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

Análise de Modelos de Dados para NoSQL Baseados em Documento em Workflows de Bioinformática

Ingrid Santana Lopes

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Brasília
2018

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

Análise de Modelos de Dados para NoSQL Baseados em Documento em Workflows de Bioinformática

Ingrid Santana Lopes

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

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

Prof.a Dr.a Edna Dias Canedo
Universidade de Brasília

Prof. Dr. Waldeyr Mendes Cordeiro da Silva
Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 28 de novembro de 2018

Dedicatória

Dedico esse trabalho aos meus pais, Verônica e Suêle, e ao meu irmão Yuri que sempre apoiaram-me e ajudaram-me a alcançar meus objetivos permanecendo ao meu lado mesmo nas horas mais estressantes.

Agradecimentos

Não foi nada fácil chegar até aqui. Foram muitos anos de momentos bons e ruins que foram compartilhados com outros. Eu nunca poderia ter cumprido essa longa jornada e chegado até aqui se estivesse sozinha. Sendo assim, ao chegar à reta final dessa minha maior aventura até então que foi a graduação, algumas pessoas devem ser citadas. Uma jornada tão difícil fez com que em minha mente por vezes surgisse a frase de Walt Disney, *"Se você pode sonhar, você pode fazer. Sempre se lembre que tudo isso começou com um rato e um sonho"*.

Dessa forma, primeiramente, agradeço a minha família. Minha mãe Verônica, meu pai Suêle e meu irmão mais novo Yuri. Aqueles que usaram de todo artifício de suporte e apoio que, aliado a preocupação inerente da família, ajudaram-me a passar pelos momentos mais difícieis da graduação. Obrigada por todos os desdobramentos para fazer com que eu chegasse na aula a tempo, chegasse em casa bem e por tentarem de tudo para aliviar todo o estresse, tristeza e desespero que às vezes me encontrei. Mas não só amenizaram o estresse, como estavam lá para celebrar cada vitória.

Não menos importantes, agradeço aos meus cachorros Shiro, Kuro e Lilo que, de maneiras tão inocentes e puras, conseguiram aliviar meu coração ansioso e desesperado com um simples olhar, brincadeira ou carinho.

Esses longos anos de UnB trouxeram amizades que gostaria que se tornassem eternas. Assim, primeiramente quero agradecer à Marcos Paulo e Rennê. Amigos inseparáveis juntos em praticamente todas as provas e trabalhos. Obrigada por todos os desesperos e dificuldades compartilhados, pelas conversas animadas, pelos trabalhos em grupo, pelas caronas divertidas e por entrarem em minha vida.

Apesar de a vida na UnB ter trazido grandes novas amizades, algumas amizades antigas se mostraram eternas e de grande importância nessa longa trajetória. Sendo assim, agradeço também aos meus queridos amigos Izabella, Matheus e Danielle. Amigos para uma vida cujo suporte e preocupação nos momentos ruins e alegria nos momentos bons ajudaram a chegar até aqui. Obrigada pelas conversas importantes e também as idiotas, pelas saídas tão difíceis de serem marcadas e principalmente por continuarem ao meu lado depois de mais de 8 anos.

Agradeço também aos psicólogos Elizabeth Takahashi e Hélio Borges cujo apoio e auxílio fizeram com que eu aguentasse essa jornada.

Devo incluir também nos agradecimentos minhas amigas Rhana e Natália, a Sensei Camila Nishikawa e a todos mais do curso de japonês da Associação Cultural Nipo-Brasileira de Brasília em Taguatinga cuja presença e alegria nas aulas de japonês todo sábado fizeram com que eu criasse forças, animo e oportunidade para relaxar e estravassar para poder aguentar cada nova semana na vida universitária.

Um agradecimento especial à todos meus professores desde do primeiro que tive quando ainda era uma pequena criança. Afinal, a jornada de aprendizado é como uma escada e todos eles foram importantes para com que eu conseguisse chegar até aqui. Agradeço especialmente ao Professor Doutor Guilherme Ramos, meu primeiro professor de computação na UnB, por despertar o interesse inicial no curso de Engenharia de Computação fazendo-me ter certeza da escolha do curso.

Agradeço especialmente também a Professora Doutora Maristela Holanda por todo o auxílio, paciência, dedicação e orientação nessa reta final e também por inspirar o interesse pela área de Bancos de Dados.

Sem mais delongas, agradeço à todos que ajudaram das mais diversas maneiras à tornar essa jornada mais fácil e possível. Gostaria de finalizar citando o pensamento final de uma das obras da autora Hiromu Arakawa, *"Não se pode aprender algo de uma lição sem dor, já que não se pode ganhar algo sem algum sacrifício. Mas, quando essa dor é superada e a lição, incorporada, o resultado é um coração infalível, um coração como aço."*

Resumo

Para a quantidade crescente de dados gerados por várias áreas do conhecimento dá-se o nome de *Big Data*. Neste cenário, pode-se dizer que as pesquisas de bioinformática necessitam de dados de proveniência, pois estes são capazes de fornecer o histórico das informações coletadas no *workflow* da pesquisa e responder questões relacionadas à origem dos dados.*Big Data* trouxe o surgimento da abordagem NoSQL (*Not Only* SQL) como uma alternativa ao uso de Modelos de Banco de Dados Relacional por não apresentar as limitações observadas no Modelo de Banco de Dados Relacional quando este é aplicado em uma grande quantidade de dados. Com foco no MongoDB, este trabalho propõe, com o auxílio de um programa, criado capaz de executar automaticamente um *workflow*, armazenar sua proveniência e dados brutos em três diferentes formatos de documentos: referencial, embutido e híbrido. Essas três maneiras diferentes são comparadas e analisadas usando parâmetros como tempo e recursos de consulta. Os resultados mostraram algumas particularidades da bioinformática e vantagens ou desvantagens para cada modelo.

Palavras-chave: NoSQL, MongoDB, Bioinformática, Proveniência, Big Data, Workflow

Abstract

The increasing amount of data named generated by several areas of knowledge is named Big Data. In this scenary, it can be said that Bioinformatic researchs needs provenance data, since it is capable of providing the history of the information collected in the research workflow and answer questions related to the data source. Big Data brought the emergence of the NoSQL (Not Only SQL) approach as an alternative to the use of Relational Database Models because it does not present the limitations observed in the Relational Database Model when it is applied in a large dataset. With focus on MongoDB, this work proposes a program that can automatically execute a workflow and store its provenance and raw data into three different document formats: reference, embedded and hybrid. Those three different ways are compared using parameters such as time and query capabilities. Results showed some bioinformatics particularities and advantages or disadvantages for each model.

Keywords: NoSQL, MongoDB, Bioinformatic, Provenance, Big Data, Workflow

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACID *Atomicity*, *Consistency*, *Isolation*, *Durability*.

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

BSON *Binary JSON*.

CAP *Consistence*, *AvailabilityPartition Tolerance*.

DDoS *Distributed Denial of Service*.

HD *Hard Drive*.

JSON *JavaScript Object Notation*.

NGS *Next Generation Sequencing*.

NoSQL *Not Only SQL*.

RAM *Random Access Memory*.

RDBM *Relational Database Model*.

SQL *Structured Query Language*.

XML *eXtensible Markup Language*.

Capítulo 1

Introdução

Com o avanço da tecnologia, aumentou a necessidade de bancos de dados capazes de armazenar, processar de forma eficaz, escrever e ler com alto desempenho uma grande quantidade de dados [1]. Foi nesse cenário que nasceu o conceito de *Big Data* como um termo usado para descrever volumes massivos de dados, estruturados ou não, que podem ser gerenciados com Bancos de Dados NoSQL [2]. O NoSQL tem como objetivo trabalhar os dados de forma que eles se tornem informação, conhecimento e então, posteriormente, sabedoria [3].

NoSQL está relacionado à ideia de gerenciamento não relacional para armazenagem de dados em larga escala [4, 5]. Tal modelo de gerenciamento é altamente escalonável, confiável e possui um modelo de dados simples além de vantagens quando comparados ao modelo relacional [6]. Ademais, pode ser classificado e definido de acordo com seu modelo de dados, dentre alguns: Chave-Valor, Documento, Coluna e Grafo [7].

O MongoDB é um sistema de armazenamento de banco de dados NoSQL baseado em documento capaz de prover recursos de consulta relativamente poderosos, atomicidade em nível de documento e suporte para tipos de dados complexos [1, 8]. Suas estruturas de dados podem ser organizadas de duas formas: embutida e referenciada. Contudo, existe a possibilidade de criar um modelo híbrido que utilize de características de ambas [9].

A proveniência, por sua vez, pode ser definida como um perfil do histórico de execução capaz de responder questões relacionadas à origem dos dados [10]. Sua teoria foi tradicionalmente aplicada em contextos de *workflows* [11] e sua utilidade reside no fato de que, uma vez que a proveniência é registrada, é mais fácil criar, recriar, avaliar ou modificar os modelos computacionais (ou experiências científicas) com base no acúmulo de conhecimento do que foi feito [12].

Considerando o *Big Data* no contexto da bioinformática, nota-se que a maior parte dos dados gerados são oriundos do campo da biologia molecular [3]. Essa grande quantidade de dados é consequência da revolução causada pelo NGS (*Next Generation Sequencing*)

que representou um grande salto na quantidade de dados gerados pela biologia e medicina molecular [13, 14].

As pesquisas de bioinformática necessitam de dados de proveniência devido à sua capacidade de fornecer o histórico das informações coletadas no decorrer da execução de um *workflow* e de responder questões relacionadas à origem dos dados. Nesse contexto, ressalta-se a importância de utilizar um modelo de armazanagem de dados capaz de atender as demandas da área de forma eficiente contendo as informações de proveniência e os dados brutos dos arquivos participantes do *worflow* do experimento. Uma vez que é possível propor diversos modelos de dados, uma análise da eficiência destes se mostra fundamental.

1.1 Objetivo

O objetivo desse trabalho é analisar diferentes implementações de modelos de dados para proveniência de dados de *workflows* de Bioinformática em NoSQL baseado em documento. De forma à alcançar tal propósito, as seguintes metas foram definidas:

1. Executar o *workflow* diretamente pelo terminal.
2. Implementar programa capaz de processar o *workflow* por conta própria e armazenar suas informações.
3. Salvar informações no MongoDB em três modelos diferentes.
4. Realizar comparações entre os três modelos.
5. Analisar e comparar os resultados obtidos.

1.2 Metodologia de pesquisa

Os métodos de pesquisa utilizados incluiram a implementação de um programa capaz de executar as atividades de um *workflow* de bioinformática e, posteriormente, armazenar os dados de proveniência e os dados brutos no MongoDB. Esses dados foram inseridos seguindo três variações de modelos de dados utilizando os conceitos de documentos referenciados, embutidos e híbridos. O tempo de inserção dos dados no MongoDB foi anotado e usado para comparação de qual variação foi mais eficiente. Além disso, buscas foram executadas sobre cada um dos modelos de forma à considerar o grau de dificuldade destas e as respectivas capacidades de busca de cada modelo. Tais modelos e os resultados obtidos foram avaliados de forma discursiva e comparativa.

1.3 Resultados

Esse trabalho apresenta como resultado a avaliação da performance de modelos de dados baseados em documento com base em *workflows* de Bioinformática implementando o conceito de proveniência de dados. Seus resultados incluem observações do tempo de inserção dos dados de proveniência e dados brutos no MongDB e capacidades de buscas para cada variação do modelo.

