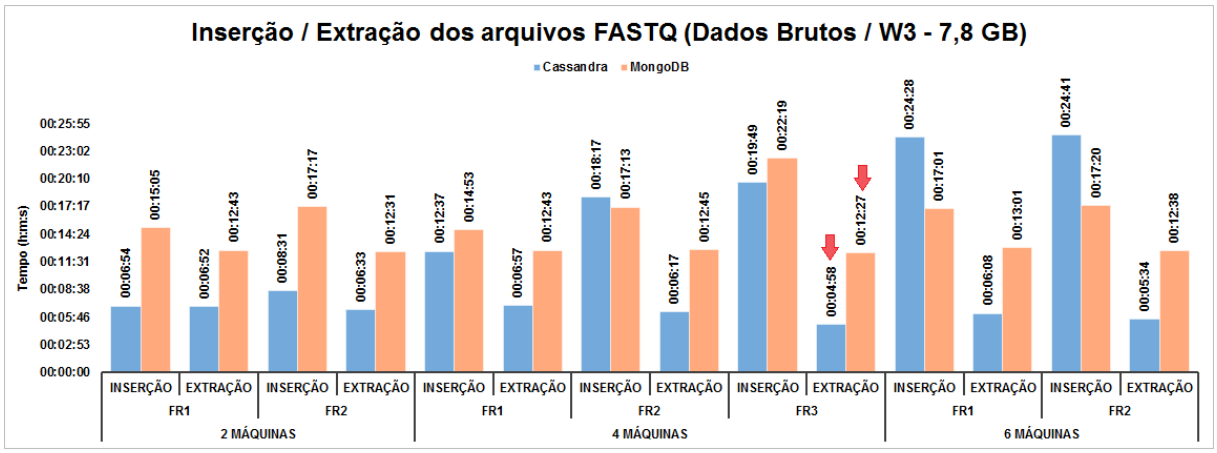

Figura 4.12: Inserção e extração dos arquivos de entrada FASTQ em cada NoSQL (*Workflow* 3).

Os arquivos obtidos na fase de filtragem possuem formatos FASTA e FASTQ, sendo 3,82 GB e 7,8 GB seus respectivos tamanhos. Tanto a inserção quanto a extração desses arquivos nos dois bancos de dados, seguiram o mesmo raciocínio da análise feita para os arquivos de entrada do *workflow* 3, na qual a operação de inserção é finalizada em menos tempo no MongoDB quando se utiliza os *clusters* com menor número de máquinas e a extração tem menor tempo quando se utiliza o Cassandra. Isso acontece, pois os arquivos FASTQ da filtragem e da entrada possuem o mesmo tamanho, e os arquivos FASTA, mesmo com tamanho menor, tem um comportamento equivalente ao formato FASTQ. Os tempos de inserção e extração dos arquivos FASTQ e FASTA da filtragem são ilustrados nas Figuras 4.13 e 4.14. Como pode ser observado na Figura 4.13, o menor tempo de extração obtido para o Cassandra foi de 3 minutos e 30 segundos, no *cluster* com 4 máquinas e fator de replicação 3, e o menor tempo de extração para o MongoDB nesta mesma configuração foi de 12 minutos e 42 segundos.

Na Figura 4.14, é possível perceber que no *cluster* com 4 máquinas e fator de replicação 3, a inserção no NoSQL Cassandra gastou 10 minutos e 46 segundos para ser finalizada e a extração 1 minuto e 35 segundos. No MongoDB, a inserção nessa mesma configuração durou 10 minutos e 52 segundos e a extração 6 minutos e 25 segundos.

Os arquivos no formato BAM e BED foram obtidos no mapeamento. Os arquivos BAM do *workflow* 3 possuem 1,24 GB de tamanho. Na Figura 4.15, com os tempos marcados, pode-se perceber que a inserção, no *cluster* com 6 máquinas e fator de replicação 2, no MongoDB durou 2 minutos e 50 segundos enquanto no Cassandra o tempo foi de 5 minutos e 3 segundos. Já a extração durou apenas 42 segundos no MongoDB e 1 minuto e

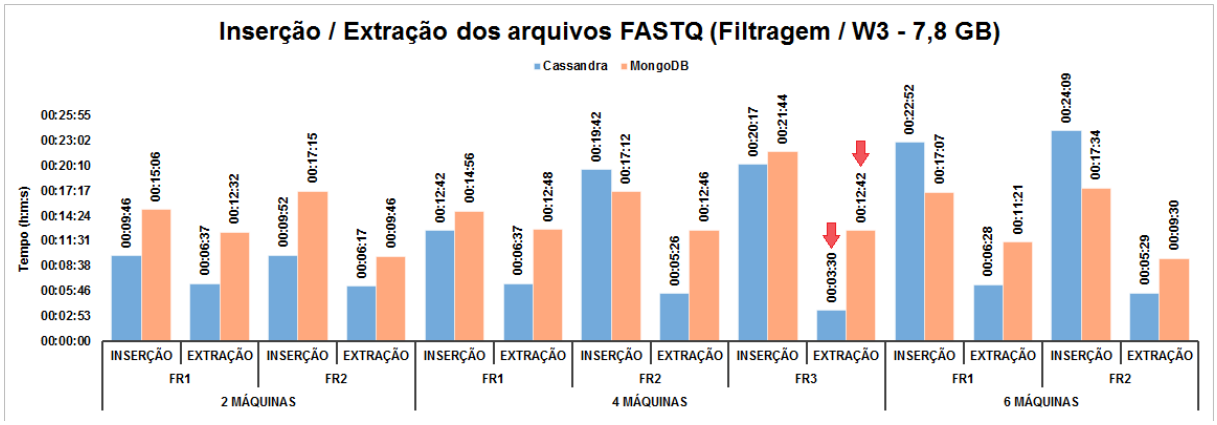

Figura 4.13: Inserção e extração dos arquivos FASTQ da fase de filtragem em cada NoSQL (*Workflow* 3).