1.4 Estrutura do Trabalho

Esta monografia está composta pelos seguintes capítulos:

- Capítulo 2: Referencial Teórico
  - Bancos de Dados NoSQL
  - Bancos de Dados baseado em Documento
  - Computação em Nuvem
  - Proveniência de Dados
  - Bioinformática
  - Trabalhos Relacionados
- Capítulo 3: Execução
  - Ambiente de Execução
  - Programa Executável
  - *Workflows* Executados
- Capítulo 4: Resultados
  - Ambiente Local
  - Ambiente de Nuvem
  - Comparação de armazenagens
  - Resultados Acadêmicos
- Capítulo 5: Conclusão

Capítulo 2

Referencial Teórico

Este capítulo é destinado para a apresentação dos conceitos fundamentais para o desenvolvimento deste trabalho. O capítulo é dividido nas seguintes seções: a Seção 2.1 trata sobre os conceitos gerais de Bancos de Dados NoSQL; a Seção 2.2 apresenta os conceitos relacionados especificamente aos Bancos de Dados NoSQL Baseados em Documento; A Seção 2.3 trata sobre a Computação em Nuvem; A Seção 2.4 discussa sobre proveniência de dados; a Seção 2.5 trata da Bioinformática e, por fim, a Seção 2.6 mostra um resumo dos trabalhos relacionados a esta dissertação.

2.1 Bancos de Dados NoSQL

2.1.1 Big Data

O passar dos anos e os avanços da ciência trouxeram ao mundo novas tecnologias que geram uma quantidade cada vez maior de dados. Tais dados possuem necessidade de serem armazenados, processados de forma eficaz e escritos e lidos com alto desempenho [1].

Até 2003, 5 exabytes foram criados por humanos enquanto em 2012 esse número subiu para 2,72 zettabytes e espera-se que aumente para 35,2 zettabytes em 2020 [3]. Dentre as várias áreas que produzem uma quantidade cada vez maior de dados, é interessante ressaltar o campo da biologia molecular que, na última década, tem dobrado a quantidade de sequências produzidas a cada sete meses [15]. Sendo os dados da biologia molecular gerados apartir do sequênciamento de DNA, é estimado que a capacidade de sequenciamento mundial cresça dos 35 petabases por ano, consideradas em 2015, para 1 exabyte por ano em 2020 e para mais de 1 zettabase por ano até 2025 [15].

Os avanços recentes do mundo digital mostram-se cada vez mais rápidos, completos, variáveis e volumosos [4] criando então o fênomeno conhecido como *Big Data*. Todavia, esse acontecimento vai mais longe ainda, podendo estar presente nos mais variados for-

matos de dado como mapas, áudio, vídeos, fotos [16]. Devido à grande escalabilidade, diversidade e complexidade, os dados categorizados como *Big Data* necessitam de novas arquiteturas, análises e mineirações de forma a extrair conhecimento [3, 13]. Além disso, eles são capazes de cobrir várias perspectivas, tópicos e escalas [16] e são importantes no âmbito da Computação em Nuvem já que milhões de pessoas usam serviços de nuvem para realizar as mais diversas operações sobre os dados coletados [2].

Multidisciplinar, o uso de *Big Data* está presente em diferentes campos como biomedicina [13], geografia [16] e saúde [17]. Devido ao vasto leque de áreas que fazem uso do *Big Data*, o processo de obtenção de conhecimento por meio de dados ganha mais importância ao ser possível aprimorar a detecção do valor dos dados de *Big Data* ao se utilizar de algumas técnicas e tecnologias, como Hadoop, MapReduce e NoSQL para trabalhar os dados de forma que eles se tornem informação, conhecimento e então, posteriormente, sabedoria [3]. Além disso, a capacidade de lidar com *Big Data* está se tornando um facilitador para realizar estudos de pesquisa sem precedentes e implementar novos modelos de prestação de serviços [13].

2.1.2 O NoSQL

O surgimento do *Big Data* criou um cenário na tecnologia onde os bancos de dados possuem necessidades diferentes das anteriormente percebidas [18]. Graças a tal mudança, as maiores demandas se tornaram por [1]:

- Maior leitura e escrita em concorrência com menor latência,
- Acesso e armazenamento de *Big Data* eficientes,
- Maior escalabilidade e disponibilidade,
- Menor custo operacional e de manutenção.

Em contra partida, embora os tradicionais modelos relacionais tenham ocupado uma alta posição na área de armazenamento de dados, ao serem confrontados com tais demandas, eles demonstraram os seguintes problemas [1]:

- Leitura e escrita lentas,
- Capacidade limitada,
- Dificuldade de expansão.

Com o intúito de resolver os desafios criados pelas novas demandas, nasceu o conceito de Bancos de Dados NoSQL [1, 2].

NoSQL indica o gerenciamento de um sistema de banco de dados não relacional criado para armazenagem de dados em larga escala e processamento paralelo de dados [4, 5]. Tal dado, eclético e não estruturado, possui uma interface de uso bem simples [4, 19], são flexíveis [9] e diferem na forma na qual seus modelos de dados são fornecidos quando em comparação com o modelo relacional e também na riqueza das funcionalidades de busca fornecidas [20].

O termo NoSQL foi usado pela primeira vez em 1998 e então reutilizado em 2009. Nessa primeira aparição, foi utilizado para se referenciar um banco de dados relacional que omitia o uso do SQL. Na segunda oportunidade, no entanto, já foi referenciado à bancos de dados não relacionais [18].

Além disso, as novas tecnologias de bancos de dados estão forcenendo respostas para problemas de escalabilidade apresentando novas soluções usando o NoSQL. Tal mudança de tecnologia, além de se mostrar capaz de construir novos bancos de dados e sistemas de armazenamentos de dados, também é capaz de coletar volumes grandes e variáveis em ambientes distribuídos [13]. Ademais, bancos de dados NoSQL oferecem flexibilidade para o modelo de dados [9].

2.1.3 Modelo Relacional x Não Relacional

Um banco de dados relacional é uma estrutura de dados no formato de tabelas que podem ser vinculadas entre si por meio de chaves estrangeiras. Cada tabela deve conter uma chave primária que possui como finalidade identificar exclusivamente qualquer parte atômica de dados dentro do intervalo. Essa chave também pode ser usada por outras tabelas para criar um relacionamento com a tabela em questão [21]. Visando garantir a integridade de dados, os modelos relacionais utilizam o conceito ACID (Atomicidade, Consistência, Isolamento e Durabilidade) em suas estruturas e transações. As propriedades ACID são importantes para que se possa garantir a segurança das transações e, para cada uma delas, a atomicidade e a consistência de dados [4, 22].

Anteriormente, independente de se o modelo de dados não correspondesse ao modelo relacional muito bem, os bancos de dados relacionais foram usados para quase todo problema de armazenamento. A consequência disso foi o aumento da complexidade devido ao uso de *frameworks* de mapeamento caros e algoritmos complexos [19]. Além do que, ao tentar garantir todos os atributos do teorema das propriedades ACID, em algumas situações, pode-se ocasionar configurações complexas e mal desempenho em transações e consultas ao banco de dados [22]. O conjunto de recursos oferecidos por bancos de dados SQL também pode se tornar uma sobrecarga desnecessária para determinadas tarefas [19].

Os bancos de dados não-relacionais (NoSQL), por sua vez, são altamente escalonáveis, confiáveis, possuem um modelo de dados simples e vantagens importantes como a possi-

blidade de manipular dados não-estruturados como documentos do Word, e-mails, áudio ou vídeo [6]. Os bancos de dados NoSQL geralmente não aderem às restrições ACID e, na verdade, se voltam para o paradigma BASE [6, 23]. O paradigma BASE nasceu da ideia do teorema CAP, proposto por [24], que tem como ponto central o fato de que um sistema poder garantir plenamente apenas dois das três propriedades listadas, enquanto a terceira é fracamente garantida [1, 24]. Tal conceito pode ser melhor visualizado em Figura 2.1. Ademais, sabe-se que as propriedades demonstradas na Figura 2.1 podem ser descritas como:

- Consistência: Todos os nós da rede devem possuir a mesma versão do arquivo;
- Disponibilidade: Todos os clientes podem sempre encontrar pelo menos uma cópia dos dados solicitados, mesmo que um cluster esteja desligado;
- Tolerância à Partições: Mesmo se o sistema estiver implantado em diferentes servidores transparentes para o cliente, ele continua com seus dados, propriedades e características.

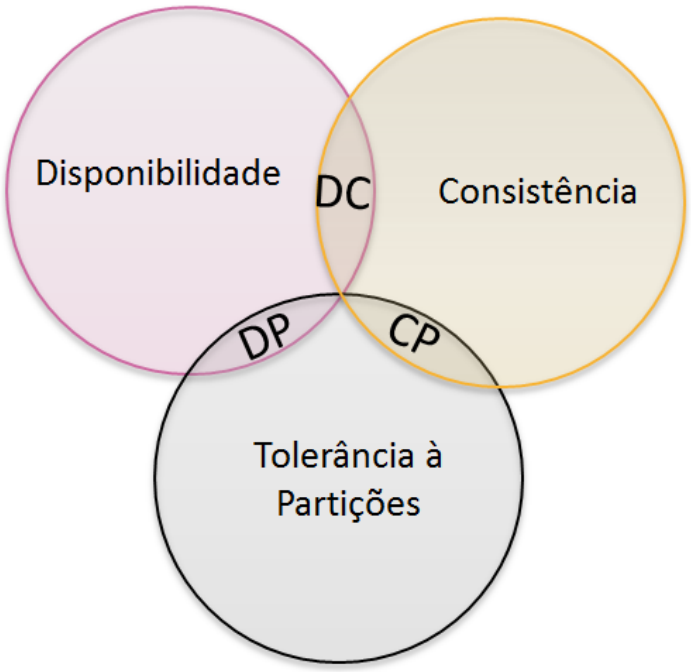

Figura 2.1: Possibilidades do Teorema CAP (Fonte: [1]).

Observando o caso de um sistema distribuído, ao ser proposto o enfraquecimento da propriedade de consistência de dados focando-se em melhorar a disponibilidade e a tolerância a criar mais partições, criou-se o BASE. Com replicação otimista e tolerância

a falhas de consistência, o modelo BASE foi a essência usada para a criação de diversos tipos de bancos de dados não relacionais graças à sua filosofia [22].

Entretanto, os bancos de dados relacionais também possuem falhas e limitações. Algumas que podem ser citadas são: a falta de criptografia, autenticação fraca entre o cliente e os servidores, vulnerabilidade a ataques de injeção SQL ou DDoS (negação de serviço) e outros. Alguns sistemas gerenciadores de bancos de dados, no entanto, fornecem implementações que tentam resolver esses problemas [6].

2.1.4 Modelos de Dados NoSQL

Os Bancos de Dados NoSQL podem ser classificados de acordo com seu modelo de dados, as opções são [8, 1, 19, 9, 7]:

- *Chave-Valor*: estruturas de dados tornadas acessíveis através de uma chave única. Como os valores estão isolados e são vetores de *bytes* não interpretados, completamente opacos ao sistema e independentes entre si, chaves se tornam a única forma de recuperar os dados. Por conseguinte, os relacionamentos devem ser tratados na lógica da aplicação. Bancos de dados chave-valor possuem dados simples, novos valores podem ser adicionados em tempo de execução e são úteis para operações simples.

- *Documento*: Parecido com chave-valor, o banco de dados baseado em documento armazena os dados em documentos que possuem chaves únicas dentro de sua coleção, mas seus valores são em formatos como JSON ou XML. Ao contrário de chave-valor, seus valores não são opacos ao sistema e podem ser buscados facilmente. Portanto, estruturas de dados complexas podem ser manuseados de forma mais conveniente e prática. Bancos de Dados do tipo documento não possuem restrições de esquema e adicionar novos documentos contendo quaisquer atributos é uma operação simples de ser feita em tempo de execução assim como adicionar novos atributos aos documentos já existentes. Os bancos de dados baseados em documentos podem ter alguns modelos de dados construídos com documentos embutidos e outros feitos com documentos referenciados. Esse tipo de sistema é muito conveniente em integrações de dados e tarefas de migração de esquemas, além de ser fácil de manter e bem flexível.

- *Coluna*: Os bancos de dados orientados a coluna são inspirados no *Google Bigtable*, um sistema de armazenamento distribuído capaz de gerenciar dados estruturados projetados para serem dimencionados para um tamanho muito grande. Como os dados não podem ser interpretados pelo sistema, relacionamentos entre os conjuntos

de dados e outros tipos de dados não são suportados. Tais dados são categorizados através da colunas, famílias de colunas ou *timestamps* usando uma tabela como modelo de dados e novas linhas e colunas podem ser adicionadas flexivelmente durante o tempo de execução. Contudo, as famílias de colunas devem ser frequentemente pré-definidas. Tal restrição leva à menos flexibilidade do que a fornecida por bancos de dados chave-valor ou documento.

- *Grafo*: Em contraste com os modelos anteriores, um banco de dados orientado à grafo utiliza o conceito matemático de grafos como o modelo de seus dados através do uso de nós e arestas que os interconectam. Especializado em gerenciamento de dados fortemente conectados, o modelo é mais adequado às aplicações com muitos relacionamentos. Nós e arestas consistem em objetos com pares de valores e chaves incorporados sendo que é possível definir que uma aresta específica é aplicável apenas entre certos tipos de nós. Ademais, os dados podem ser armazenados tanto em arestas quanto em nós.

2.2 Banco de Dados baseado em Documento

Devido a facilidade de uso, os bancos de dado NoSQL baseados em documento são os mais populares sendo o MongoDB o sistema de gerenciamento central principal para esse modelo além de ser o mais frequentemente usado e proeminente [9, 19].

2.2.1 MongoDB

O MongoDB vem sendo desenvolvido desde 2009 [2] e é um sistema de armazenamento de banco de dados não-relacional baseado em documento e que utiliza replicação mestre-escravo. Ele é capaz de prover recursos de consulta relativamente poderosos, atomicidade no nível do documento e suporte para tipos de dados complexos [8, 1]. Sua filosofia está focada em combinar as capacidades críticas de bancos de dados relacionais com as inovações das tecnologias NoSQL [25] e foi desenvolvido visando a armazenagem da informação com alta performance e escalabilidade [9]. Consequentemente, está sendo usado em muitos projetos com dados crescentes em vez de bancos de dados relacionais [1]. Ele também possui versões que são compatíveis com diferentes edições de sistemas operacionais Windows, Linux, Solaris e Mac e é uma aplicação *open source*. Ademais, atualmente é a aplicação mais popular na categoria de bancos de dados orientados a documentos [2, 9]. A popularidade do MongoDB pode também ser vista em Figura 2.2.

Seus dados são armazenados como documentos BSON, visto na Figura 2.3, uma extensão da representação JSON [25]. Esse tipo de documento fornece funcionalidade recursiva

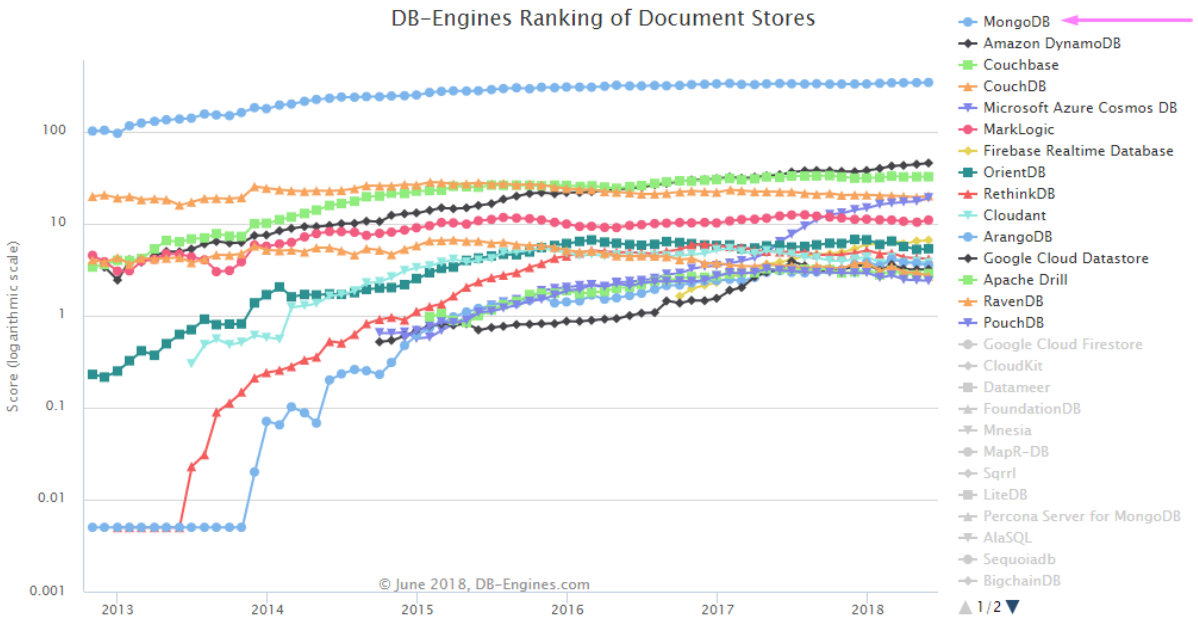

Figura 2.2: Bancos de Dados Baseados em Documento mais utilizados (Fonte: [26]).

e permite uma análise de banco de dados mais eficiente [2]. Tais documentos são armazenados em coleções, tal como mostrado na Figura 2.4, que são guardadas no banco de dados [27].

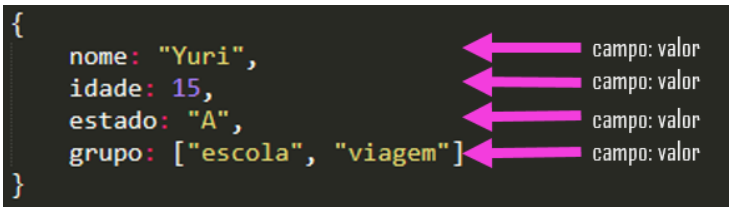

Figura 2.3: Exemplo de Arquivo BSON.

O MongoDB também possui uma arquitetura de armazenamento flexível sendo possível estendê-lo com novos recursos e configurá-lo para o melhor uso de arquiteturas de hardware específicas. Além de tudo, essa arquitetura permite que o banco de dados gerencie automaticamente o movimento de dados entre tecnologias de mecanismos de armazenamento com o uso de replicação nativa. Essa abordagem pode reduzir significativamente o desenvolvimento e a complexidade operacional [25]. O MongoDB possui também multiplas instâncias de bancos de dados, onde cada banco de dados pode ter várias coleções que podem conter vários documentos do tipo BSON [9]. A estrutura do MongoDB pode ser vista em Figura 2.5. Todavia, ele contém limitações como a susceptibilidade à ataques

Figura 2.4: Exemplo de Coleção.

de injeção e o armazenamento não criptografado de seus dados. Contudo, tais deficiências são comuns a bancos de dados NoSQL [6].

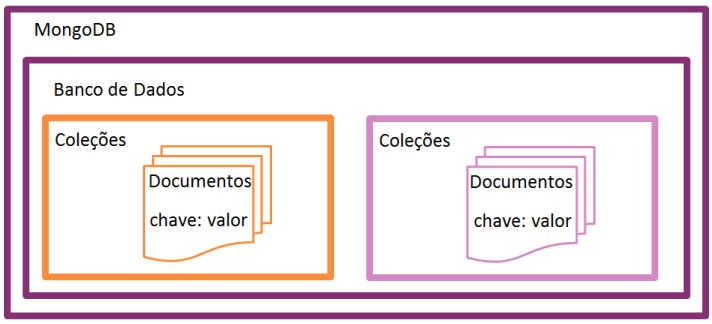

Figura 2.5: Estrutura do MongoDB.

O MongoDB possui duas maneiras primárias de organização de seus dados orientados à documento [9]:

- Embutido: Um modelo desnormalizado que armazena a estrutura através de agrupamento de dados de acordo com as suas familiaridades em um grande documento.
- Referenciado: Um modelo normalizado que trabalha com formatos padronizados que usa conexões para criar relacionamentos com outros documentos.

Também existe, no entanto, a possibilidade de criar um modelo híbrido que possui características tanto de embutido como de referencial. Fora isso, os índices podem ser usados em qualquer uma das estruturas e eles são capazes de suportar a busca quando é necesário todos os campos. Dessa forma, eles representam um aumento de perfomance da busca embora o tempo de resposta de busca seja o motivo de discusão quanto a utilização de bancos de dados orientados à documento. Embora possam ajudar nas buscas, a criação

de índices implica em gasto com performance e, por isso, costumam ser úteis em conjuntos de dados grandes [9].

Um fator que deve ser observado é que o tamanho dos documentos do MongoDB são limitados a 16 MB [25, 28]. Como alternativa para armazenar documentos maiores no MongoDB, a convenção usa o conceito de GridFS como altenativa [29]. Com o GridFS, é possível inserir documentos maiores que 16MB por meio da divisão do arquivo em partes denominadas *chunks*. Tais partes, são fragmentos geralmente de 255kB criados pelo GridFS [28]. Os dados da chave de identificação do *chunk* inicial de cada arquivo é inserido no documento *dataFile*. Todos os *drivers* oficiais do MongoDB suportam tal convenção [29] que separa os arquivos em duas coleções [28]:

- *chunks* : responsável por armazenar as partições binárias.
- *files* : responsável por armazenar os metadados do arquivo.

Ambas as coleções são guardadas em um *bucket* que, embora possa ser nomeado como o usuário quiser, é comumente denominado *fs* [28].

2.2.2 *MongoDB C Driver*

O *MongoDB C Driver* fornece a base para *drivers* MongoDB em liguagem de alto nível sendo também a biblioteca cliente oficial para aplicativos em C. Seu projeto inclui tanto a biblioteca "libmongoc", como também a biblioteca "libbson". A primeira é uma biblioteca em C para MongoDB e a segunda fornece rotinas úteis para criação, análise e interação com documentos BSON. Ademas, o *MongoDB C Driver* é compatível com todas as principais plataformas [30, 31].

Com o *MongoDB C Driver* e suas bibliotecas, é possível criar conexão com o MongoDB, criar e deletar bancos, adicionar documentos ou coleções, alterá-las ou deleta-las, realizar operações variadades de manipulação e conversão de dados do tipo BSON, entre outros [30, 31].

2.3 Computação em Nuvem

A Computação em Nuvem, ou *Cloud Computing*, é um novo paradigma que surgiu pouco antes do Big Data [32] e que vem ganhando maior importância no meio da TI conforme se desenvolve e passa a ser aplicada em várias áreas, tais como na Biologia e no Comércio [33]. Em 2012, uma pesquisa da *Cloud Industry Forum* mostrou que 53% de 300 empresas analisadas estão usando computação em nuvem e 73% destas 300 planejam aumentar o uso de nuvens durante o período de 12 meses [33].

Capaz de fornecer elasticidade, recursos agrupados, acesso sob demanda, serviços auto-atendimento e *pay-as-you-go* [32], uma nuvem é um tipo de sistema distribuído e paralelo que consiste na coleção de interligações de computadores virtualizados. Tais computadores são provisionados e apresentados como um ou mais recursos computacionais unificados baseados em Acordos em Nível de Serviços, estabelecidos através da negociação entre o provedor e o consumidor [34]. Com isso, computação em nuvem pode ser definida como um paradigma computacional de larga escala, no qual um conjunto gerido de poder computacional, virtualizado, dinamicamente escalável, com plataformas e serviços são entregues sob demanda a consumidores externos através da Internet [35].

A plataforma de nuvem tem como objetivo fornecer infraestrutura (*Infrastructure as a Service*), plataforma (*Platform as a Service*) e software (*Software as a Service*) como serviços, sendo eles disponibilizados por meio do modelo *pay-as-you-go* [36]. Esse modelo permite que o usuário pague pelo serviço conforme o seu consumo. Na computação em nuvem, o armazenamento de dados é realizado com alta segurança e privacidade garantidas por meio de serviços fornecidos sob demanda que podem ser acessados por qualquer lugar do mundo. As nuvens podem ser tanto centralizadas como distribuídas e são baseadas em Acordos de Nível de Serviço [36].