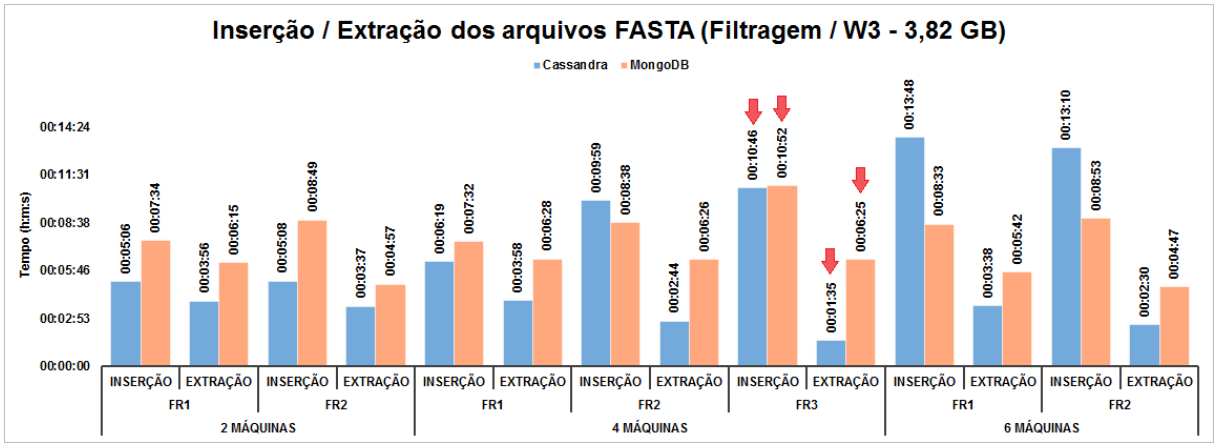

Figura 4.14: Inserção e extração dos arquivos FASTA da fase de filtragem em cada NoSQL (*Workflow* 3).

28 segundos no Cassandra. Apenas na configuração com 2 máquinas e fator de replicação 1 o Cassandra teve um tempo de inserção menor que o MongoDB.

Os tempos obtidos com os testes dos arquivos no formato BED do *workflow* 3, foram constantes em todos ambientes computacionais utilizados. O tempo para inserir e extrair os arquivos em ambos NoSQL foi de 3 segundos, como é mostrado na Figura 4.16.

Na fase de montagem, os arquivos de saída no formato FASTA possuem tamanho de 21,7 MB. Os tempos de inserção em ambos NoSQL tiveram pouca variação, entre 2 e 4 segundos. A extração dos arquivos, permaneceu com tempo constante em todos os ambientes computacionais utilizados. Essa pouca variação nos tempos ocorre por se tratar de arquivos pequenos e por esse motivo não há perda ou ganho de desempenho nas operações de inserção e extração de arquivos. Os gráficos da Figura 4.17 mostram os tempos de inserção e extração dos arquivos FASTA da fase de montagem do *workflow* 3.

A última fase do *workflow* 3 é a anotação. Nesta fase, foram gerados arquivos no

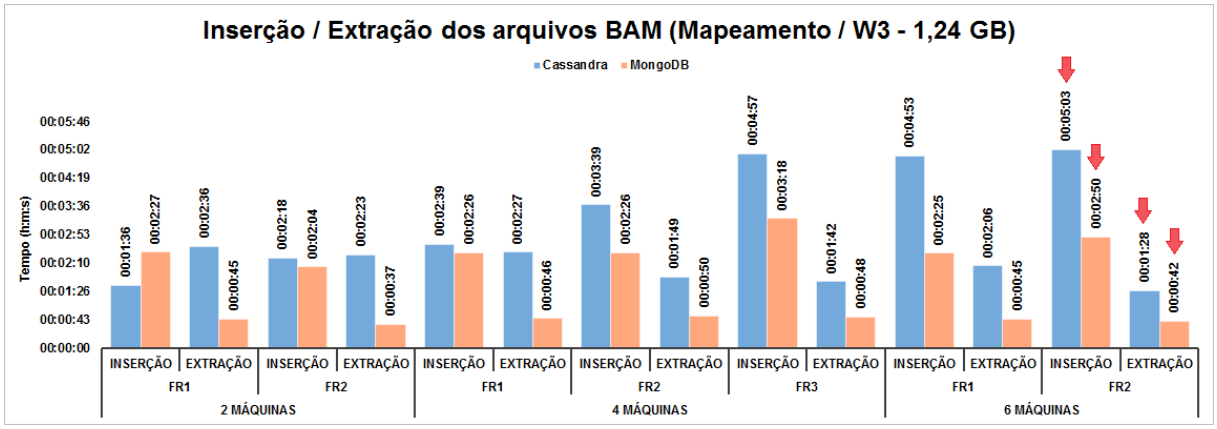

Figura 4.15: Inserção e extração dos arquivos BAM da fase de mapeamento em cada NoSQL (*Workflow* 3).

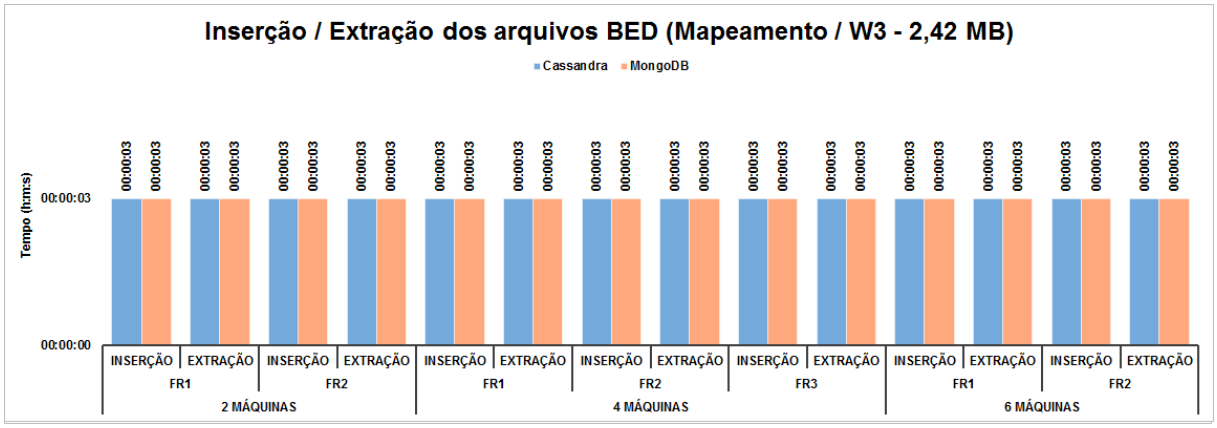

Figura 4.16: Inserção e extração dos arquivos BED da fase de mapeamento em cada NoSQL (*Workflow* 3).