2.3.1 *Google Cloud Plataform*

O *Google Cloud Plataform* é formado por um conjunto de artefatos espalhados por centros de dados do *Google*, tais como computadores, unidades de disco rígido, máquinas virtuais e outros. A presença de centros de dados em diversos lugares fornece uma boa distribuição de recursos capaz de conceder diversas vantagens. Entre os benefícios, pode-se salientar a latência reduzida e redundância para casos de falha [37].

2.4 Proveniência de Dados

A proveniência pode ser definida como um perfil do histórico de execução capaz de responder questões relacionadas à origem dos dados [10], fornecendo a linhagem ou o histórico de como os dados são gerados, usados, modificados e o processo pelo qual eles chegaram no banco de dados em questão [11, 38]. Quando se tem por objetivo a acurácia e a intemporalidade dos dados, esta ganha se torna importante [38].

Sua teoria foi tradicionalmente aplicada em contextos de *workflows* [11] e sua utilidade reside no fato de que, uma vez que a proveniência é registrada, é mais fácil criar, recriar, avaliar ou modificar os modelos computacionais (ou experiências científicas) com base no acúmulo de conhecimento do que foi feito [12]. Na proveniência aplicada a bancos de dados, ou rastreamento de linhagem [11], a origem dos dados que são usados como

matéria-prima de importância científica essencial e os processos que transformaram esses dados no produto final são identificados e devidamente armazenados [39].

Devido a importância da estruturação dos dados escolhida para se chegar no estágio de recuperação e compreensão destes, desenvolveram-se diversos modelos de proveniência como: Modelo W7 [40], Vocabulário de Proveniência [41], ProV-DM [42], entre outros [39].

2.4.1 PROV-DM

O PROV-DM é um modelo conceitual de armazenagem de dados de proveniência proposto pelo W3C (*World Wide Web Consortium*) cuja estrutura básica define dois elementos iniciais, ou vértices, *Atividade* e *Entidade* [39]. Tais atividades e entidades tem como propósito descrever a produção, entrega ou enunciado de determinado objeto [43]. A *Atividade* é responsável por representar o processo que indica a origem de um objeto de proveniência enquanto a *Entidade* modela qualquer objeto representado em um tipo de proveniência [39]. A *Entidade* possui dois subtipos de elementos, um deles é o *Agente* devido à sua responsabilidade de representar algo que possui uma incumbência sobre uma atividade sendo realizada, sobre a existência de uma entidade ou sobre a atividade de outro agente [43, 44], o outro é a *Coleção* pois esta representa o conjunto de atividades [39].

Os vértices possuem relações de causalidade entre si que são direcionadas do efeito para a causa [43]. As relações são ilustradas na Figura 2.6, Figura 2.7 e Figura 2.8 [43, 44]:

- *used*: Apenas entre atividades e entidades, indica que determinada atividade usou uma entidade.
- *wasGeneratedBy*: Relação presente entre entidades e atividades que visa representar que uma entidade foi gerada por uma atividade.
- *wasInformedBy*: Ligação existente entre atividades que indica uma atividade informada usou uma entidade que foi gerada pela atividade que a informou. Contudo, tal atividade ou não é de interesse ou é desconhecida.
- *wasStartedBy*: Conexão formada entre atividades e entidades responsável por assinalar a iniciação de uma entidade já criada por uma atividade.
- *wasEndedBy*: Igualmente entre atividade e entidade, visa registrar a finalização de uma entidade por uma atividade.
- *wasInvalidatedBy*: Indica que uma entidade foi tornada indisponível para uso.
- *wasRevisionOf*: Aponta a derivação de uma entidade apartir de outra de maneira corretiva.

- *wasDerivedFrom*: Aponta a derivação de uma entidade apartir de outra de maneira evolutiva.
- *wasQuotedFrom*: Indica que uma entidade foi derivada de uma outra entidade através de cópia total ou parcial.
- *hadPrimarySource*: Relação de derivação que indica a determinação de fontes primárias.
- *wasAssociatedWith*: Existente entre atividades e agentes, é encarregada de determinar a associação de uma atividade a um agente.
- *wasAttributedTo*: Presente entre entidade e agentes, indica que determinada atividade foi associada a um agente específico.
- *actedOnBehalfOf*: Uma relação de delegação que indica a atribuição de autoridade e responsabilidade a um agente para executar uma atividade específica como delegado ou representante.

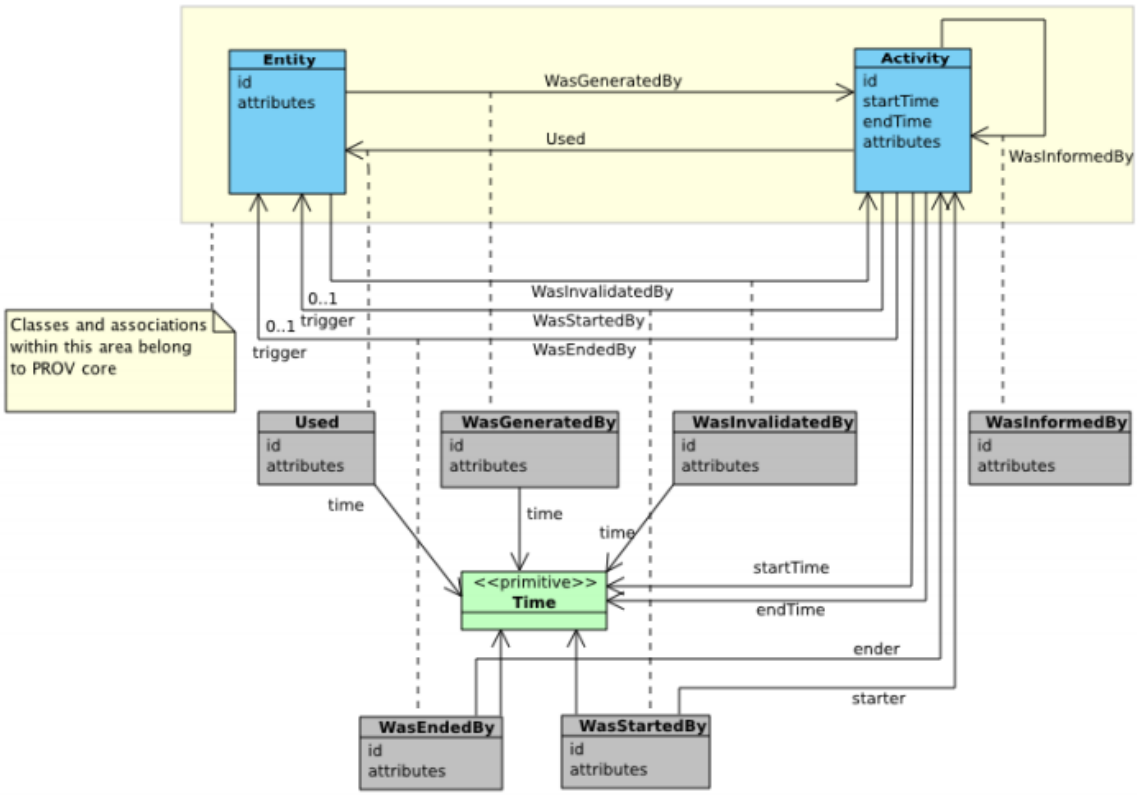

Figura 2.6: Visão geral de Atividade e Entidade (Fonte: [44]).

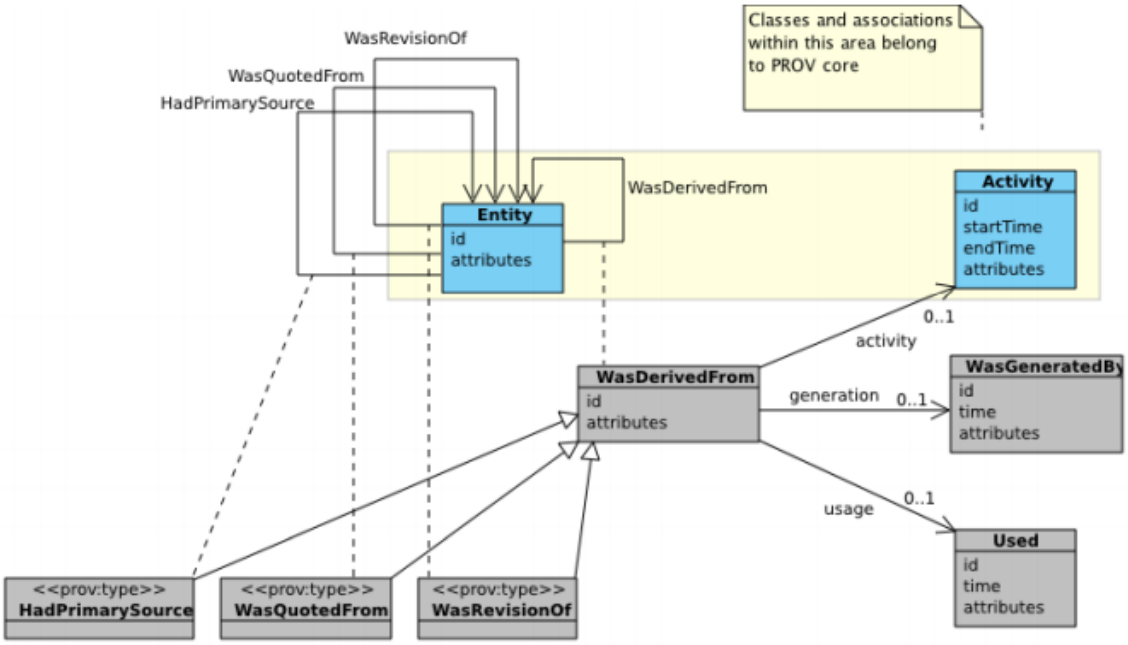

Figura 2.7: Visão geral do componente de derivação (Fonte: [44]).

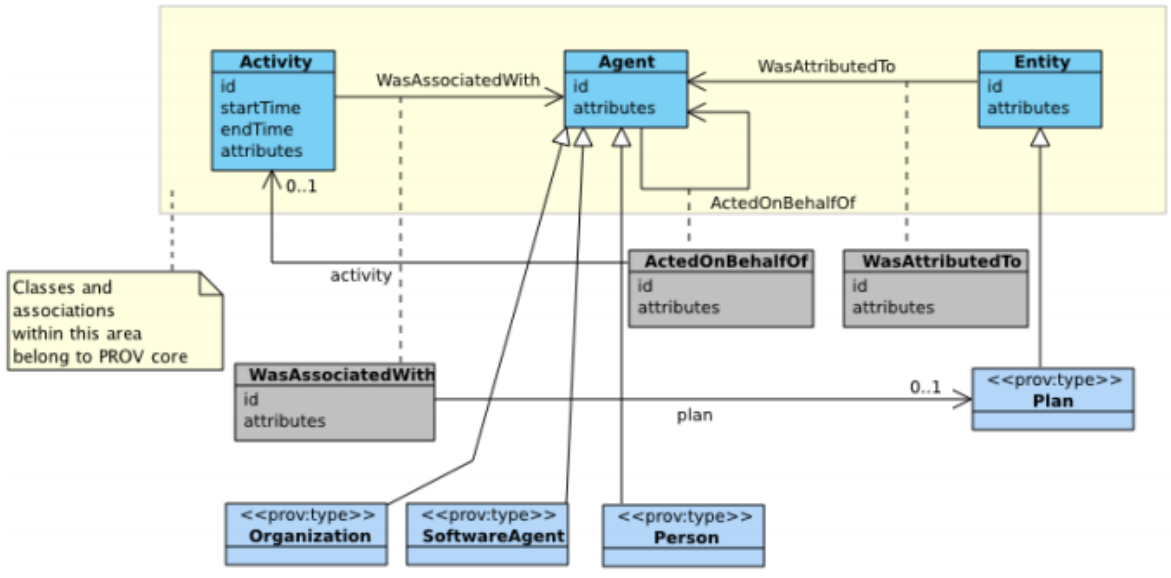

Figura 2.8: Visão Geral de Agentes e Responsabilidade (Fonte: [44]).

2.5 Bioinformática

A Bioinformática surgiu devido à geração de dados biológicos em grande quantidade que tornaram a utilização do computador indispensável para sua análise. Ela pode ser interpretada como o uso de técnicas computacionais que visam entender e organizar a

informação associada a macromoléculas biológicas [45].

A maior parte dos dados de bioinformática gerados são oriundos do campo da biologia molecular [3]. Essa grande quantidade de dados é consequência da revolução causada pelo NGS que representou um grande salto na quantidade de dados gerados pela biologia e medicina molecular. Ademais, os novos programas voltados para a área fizeram com que a NGS possuisse análise de alto rendimento [13, 14].

No campo da biologia, particularmente na biologia molecular, um banco de dados NoSQL adquire uma importância única. Tal importância está no fato de que, uma vez que a exploração e análise da estrutura do genoma e sua evolução, são essenciais para diversos avanços como a criação de novas terapias no campo da medicina [46] e para entender a adaptação de organismos [5]. Tais genomas são gerados uma grande quantidade de dados pela NGS, criando um desafio computacional em termos de armazenamento, busca, compartilhamento, análise e visualização [5, 46, 17, 13]. Ademais, com a vantagem da escalabilidade, o uso de NoSQL na bioinformática é um domínio que tem se tornado mais presente nas pesquisas e soluções [14].

A persistência do NoSQL trouxe pesquisas [47, 17, 14] que provaram que o modelo NoSQL pode atender às demandas da área melhor do que o modelo RDBM nos cenários considerados, pois o segundo costuma ter geralmente tem um desempenho pior para gerenciar e armazenar uma enorme quantidade de dados usando SQL devido às suas limitações [17]. Dentre as limitações observadas, estão a impossibilidade do banco de dados relacional em se adaptar a grande tráfego enquanto mantém o custo aceitável, a quantidade de tabelas necessárias para manter as relações do modelo e a base de dados ser dependende de um grande número de tabelas temporárias para armazenagem de resultados intermediários [14]

2.5.1 *Workflows* de Bioinformática

Um *workflow* pode ser definido como um conjunto de tarefas com informações de entrada e saída à serem seguidas para se alcançar um objetivo final [48]. Quando voltada para a bioinformática, os arquivos de entrada de um *workflow* são as sequências geradas pelo sequenciador NGS e o *workflow* pode ser dividido em fases bem definidas que fazem uso desses arquivos e também de diversos programas [39], ferramentas e bibliotecas [49]. As operações feitas no decorrer de um *workflow* de Bioinformática envolvem séries de transformações executadas das mais diversas maneiras sobre as milhões de sequências geradas pelo NGS [50].

A criação de um *workflow* para análise e interpretação de uma sequência de dados visa adquirir algum conhecimento específico resultante que seja inequívoco, consistente e repetível dadas as mesmas circunstâncias [49].

2.6 Trabalhos Relacionados

A proveniência e a sua utilidade são discutidas em Mattoso *et al* [12] e, ao considerar a aplicação de proveniência em bancos de dados NoSQL baseados em documentos de forma geral, pode-se citar diversos trabalhos que propõem conceitos e ideias diferenciadas. Tao Li *et al* [10] propuseram uma estrutura chamada *ProvenanceLens*, que fornece gerenciamento de proveniência em ambientes de nuvens. Em seguida, comparam sua performance ao usar o MySQL, o MongoDB e o Neo4J. You-Wei Cheah *et al* [11] apresentaram o Milieu, um *framework* focado na coleção de proveniência para experiências científicas e usa um armazenamento MongoDB. Ambos, [10, 11], enumeram características que são desejáveis para seus sistemas e categoriza a proveniência em tipos. Anu Mary Chacko *et al* [51], usam a idéia de Wrappers de Dados Estrangeiros em um sistema de gerenciamento de proveniência chamado PERM para implementar a proveniência usando um armazenamento MongoDB.

Capturas de dados de proveniência de *workflows* foram feitas por [52, 53, 54]. Costa *et al* [52] foram capazes de capturar os dados de proveniência de um *workflow* usando o modelo PROV-Wf. Contudo, essa proveniência foi capturada e armazenada em um sistema de bancos de dados relacional apesar de o *workflow* ser executado tanto em ambientes locais como de nuvem. Hondo *et al* [53] apresentam um estudo comparativo entre os bancos de dados NoSQL Cassandra, MongoDB e OrientDB através da execução de um *workflow* de bioinformática de montagem de DNA e armazenamento dos dados de proveniência deste. Hondo *et al* [54], continuando seus trabalhos, executam um estudo sobre *workflows* de bioinformática armazenando as informações de proveniência destes ao executados em nuvem nos mesmos bancos de dados supracitados e então realizando buscas nos bancos de dados criados.

Tratando especificamente dos Bancos de Dados Orientados a Documento, cujo MongoDB é um exemplo, e os trabalhos relacionados à ele, pode-se notar que Sempére *et al* [5] apresentam e analisam uma ferramenta *web* com recursos de compartilhamento chamada Gigwa que depende do MongoDB e é capaz de fornecer uma maneira fácil e intuitiva de explorar grandes quantidades de dados de genotipagem usando filtros de diferentes combinações. Vukicevic *et al* [55] propuseram um sistema de mineração de dados genéticos usando *metalearning*, uma metodologia que encontra padrões e tendências previamente desconhecidos com base no conhecimento passado e usa essa informação para construir

modelos preditivos. Reis *et al* [9] fazem uma análise comparativa entre modelos embutidos, referenciados e híbridos de dados no MongoDB. Os modelos, embora não sendo dados de bioinformática, são comparados entre si com buscas e representam dados de *Big Data*.

Este trabalho se diferencia dos anteriores ao obter a proveniência de um *workflow* de bioinformática com dados *Big Data* ao executá-lo automáticamente com auxílio de um programa criado para, não só realizar as etapas do *workflow*, como também adquirir suas informações de proveniência. Dessa forma, não seria preciso que o pesquisador necessite entrar emcontato com as operações de inserção de dados. Ademais, as informações de proveniência coletadas são então, posteriormente, inseridas no MongoDB pelo mesmo programa em três variações diferentes de modelo (embutido, referencial e híbrido). Essas três abordagens de modelagem são então analisadas e comparadas entre si levando em conta o tempo que o programa levou para criar seus bancos e inserir os dados de proveniência e os dados brutos dos arquivos usados na execução de cada *workflow* utilizado. Também são realizadas considerações à respeito das capacidades e facilidades nas realizações de buscas no banco de dados.

Capítulo 3

Execução

Esse capítulo apresenta uma explicação minuciosa das partes envolvidas na execução desta pesquisa utilizadas com a finalidade de se alcançar a solução para os objetivos apresentados como metas. O capítulo é dividido nas seguintes seções: a Seção 3.1 discorre sobre os modelos propostos e acerca dos ambientes de execução; a Seção 3.2 apresenta em mais detalhes o programa proposto, o *ProvTriplé*, e a Seção 3.3 apresenta os *workflows* utilizados nessa pesquisa.

3.1 Modelos e Ambiente de Execução

De acordo com [56], ainda é um desafio de pesquisa no domínio da bioinformática, particularmente na presença de grandes volumes de dados, a persistência de dados gerados durante a execução de um *workflow*. Ao se executar um *workflow* de bioinformática, geralmente utilizam-se dados de genomas, transcriptomas e outros. Tais tipos de elementos podem gerar um grande volume de dados, o que torna muito importante o armazenamento de proveniência a fim de fornecer o rastreamento e a reprodutibilidade de um experimento [56, 54]. Em Hondo *et al* [54] é proposto um modelo de armazenagem de proveniência de um *workflow* de bioinformática que pode ser visto na Figura 3.1.

Usou-se o modelo de dados proposto em Hondo *et al* [54] como base para, considerando os estudos feitos quanto à armazenagem de proveniência e Bancos de Dados Orientados a Documento, criar três variações de um mesmo modelo usando armazenagens de documentos embutidos, referenciais ou híbridos. Em Reis *et al* [9], é feita uma comparação e análise de performance de três variações diferentes de um modelo de dados. Uma variação foi usando documentos completamente embutidos, outra usou uma modelagem totalmente referencial e a terceira variação considerada foi uma forma híbrida entre as duas anteriores. Primeiramente, focou-se em modelos que seriam armazenados e executados localmente e, por isso, não possuem os campos relativos à armazenagem em Nuvem.

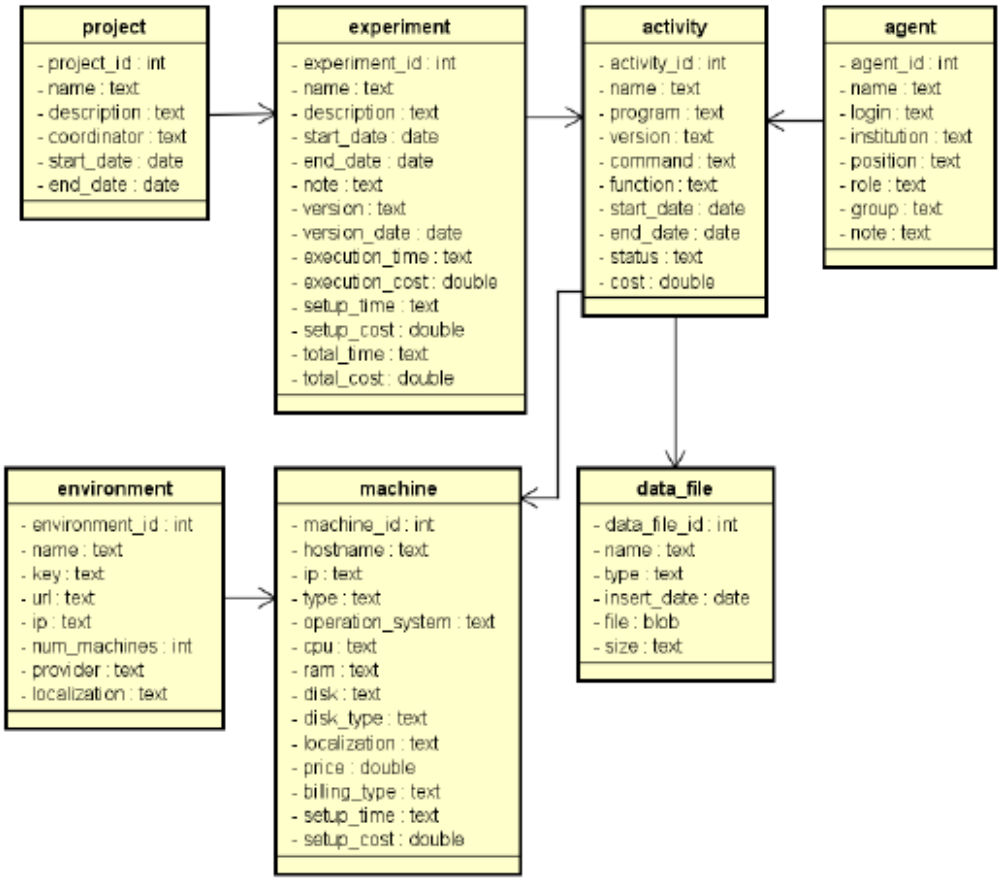

Figura 3.1: Modelo de dados usado de base (Fonte: [54]).

As configurações do computador utilizado podem ser vistas na Tabela 3.1. Em seguida, quando feita a armazenagem em nuvem, foram acrescentadas as informações relativas à tal armazenagem.