formato TXT, com tamanho de 1 GB. Os tempos de inserção no Cassandra, variam de acordo com o fator de replicação utilizado. Os *clusters* com fatores de replicação 2 e 3, gastaram mais tempo na inserção, sendo que o maior tempo obtido foi 3 minutos e 10 segundos no ambiente computacional com 6 máquinas e fator de replicação 2. Já no MongoDB, nessa mesma configuração do *cluster*, o tempo de inserção foi de 2 minutos e 14 segundos.

Na extração dos arquivos no formato TXT, o tempo de resposta do Cassandra foi menor que o do MongoDB em todos os ambientes computacionais configurados. Como pode ser visto na Figura 4.18, no *cluster* com 4 máquinas e fator de replicação 3, o tempo gasto para extrair os arquivos do Cassandra foi de 21 segundos, enquanto no MongoDB foi de 1 minuto e 35 segundos.

A partir dos resultados obtidos com dados brutos e com o dados de saída de cada fase, é possível fazer uma análise do *workflow* completo. A soma de todos os arquivos do

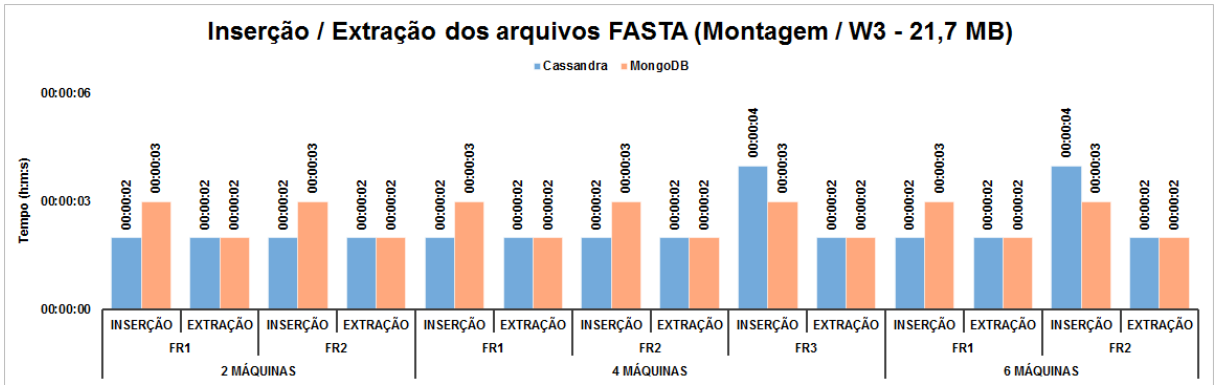

Figura 4.17: Inserção e extração dos arquivos FASTA da fase de montagem em cada NoSQL (*Workflow* 3).

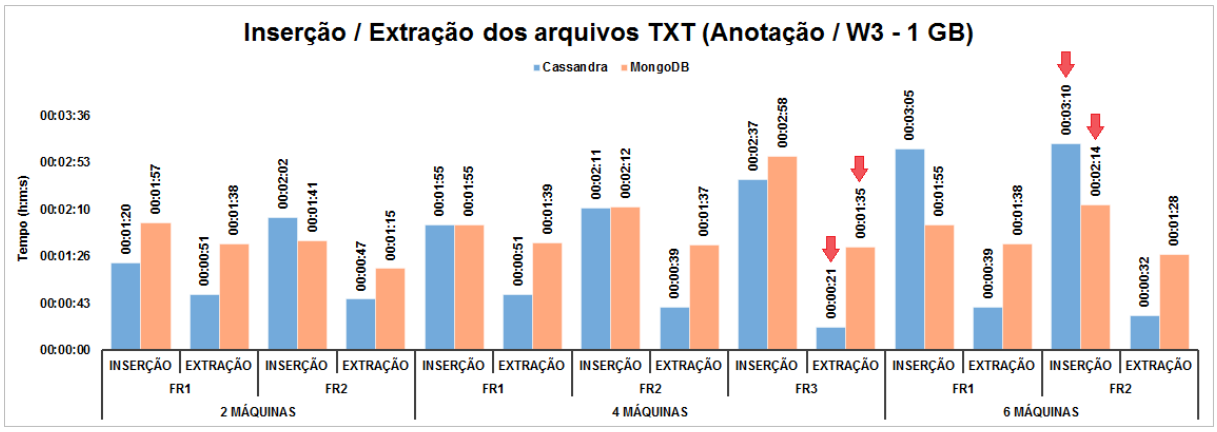

Figura 4.18: Inserção e extração dos arquivos TXT da fase de anotação em cada NoSQL (*Workflow* 3).

workflow 3 totaliza 31,22 GB. A inserção desses arquivos no Cassandra, possui tempo de resposta melhor quando é realizada em *clusters* com menos máquinas e fator de replicação maior, pois neste SGBD, quanto maior número de máquinas integradas, menor o desempenho para a inserção. O tempo de inserção dos arquivos no MongoDB é semelhante ao Cassandra, no que diz respeito à quantidade de máquinas. No entanto, em *clusters* compostos com menos máquinas, o Cassandra apresenta melhor tempo de resposta. Como pode ser visto em destaque na Figura 4.19, no *cluster* com 2 máquinas e fator de replicação 2, o Cassandra gastou 27 minutos e 56 segundos para inserir todos os arquivos, enquanto o MongoDB gastou 47 minutos e 11 segundos para inserir os mesmos arquivos.