O começo, em uma execução local, teve como alvo a realização de aprimoramentos e refinamentos gerais e primordiais do programa implementado, denominado *ProvTriplé*, que tem como finalidade executar o *workflow* automaticamente. As variações do modelo em ambiente local podem ser vistas em Figura 3.2, Figura 3.3 e Figura 3.4. O modelo proposto sem abordagem aplicada em Nuvem Computacional, ou seja, em ambiente local,

Tabela 3.1: Características de Ambiente Local.

Parâmetro	**Valor**
Sistema Operacional	Ubuntu 14.04
RAM	1.8GB
HD	500 GB
Quantidade de CPUs	1

Tabela 3.2: Características de Ambiente em Nuvem.

Parâmetro	**Valor**
Sistema Operacional	Ubuntu 14.04
RAM	3GB
HD	60 GB
zona	us-west1-b
Tipo	Disco permanente SSD
Quantidade de CPUs	1

foi usado apenas para o primeiro *workflow* trabalhado. Além disso, assim como o modelo proposto por Hondo *et al* [54], usado como base, as variações de modelo propostas neste trabalho seguem os conceitos propostos pela modelagem PROV-DM.

Posteriormente voltou-se o foco para a armazenagem na nuvem que foi consequentemente responsável por gerar os modelos vistos em Figura 3.5, Figura 3.6 e Figura 3.7. Nesses modelos, já é possível ver campos e documentos relacionados às informações de proveniência da nuvem utilizada. Sendo o ambiente de execução principal desse trabalho, todos os *workflows* foram executados no ambiente de nuvem. A nuvem utilizada foi criada usando o *Google Cloud Platform* e possui as definições mencionadas em Tabela 3.2.

Tais valores foram escolhidos para tentar aproximar mais o ambiente de testes em nuvem do ambiente de testes local. Isso pode ser visto na escolha do Sistema Operacional e na quantidade de CPU escolhidas que são as mesmas. No entanto, devido a dificuldades enfrentadas no ambiente local causada pela grande quantidade de dados do primeiro *workflow*, escolheu-se por aumentar a quantidade de memória RAM de 1.8GB para 3GB.

3.1.1 Descrição dos Modelos

Para facilidade de nomenclatura nas demais partes dessa pesquisa, nesse trabalho, optou-se por denominar os modelos de armazenagem de dados de proveniência em documentos referenciados, com ou sem dados de nuvem, de *Proveniência Referencial*. Do mesmo modo, os modelos de armazenagem de dados de proveniência em documentos embutidos, com ou sem dados de nuvem, foram designados como *Proveniência Embutida* e os modelos de armazenagem de dados de proveniência em documentos híbridos, com ou sem dados de nuvem, ficaram denominados como *Proveniência Híbrida*.

Ao se propor os novos modelos, algumas características do modelo proposto em [54] foram mantidas, enquanto outras foram alteradas. Embora cada versão da modelagem tenha suas particularidades, alguns atributos se mantém para todas as versões. Os campos presentes em cada documento podem ser mencionados como semelhanças ou diferenças geralmente dependendo do caso. Para os modelos ainda de armazenagem local, cita-se como mudança a inexistência dos documentos *Environment* e *Machine* pois os mesmos

contém os dados da armazenagem em nuvem. Para os casos da Figura 3.2, Figura 3.3, Figura 3.4, Figura 3.5, Figura 3.6 e Figura 3.7, as setas vermelhas representam as referências enquanto as setas pretas representam quais são os documentos embutidos de acordo com a cor. Nas três formas também é possível ver as coleções *fs.chunks* e *fs.files*, referentes as coleções criadas pelo GridFS automáticamente. Tal metodologia é necessária devido ao tamanho de alguns dos arquivos usados no *workflow* que excede o tamanho máximo suportado pelo MongoDB. Ressalta-se que o *ProvTriplé* foi implementado de forma que à particionar os dados brutos seguindo a estrutura do GridFS e, posteriormente, inseri-los no MongoDB.

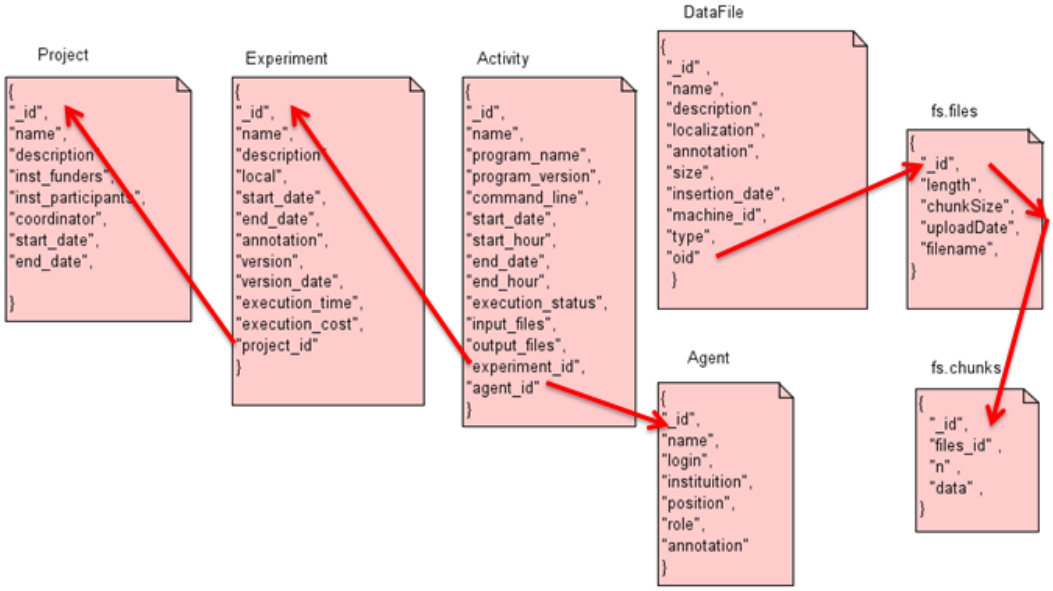

Figura 3.2: Modelo de armazenagem de proveniência referencial sem nuvem.

Para Figura 3.2, nota-se as referências entre os documentos de forma que cada um está contido dentro de sua respectiva coleção. Ou seja, existe uma coleção para *Project*, outra para *Experiment*, outra para *Activity*, outra para *Agent* e outra para *DataFile*.

Em Figura 3.3, nota-se os documentos embutidos por meio da separação da cor destes. As coleções presentes são apenas as padrões do GridFS e a que contém todos os outros dados como de *Project*, *Experiment*, *Activity*,*Agent* e *DataFile*. Por ser a única com dados não brutos, ela foi chamada de *default*. Apesar da ideia de uma modelagem completamente embutida ser de ter apenas uma coleção, isso não é possível devido ao tamanho excessivo dos dados brutos de bioinformática que torna enviável embuti-los em outro documento.

Em Figura 3.4 usa-se a modelagem híbrida para criar algumas coleções com documentos embutidos e outras representadas de forma referencial. Por exemplo, nota-se que *Project*, *Experiment*, *Activity* e *Agent* estão dentro de um mesmo documento de forma

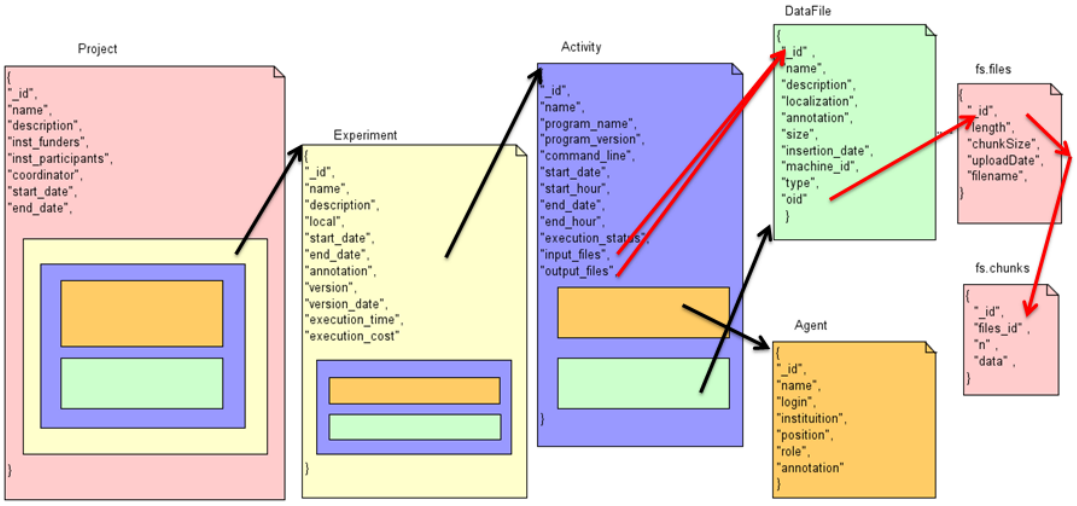

Figura 3.3: Modelo de armazenagem de proveniência embutida sem nuvem.

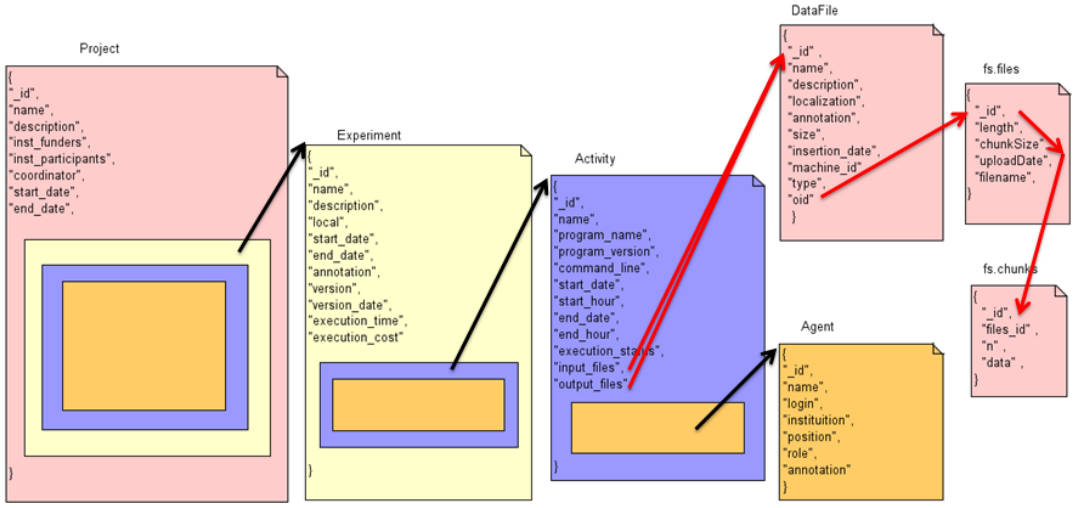

Figura 3.4: Modelo de armazenagem de proveniência híbrida sem nuvem.

embutida que pertence à uma mesma coleção, enquanto *DataFile* está presente em uma coleção própria.

A diferença entre Figura 3.5 e Figura 3.2 pode ser observada na inserção das coleções de *Provider*, *Cluster* e *Machine*. Assim como na Figura 3.2, nota-se as referências entre os documentos de forma que cada um está contido dentro de sua respectiva coleção.

Além da inserção dos documentos embutidos representando, *Provider*, *Cluster* e *Machine*, que não estão presentes em Figura 3.3, nota-se também em Figura 3.6 as demais semelhanças com Figura 3.3. Pode-se observar os documentos embutidos através da separação da cor destes. Sendo assim, as coleções presentes são apenas as padrões do GridFS e a que contém todos os outros dados como de *Project*, *Experiment*, *Activity*, *Agent*, *Da-*

Figura 3.5: Modelo de armazenagem de proveniência referencial com nuvem.

taFile, *Provider*, *Cluster* e *Machine*, por ser única, ela foi chamada de *default*. Apesar da ideia de uma modelagem completamente embutida considerar a existência de apenas uma coleção, isso não é possível devido ao tamanho dos dados brutos de bioinformática.

Na Figura 3.7 usa-se a modelagem híbrida para criar algumas coleções com documentos embutidos e outras representadas de forma referencial. Nota-se facilmente as semelhanças entre Figura 3.4 e Figura 3.7. Por exemplo, nota-se que *Project*, *Experiment*, *Activity* e *Agent* seguem estando dentro de um mesmo documento que pertence à uma mesma coleção

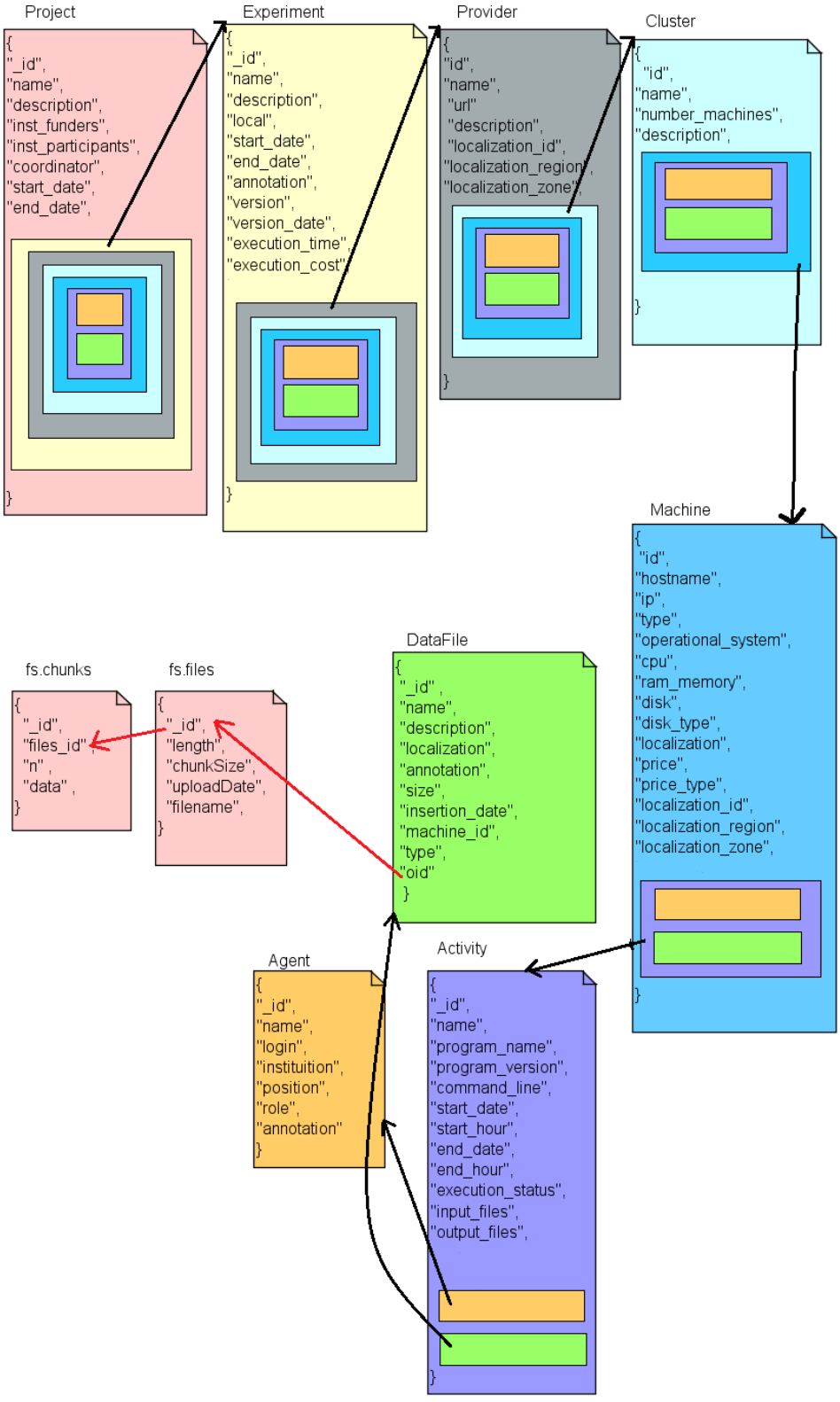

Figura 3.6: Modelo de armazenagem de proveniência embutido com nuvem.

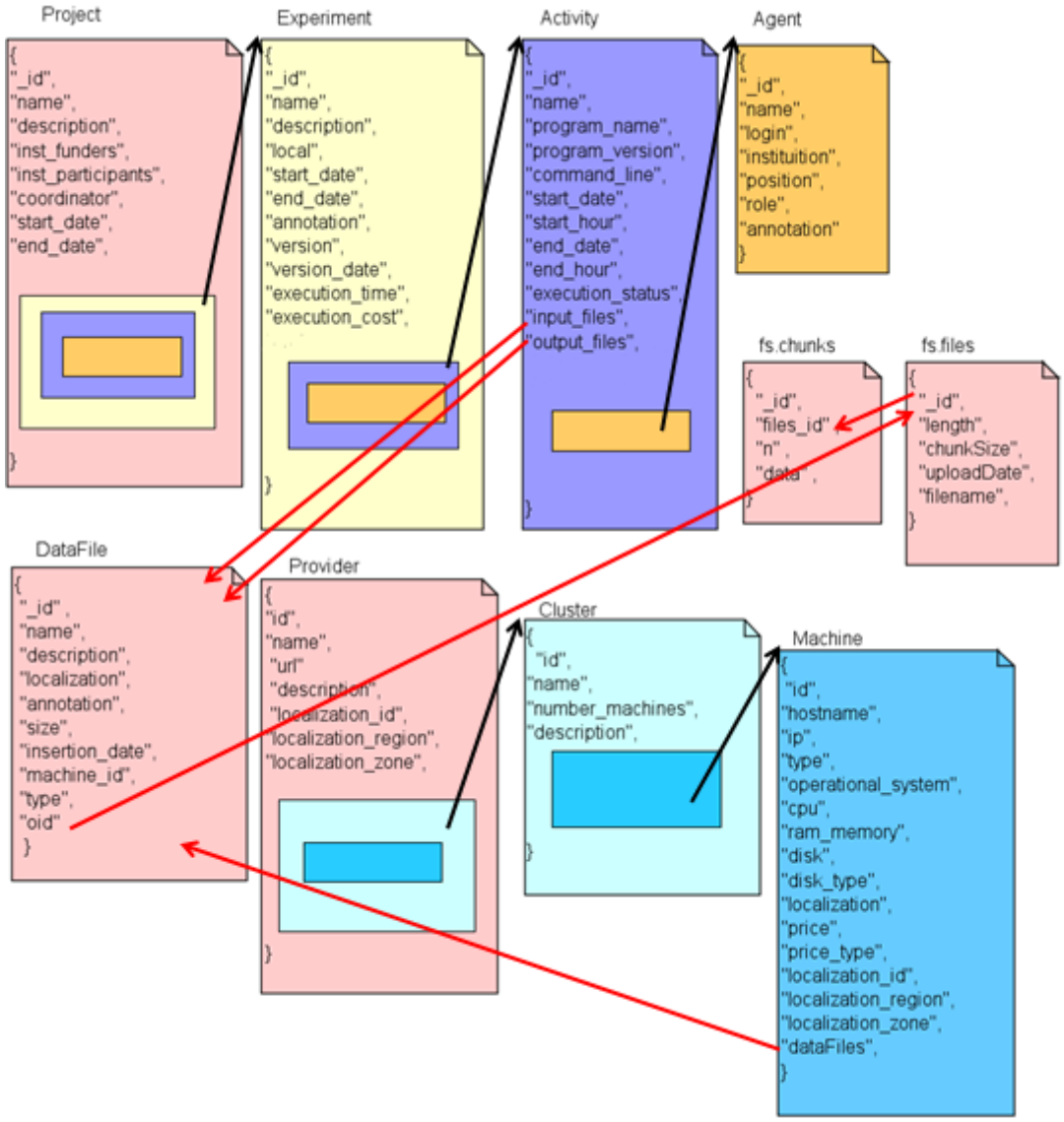

Figura 3.7: Modelo de armazenagem de proveniência híbrido com nuvem.

denominada *Project* e que *DataFile* está presente em uma coleção própria. Contudo, é possível observar também os documentos referentes a *Project*, *Cluster* e *Machine* que estão dentro da coleção *Provider*.

Um fator que deve ser observado para as modelagens propostas é que, embora exista um documento que contém as informações básicas dos arquivos usados para executar o *workflow*, também existe a armazenagem dos dados brutos deste arquivo. Tal armazenagem é feita usando o conceito de GridFS do MongoDB uma vez que os documentos do MongoDB são limitados ao tamanho de 16 MB [25, 28]. A tecnologia de GridFS permite com que seja possível inserir documentos maiores que 16MB através por meio da divisão do arquivo em partes denominadas *chunks*. Tais partes, são fragmentos geralmente de

255kB criados pelo GridFS [28]. Os dados da chave de identificação do *chunk* inicial de cada arquivo é inserido no documento *dataFile*.

3.2 Programa Executável

Tendo os *workflows* como base, foi criado um programa implementado em C com auxílio do *driver* de C para MongoDB, o MongoC. Tal programa, denominado *ProvTriplé*, executa este *workflow* de forma mais facilitada criando enquanto armazena as informações básicas e de proveniência no MongoDB com auxílio do driver do MongoC. O *ProvTriplé* é capaz de, por conta própria e com base nos *workflows*, definir algumas informações de proveniência e, com base no próprio sistema computacional, definir demais informações como Sistema operacional, Tempo de Início e Tempo de Fim. A Figura 3.8 apresenta um fluxograma simplificado da execução do *ProvTriplé* e, através dela, é possível compreender o processo de execução deste na execução do *workflow* e criação dos modelos de dados.

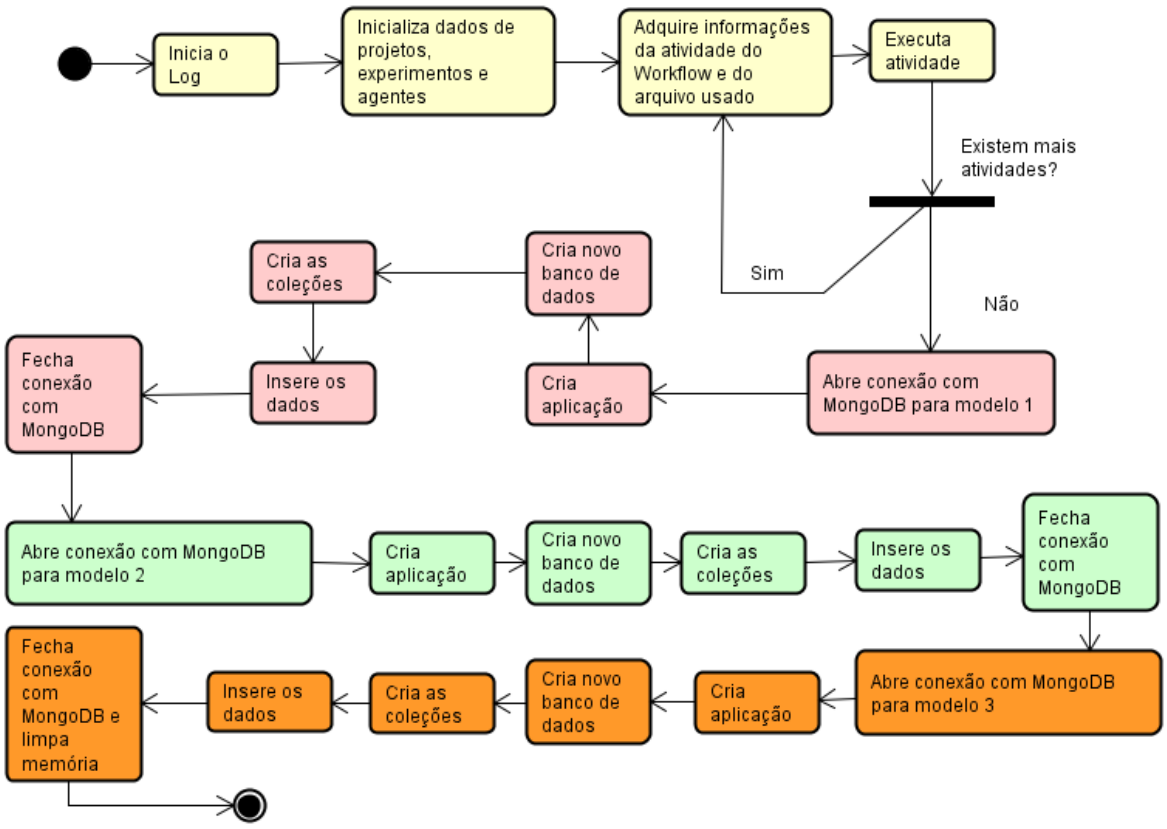

Figura 3.8: Fluxograma simplificado de execução.