Por outro lado, nos dois NoSQL analisados, a extração é realizado em menos tempo quando é utilizado o Cassandra. O tempo de extração de todos os arquivos do *workflow* 3 do Cassandra é quase um terço do tempo gasto no MongoDB. Por exemplo, como pode ser visto na Figura 4.19, no *cluster* com quatro máquinas e fator de replicação 3, foi gasto 12 minutos e 11 segundos para extrair todos os arquivos, enquanto no MongoDB foi gasto

34 minutos e 2 segundos.

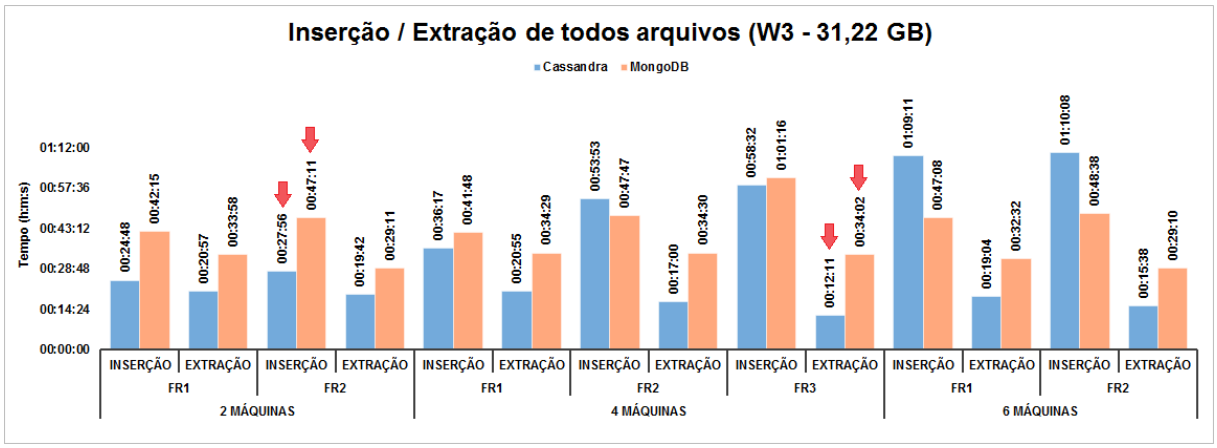

Figura 4.19: Inserção e extração de todos os arquivos em cada NoSQL (*Workflow* 3).

4.2.4 Comparação entre os NoSQLs

Nas subseções anteriores, foram analisados os tempos de inserção de cada formato de arquivo em cada NoSQL em estudo. Além dessa análise, é também de suma importância, fazer uma comparação entre o tempo de execução dos *workflows* e o tempo gasto com os SGBDs. Para isso, foram considerados dois ambientes: o primeiro com o menor fator de replicação utilizado, e com o menor número de máquinas que integram o *cluster*; o segundo com o maior fator de replicação e consequentemente com quatro máquinas integradas, pois foi o único *cluster* com fator de replicação 3.

A Tabela 4.5, refere-se às porcentagens dos tempos de execução do *workflow* (TEW), tempo de inserção dos arquivos (TI) e tempo de extração (TE), em relação ao tempo total do *workflow* (TTW). Como pode ser observado na tabela, o TTW corresponde à 100%, ou seja, é obtido através da soma dos TEW, TI e TE. É possível perceber que em ambos os ambientes analisados, o tempo gasto com as operações de inserção e extração é mínimo quando comparado ao tempo de execução do *workflow*. Como por exemplo, para o *workflow* 1 (W1) quando se utilizou o Cassandra, 94,03% do TTW foi gasto para o TEW, apenas 2,84% para inserir os arquivos e 3,13% para extratir. Além disso, apenas na operação de inserção dos arquivos do *workflow* 1, o MongoDB teve desempenho melhor que o Cassandra, sendo que o primeiro gastou 7,51% do TTW para inserir, enquanto o segundo 8,36%.

As Figuras 4.20 e 4.21, ilustram de forma gráfica a mesma comparação entre os tempos de execução dos *workflows*, e os tempos de inserção e extração dos arquivos exibidos na Tabela 4.5. Em ambas as figuras, cada barra completa representa o tempo total de cada *workflow* em cada NoSQL, ou seja, 100%, e é dividida da seguinte maneira: a cor cinza

Tabela 4.5: Relação percentual entre o tempo execução, inserção e extração dos arquivos de cada *workflow*.

		2 máquinas - FR1		**4 máquinas - FR3**	
		CASSANDRA	**MONGODB**	**CASSANDRA**	**MONGODB**
W1	**TEW**	94,03%	90,44%	89,22%	88,74%
	TI	2,84%	5,74%	8,36%	7,51%
	TE	3,13%	3,82%	2,42%	3,75%
	TTW	**100%**	**100%**	**100%**	**100%**
W2	**TEW**	99,48%	99,14%	99,17%	98,96%
	TI	0,27%	0,47%	0,69%	0,65%
	TE	0,25%	0,39%	0,14%	0,39%
	TTW	**100%**	**100%**	**100%**	**100%**
W3	**TEW**	96,86%	94,85%	95,16%	93,6%
	TI	1,70%	2,84%	4,02%	4,07%
	TE	1,44%	2,31%	0,82%	2,27%
	TTW	**100%**	**100%**	**100%**	**100%**

representa o tempo de execução de cada *workflow* (TEW), a cor amarela representa o tempo de inserção dos arquivos (TI) e a cor vermelha refere-se ao tempo de extração (TE).

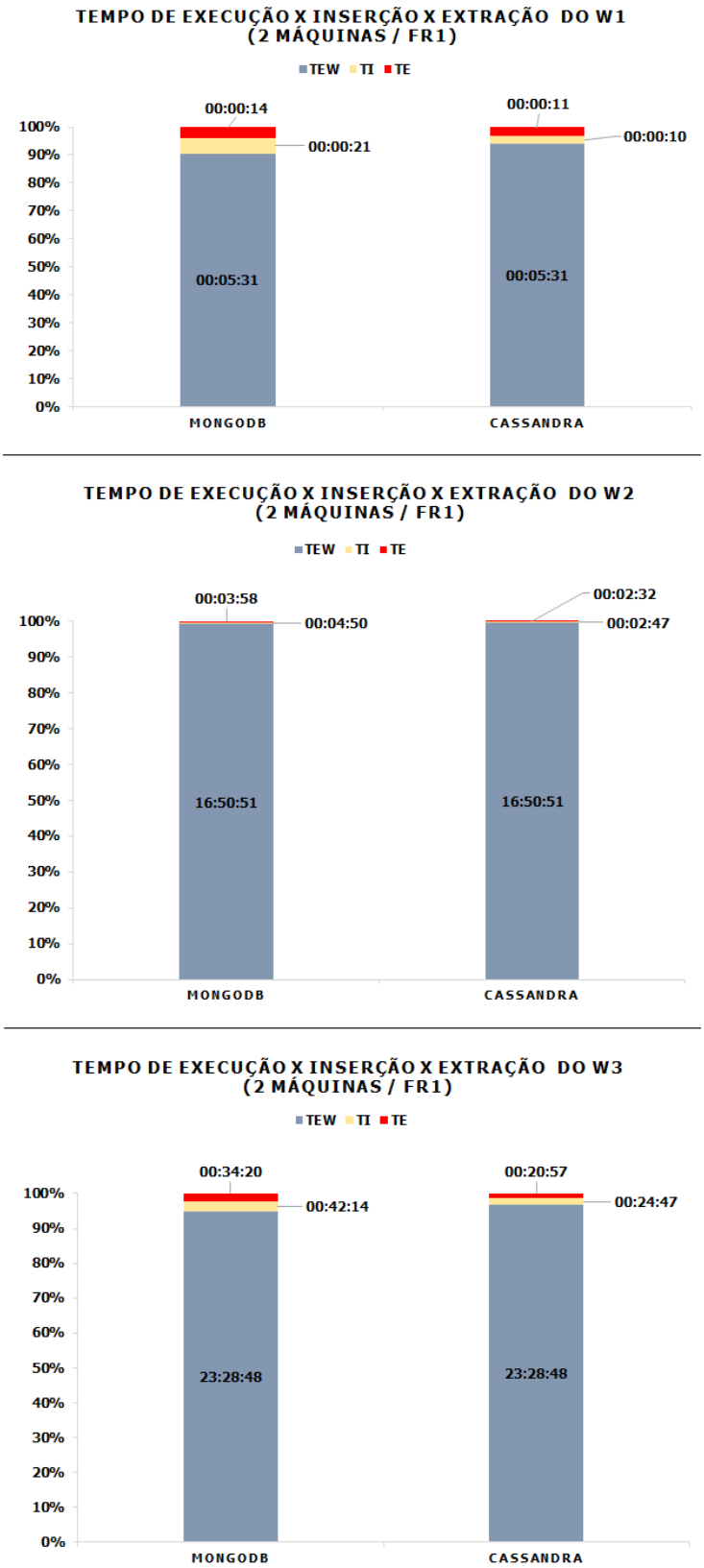

Figura 4.20: Comparação entre o tempo de execução, inserção e extração dos *workflows* 1, 2 e 3 no *cluster* com 2 máquinas e FR1.

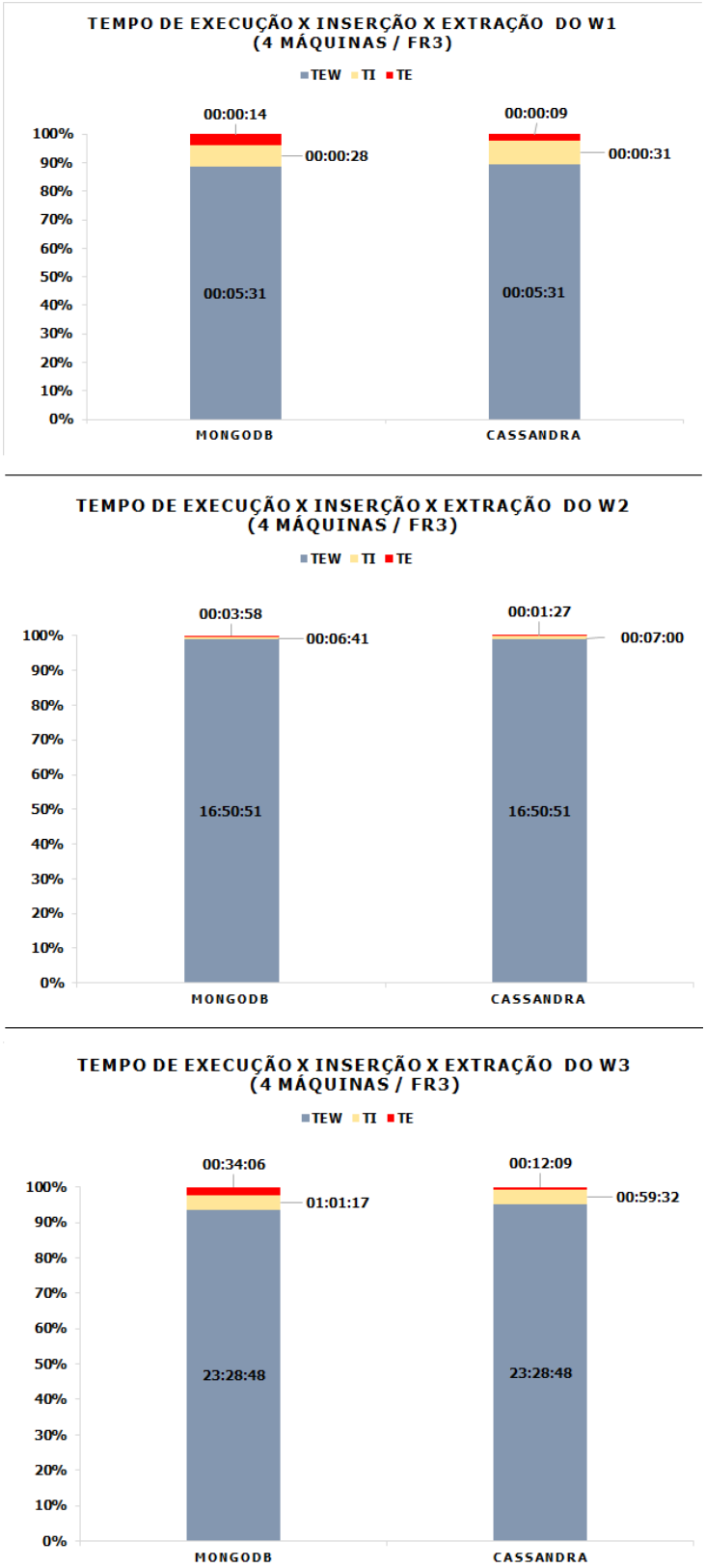

Figura 4.21: Comparação entre o tempo de execução, inserção e extração dos *workflows* 1, 2 e 3 no *cluster* com 4 máquinas e FR3.

Capítulo 5

Conclusão e Trabalhos Futuros

O armazenamento de dados biológicos originados de *workflows* de Bioinformática é um desafio em ambientes computacionais distribuídos. Esta dissertação apresentou a análise do armazenamento de arquivos com dados biológicos, considerando os diversos formatos de arquivos existentes nesse contexto e avaliando o impacto da replicação desses dados nos NoSQL Cassandra e MongoDB. Com os resultados obtidos e a análise apresentada, foi possível identificar diversas situações de uso dos SGBD Cassandra e MongoDB. De acordo com a necessidade do usuário, deve-se determinar qual é a melhor configuração para o ambiente computacional distribuído que deseja utilizar.

De acordo com as análises anteriores, em um ambiente computacional distribuído com maior número de máquinas e fator de replicação maior, o tempo gasto para as operações de inserção de dados biológicos, em ambos NoSQL, é maior, ou seja, há um ganho de disponibilidade dos dados porém uma perda de desempenho para inserção. Nesses mesmos ambientes, no qual o número de máquinas e o fator de replicação é maior, foi mostrado também que o sistema gerenciador de banco de dados NoSQL Cassandra possui um menor tempo para as leituras dos arquivos dos *workflows* que o MongoDB. Além disso, as análises apresentadas demonstraram que o tempo de execução de um *workflow* de Bioinformática é muito maior que o tempo de inserção e extração de seus respectivos arquivos.

Em projetos de Bioinformática, os arquivos com os dados biológicos são inseridos apenas uma vez e existem diversas requisições (leituras ao banco de dados) de tais arquivos. A partir das relações percentuais dos tempos de inserção e extração dos dados biológicos, comparadas aos tempos de execução dos *workflows*, pôde-se perceber que no geral o Cassandra finaliza suas operações em menos tempo. Isso se deve à limitação de tamanho dos documentos no MongoDB, e portanto pode-se dizer que o Cassandra, dentre esses dois NoSQL analisados, é a melhor opção para armazenamento dos tipos de dados estudados nesta dissertação. Logo, ele pode ser considerado uma boa opção de NoSQL para ser integrado a uma arquitetura de um sistema gerenciador de *workflows* em Bioinformática.

A discussão feita nessa dissertação pode trazer um direcionamento para análises em outros modelos de bancos de dados NoSQL com a implementação de aplicações semelhantes às desenvolvidas nessa dissertação e comparar o desempenho entre os modelos de sistemas gerenciadores de bancos de dados NSQL. Além disso, as execuções e as análises dos bancos de dados poderiam ser realizadas utilizando um ambiente em nuvem.

Outros trabalhos futuros a serem desenvolvidos são: análise de outras características além do fator de replicação no ambiente distribuído, como por exemplo, a latência da rede; integração de um banco de dados NoSQL com um sistema gerenciador de *workflows* de Bioinformática; e por fim, uma arquitetura completa para o gerenciamento tanto os dados biológicos quanto da proveniência desses dados.

Referências

[Abramova and Bernardino, 2013] Abramova, V. and Bernardino, J. (2013). NoSQL databases: Mongodb vs cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pages 14–22. ACM. 19, 24

[Aken et al., 2016] Aken, B. L., Ayling, S., Barrell, D., Clarke, L., Curwen, V., Fairley, S., Banet, J. F., Billis, K., Girón, C. G., Hourlier, T., et al. (2016). The Ensembl gene annotation system. *Database*. 10

[Anderson et al., 2010] Anderson, J. C., Lehnardt, J., and Slater, N. (2010). *CouchDB: the definitive guide*. "O'Reilly Media, Inc.". 13

[Aniceto et al., 2015] Aniceto, R., Xavier, R., Guimarães, V., Hondo, F., Holanda, M., Walter, M. E., and Lifschitz, S. (2015). Evaluating the Cassandra NoSQL Database Approach for Genomic Data Persistency. *International Journal of Genomics*, 2015. 25, 26

[Bloom and Sharpe, 2004] Bloom, T. and Sharpe, T. (2004). Managing data from high-throughput genomic processing: a case study. In *Proceedings of the Thirtieth International Conference on Very Large Databases - Volume 30*, pages 1198–1201. VLDB Endowment. 24

[Boicea et al., 2012] Boicea, A., Radulescu, F., and Agapin, L. I. (2012). MongoDB vs Oracle-Database Comparison. In *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, pages 330–335. IEEE. 19, 21

[Brewer, 2012] Brewer, E. (2012). Cap twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29. 11, 12

[Brewer, 2000] Brewer, E. A. (2000). Towards robust distributed systems. In *PODC*, volume 7. 12

[Camacho et al., 2013] Camacho, C., Madden, T., Ma, N., Tao, T., Agarwala, R., and Morgulis, A. (2013). Blast command line applications user manual. 6, 7, 8, 28

[Cattell, 2011] Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, 39(4):12–27. 1

[Chang et al., 2008] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4. 14

[Chodorow, 2013] Chodorow, K. (2013). *MongoDB: the definitive guide, Second Edition*. "O'Reilly Media, Inc.". 13, 19, 32

[Cock et al., 2010] Cock, P. J., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic acids research*, 38(6):1767–1771. 9

[Corbellini et al., 2017] Corbellini, A., Mateos, C., Zunino, A., Godoy, D., and Schiaffino, S. (2017). Persisting big-data: The NoSQL landscape. *Information Systems*, 63:1–23. 2, 13, 20

[da Silva et al., 2016] da Silva, G. H. G., Holanda, M., and Araujo, A. (2016). Data replication policy in a cloud computing environment. In *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE. 21, 22, 25, 26

[Davidson and Freire, 2008] Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350. ACM. 1

[DB-Engines, 2016a] DB-Engines (2016a). DB-Engines Ranking - Trend of Document Stores Popularity. http://db-engines.com/en/ranking_trend/document+store. xi, 16

[DB-Engines, 2016b] DB-Engines (2016b). DB-Engines Ranking - Trend of Wide Column Stores Popularity. http://db-engines.com/en/ranking_trend/wide+column+store. xi, 15

[DB-Engines, 2016c] DB-Engines (2016c). DB-Engines Ranking - Trend Popularity. http://db-engines.com/en/ranking_trend. xi, 2, 15

[de Paula et al., 2013] de Paula, R., Holanda, M., Gomes, L. S., Lifschitz, S., and Walter, M. E. M. (2013). Provenance in Bioinformatics workflows. *BMC Bioinformatics*, 14(11):1. 4

[de Queiroz et al., 2013] de Queiroz, G. R., Monteiro, A. M. V., and Câmara, G. (2013). Bancos de Dados Geográficos e Sistemas NoSQL: onde estamos e para onde vamos. *Revista Brasileira de Cartografia*, 65(3). 20

[de Souza and dos Santos, 2015] de Souza, V. C. O. and dos Santos, M. V. C. (2015). Maturing, Consolidation and Performance of NoSQL Databases-Comparative Study. In *Proceedings of the Annual Conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective - Volume 1*, page 32. Brazilian Computer Society. 11

[DeCandia et al., 2007] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220. 13

[Elmasri, 2008] Elmasri, R. (2008). *Fundamentals of database systems*. Pearson Education India. 22

[Fonseca et al., 2012] Fonseca, N. A., Rung, J., Brazma, A., and Marioni, J. C. (2012). Tools for mapping high-throughput sequencing data. *Bioinformatics*, page bts605. 6

[George, 2011] George, L. (2011). *HBase: the definitive guide*. "O'Reilly Media, Inc.". 14

[Gordon and Hannon, 2010] Gordon, A. and Hannon, G. (2010). Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*. 5

[Grabherr et al., 2011] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., et al. (2011). Trinity: reconstructing a full-length transcriptome without a genome from RNA-Seq data. *Nature biotechnology*, 29(7):644. 5, 7, 28

[Gray et al., 1996] Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. *ACM SIGMOD Record*, 25(2):173–182. 23

[Guimaraes et al., 2015] Guimaraes, V., Hondo, F., Almeida, R., Vera, H., Holanda, M., Araujo, A., Walter, M. E., and Lifschitz, S. (2015). A study of genomic data provenance in NoSQL document-oriented database systems. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 1525–1531. IEEE. 20, 25, 26

[Guo et al., 2016] Guo, M., Qian, K., and Yang, L. (2016). Hands-On Labs for Learning Mobile and NoSQL Database Security. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 606–607. 2, 14

[Haas et al., 2013] Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., Couger, M. B., Eccles, D., Li, B., Lieber, M., et al. (2013). De novo transcript sequence reconstruction from RNA-Seq using the Trinity platform for reference generation and analysis. *Nature protocols*, 8(8):1494–1512. 6, 27

[Han et al., 2011] Han, J., Haihong, E., Le, G., and Du, J. (2011). Survey on NoSQL database. In *Pervasive computing and applications (ICPCA), 2011 6th International Conference on*, pages 363–366. IEEE. xi, 13

[Harrison, 2015] Harrison, G. (2015). *Next Generation Databases: NoSQL, NewSQL, and Big Data*. Apress; 1st ed. edition (December 14, 2015), first edition. ISBN-10: 1484213300. xi, 33

[Haughian et al., 2016] Haughian, G., Osman, R., and Knottenbelt, W. J. (2016). Benchmarking Replication in Cassandra and MongoDB NoSQL Datastores. In *International Conference on Database and Expert Systems Applications*, pages 152–166. Springer. 25, 26

[Hewitt, 2010] Hewitt, E. (2010). *Cassandra: the definitive guide*. "O'Reilly Media, Inc.". 16, 17, 18

[Hoffmann et al., 2015] Hoffmann, S., Otto, C., Kurtz, S., Langenberger, D., Sharma, C., Vogel, J., Khaitovitch, P., F. Stadler, P., and Hackermueller, J. (2015). Segemehl. http://www.bioinf.uni-leipzig.de/Software/segemehl/. 5

[Holt et al., 2015] Holt, V., Ramage, M., Kear, K., and Heap, N. (2015). The usage of best practices and procedures in the database community. *Information Systems*, 49:163–181. 2, 14

[Huacarpuma et al., 2011] Huacarpuma, R. C., Holanda, M., and Walter, M. E. (2011). A conceptual model for transcriptome high-throughput sequencing pipeline. In *Advances in Bioinformatics and Computational Biology*, pages 71–74. Springer. 4, 24, 26

[Jain, 2017] Jain, V. (2017). *Big Data and Hadoop*. ISBN: 9789382609131, Disponível em: https://books.google.com.br/books?id=i6NODQAAQBAJ. 17

[Kent et al., 2002] Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., and Haussler, D. (2002). The human genome browser at UCSC. *Genome research*, 12(6):996–1006. 10

[Klein et al., 2015] Klein, J., Gorton, I., Ernst, N., Donohoe, P., Pham, K., and Matser, C. (2015). Performance evaluation of NoSQL databases: a case study. In *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, pages 5–10. ACM. 25, 26

[Krishnan, 2013] Krishnan, K. (2013). *Data warehousing in the age of big data*. Newnes. 14

[Lakshman and Malik, 2010] Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40. 14, 15, 16, 18

[Langmead et al., 2010] Langmead, B., Hansen, K. D., Leek, J. T., et al. (2010). Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome Biol*, 11(8):R83. 7

[Langmead et al., 2009a] Langmead, B., Schatz, M. C., Lin, J., Pop, M., and Salzberg, S. L. (2009a). Searching for SNPs with cloud computing. *Genome Biol*, 10(11):R134. 7

[Langmead et al., 2009b] Langmead, B., Trapnell, C., Pop, M., Salzberg, S. L., et al. (2009b). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biol*, 10(3):R25. 5, 7, 28

[Latgé, 1999] Latgé, J.-P. (1999). Aspergillus fumigatus and aspergillosis. *Clinical microbiology reviews*, 12(2):310–350. 27

[Leavitt, 2010] Leavitt, N. (2010). Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14. 11

[Leipzig, 2016] Leipzig, J. (2016). A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*. 4

[Lerner, 2010] Lerner, R. M. (2010). At the forge: Cassandra. *Linux Journal*, page 7. 14

[Li et al., 2009] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., et al. (2009). The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079. 9

[Lifschitz, 2006] Lifschitz, S. (2006). Algumas Pesquisas em Bancos de Dados e Bioinformática. In *XXVI Congresso da Sociedade Brasileira da Computação. Campo Grande*. 24

[Lipman and Pearson, 1985] Lipman, D. J. and Pearson, W. R. (1985). Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441. 8

[Loukopoulos and Ahmad, 2004] Loukopoulos, T. and Ahmad, I. (2004). Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285. 21, 22

[Marioni et al., 2008] Marioni, J. C., Mason, C. E., Mane, S. M., Stephens, M., and Gilad, Y. (2008). RNA-Seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517. 27

[Mattoso et al., 2014] Mattoso, M., Dias, J., Costa, F., de Oliveira, D., and Ogasawara, E. (2014). Experiences in using provenance to optimize the parallel execution of scientific workflows steered by users. 1, 4

[Mohamed et al., 2014] Mohamed, M. A., Altrafi, O. G., and Ismail, M. O. (2014). Relational vs. NoSQL databases: A survey. *International Journal of Computer and Information Technology*, 3(03):598–601. 2, 10, 11

[Mossucca et al., 2012] Mossucca, L., Terzo, O., Goga, K., Acquaviva, A., Abate, F., and Provenzano, R. (2012). Ngs workflow optimization using a hybrid cloud infrastructure. *International Journal on Advances in Networks and Services Volume 5, Number 3 & 4, 2012*. 6

[Neubauer, 2010] Neubauer, P. (2010). Graph databases, nosql and neo4j. *Retrieved*, 12(10):2014. 14

[Osman and Piazzolla, 2014] Osman, R. and Piazzolla, P. (2014). Modelling replication in NoSQL datastores. In *International Conference on Quantitative Evaluation of Systems*, pages 194–209. Springer. 2, 25, 26

[Özsu and Valduriez, 2011] Özsu, M. T. and Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media. 23

[Pankhudi Swaroop, 2016] Pankhudi Swaroop, Vijit Gupta, K. R. S. S. N. R. (2016). NoSQL Paradigm and Performance Evaluation. In *Scientific Society of Advanced Research and Social Change*, volume 3. SSARSC International Journal of Geo Science and Geo Informatics. ISSN 2348-6198. 2, 11, 13, 21

[Pinheiro et al., 2014] Pinheiro, R., Aires, B., Araujo, A. F., Holanda, M., Walter, M. E., and Lifschitz, S. (2014). Storing provenance data of genome project workflows using graph database. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 16–22. 5

[Pourabbas, 2014] Pourabbas, E. (2014). *Geographical Information Systems: Trends and Technologies*. A science publishers book. Taylor & Francis. 2, 13

[Robinson and Oshlack, 2010] Robinson, M. D. and Oshlack, A. (2010). A scaling normalization method for differential expression analysis of RNA-Seq data. *Genome biology*, 11(3):1. 27

[Röhm and Blakeley, 2009] Röhm, U. and Blakeley, J. (2009). Data management for high-throughput genomics. In *CIDR*, page 97–111. 24, 26

[Saldanha et al., 2012] Saldanha, H., Araújo, A., Borges, C., Ribeiro, E., Setubal, J. C., Walter, M. E., Holanda, M., Gallon, R., and Togawa, R. (2012). *Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows*. INTECH Open Access Publisher. 28

[Schmieder and Edwards, 2011] Schmieder, R. and Edwards, R. (2011). Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, 27(6):863–864. 5, 6, 7, 28

[Stonebraker, 2010] Stonebraker, M. (2010). SQL Databases V. NoSQL Databases. *Commun. ACM*, 53(4):10–11. 1, 10, 11

[Swaminathan, 2015] Swaminathan, S. N. (2015). *Quantitative Analysis of Scalable NOSQL Databases*. PhD thesis, UNIVERSITY OF TEXAS AT ARLINGTON. 11, 23, 24

[Team, 2014] Team, R. C. (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, 2012. 6

[Trapnell et al., 2009] Trapnell, C., Pachter, L., and Salzberg, S. L. (2009). Tophat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1111. 5, 7, 28

[Trapnell et al., 2012] Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D. R., Pimentel, H., Salzberg, S. L., Rinn, J. L., and Pachter, L. (2012). Differential gene and transcript expression analysis of RNA-Seq experiments with TopHat and Cufflinks. *Nature Protocols*, 7(3):562–578. 7

[Vieira et al., 2012] Vieira, M. R., Figueiredo, J. M. d., Liberatti, G., and Viebrantz, A. F. M. (2012). Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de Big Data. *Simpósio Brasileiro de Bancos de Dados*. 10

[Wiesmann et al., 2000] Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., and Alonso, G. (2000). Understanding replication in databases and distributed systems. In *Distributed Computing Systems, 2000. Proceedings. 20th International Conference on*, pages 464–474. IEEE. 22, 23

[Ye and Li, 2011] Ye, Z. and Li, S. (2011). A request skew aware heterogeneous distributed storage system based on Cassandra. In *Computer and Management (CAMAN), 2011 International Conference on*, pages 1–5. IEEE. 25, 26