Para a execução do *workflow* e do *ProvTriplé*, foram necessárias as instalações de alguns programas, *drivers* ou *frameworks*. Seus nomes e versões utilizadas no trabalho

Tabela 3.3: Nome e versão dos programas utilizados.

Nome dos programas	**Versão utilizada**
HISAT	2.0.5
samtools	1.4.1
ABYSS	1.3.6
HTSeq	0.7.2
sratoolkit	2.7.1
Sickle	1.33
MongoC	1.6.3

podem ser vistos na Tabela 3.3 sendo que apenas o MongoC foi o necessário especificamente para o *ProvTriplé* enquanto as demais ferramentas foram necessárias para a execução do *workflow*.

Primeiramente, o *ProvTriplé* obtém os dados necessários para a execução do *Workflow*. Cada comando a ser executado possui suas particularidades, como por exemplo, quais serão os nomes dos arquivos de entrada e saída. O *ProvTriplé* é capaz de separar os arquivos de etapa do *workflow* entre *entrada* ou *saída* de acordo com a já existência de cada arquivo ou não na pasta raíz. Tendo as informações necessárias, o *ProvTriplé* monta a sequência de caracteres que representa o comando e então o executa através de uma chamada ao sistema. Após o final da execução do comando, armazena-se em diversas estruturas e variáveis os dados que são pertinentes para serem inseridos no MongoDB. Um dado importante que deve ser salientado é, por exemplo, o nome dos arquivos. Os nomes dos arquivos, sejam eles usados como entrada ou saída, são armazenados em uma estrutura de lista que tenta também evitar a repetição dos mesmos. Essa estrutura é de grande ajuda para a posterior inserção dos dados brutos e informações gerais dos arquivos no MongoDB em passos posteriores. Os passos de obtenção de dados necessários, execução dos comandos e armazenamento de dados importantes no MongoDB são repetidos para todos os comandos do *Workflow*.

Uma vez que todos os comandos são executados, o *ProvTriplé* usa o MongoC para abrir uma conexão com o MongoDB. Através de funções específicas do MongoC, cria-se uma nova aplicação, um banco de dados e a quantidade de coleções necessárias a cada modelo. Cada modelo é inserido em um banco de dados MongoDB diferente para evitar sobreposição de dados e todas as suas operações são feitas de forma sequencial. Ou seja, a conexão com o MongoDB para iniciar a inserção de dados de um modelo só é iniciada quando todos os passos do modelo anterior já foram executados. No caso dos dados brutos, eles são fragmentados e então inseridos no MongoDB no sistema de GridFS. Assim que todos os documentos desejados de um modelo são criados, fecha-se a conexão com o MongoDB faz-se o mesmo para o modelo seguinte.

3.3 *Workflows* executados

Os modelos de dados propostos e o *ProvTriplé* foram testados através da execução de dois *workflows* de bioinformática. A seguir são descritos os *workflows* analisados nesse processo.

3.3.1 *Workflow* 1

Por meio desse *workflow*, leituras obtidas por sequenciamento de alto desempenho são mapeadas em um genoma de referência, no caso, o cromossomo 22 do genoma humano, com o uso da ferramenta HISAT [57].

Este *workflow* foi sugerido por conter as fases de análise de genoma, como, por exemplo, as fases de mapeamento e contagem. Para que seja possível a execução do *workflow*, se faz necessária a existência de alguns arquivos de bioinformática. São eles o *Homo_sapiens.GRCh38.88.gtf*, *Homo_sapiens.GRCh38.dna.chromosome.22.fa* e, por último, o *SRR5181508*. Enquanto os arquivos *Homo_sapiens .GRCh38.88.gtf* e *Homo_sapiens .GRCh38.dna.chromosome.22.fa* podem ser obtidos através do *Ensemble Genome Browser* [58] e são necessários para fazer o mapeamento e contagem do cromossomo número 22 da espécie humana, o *SRR5181508* pode ser adquirido por meio do *European Nucleotide Archive* [59].

O primeiro *workflow* usado para os testes e implementação do *ProvTriplé* pode ser visto na Figura 3.9 e leva aproximadamente 23 minutos para ser executado no ambiente de nuvem criado por meio do *Google Cloud Platform*.

Através da Figura 3.9 pode-se notar que alguns arquivos servem de entrada, saída ou entrada e saída no decorrer do workflow e ao considerar cada um dos comandos a serem executados. Por meio da separação de cores, é possível ver também quais comandos representam cada fase onde a cor rosa representa a fase de filtragem, a cor azul a fase de mapeamento, roxo a fase de conversão de arquivos e laranja a fase de contagem. Após a conversão do formato de arquivo de mapeamento pelo conjunto de ferramentas samtools [60], realiza-se a contagem de número de *reads* mapeadas em regiões gênicas utilizando o programa HTSeq-count [61] e o arquivo de anotação do genoma de referência em formato GTF.

3.3.2 *Workflow* 2

O *workflow* 2 é encarregado da montagem do genoma de uma bactéria resistente a múltiplos fármacos denominada *Enterobacter kobei*. Ele é exibido na Figura 3.10 e foi escolhido por diferenciar-se do *workflow* 1 e, dessa forma, tornar possível testar outras caracterísi-

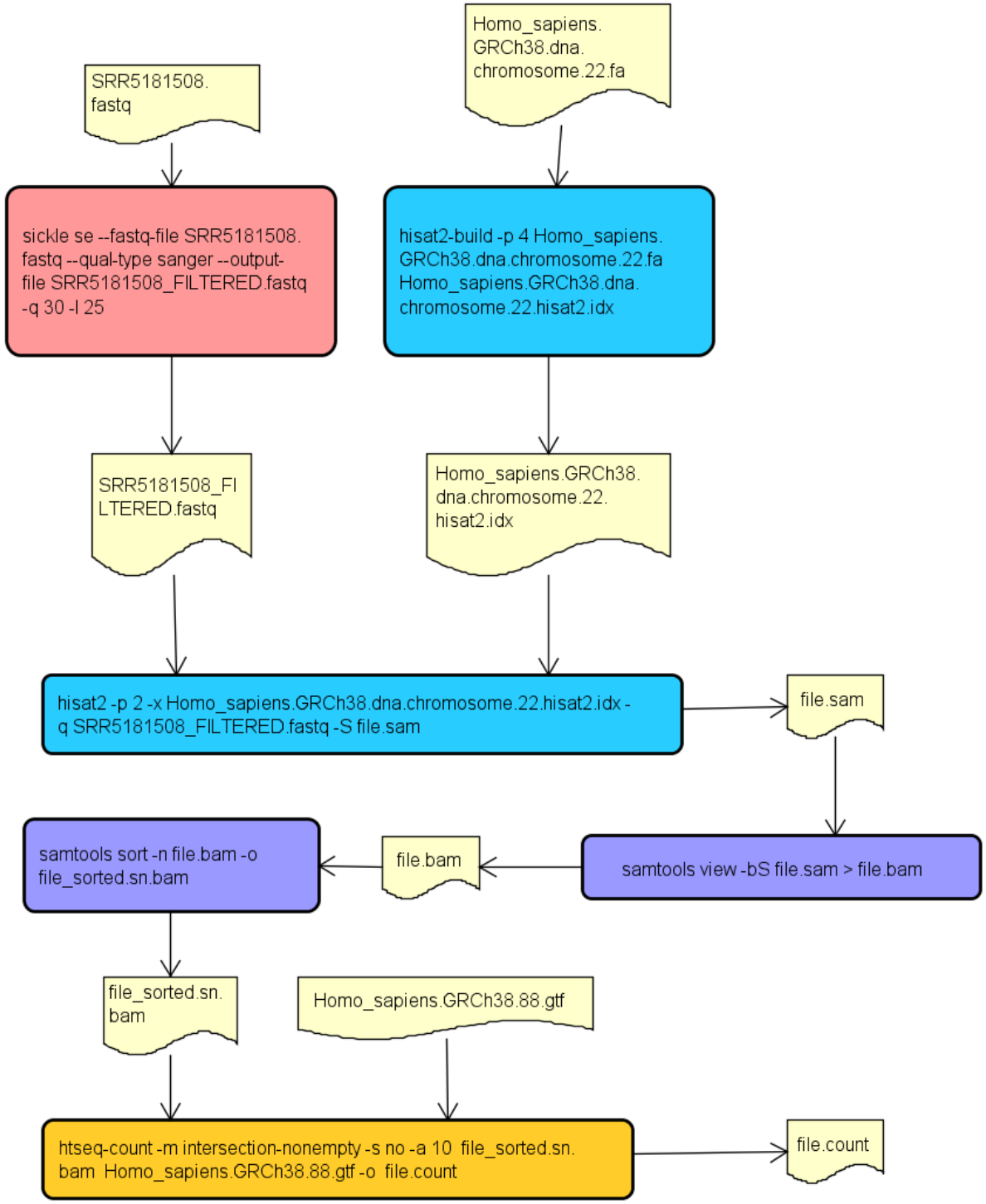

Figura 3.9: *Workflow* 1.

cas e partes do *ProvTriplé*. Ele apresenta as fases de filtragem e montagem. Além disso, enquanto a fase de filtragem usa o mesmo programa usado do *workflow* anterior, a fase de montagem possui um programa que não foi aplicado anteriormente.

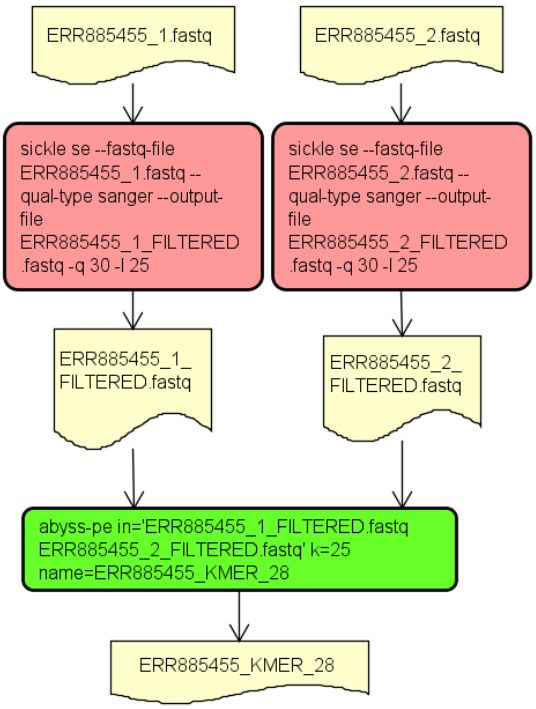

Figura 3.10: *Workflow* 2.

O *workflow* usado para os testes e implementação do programa proposto pode ser visto na Figura 3.10 e ele leva cerca de 10 minutos para ser executado no ambiente de nuvem computacional criado utilizando o *Google Cloud Platform*. Para que seja possível a execução deste *workflow*, faz-se necessário a existência de alguns arquivos de bioinformática. Esses arquivos são o *ERR885455_1.fastq* e o *ERR885455_2.fastq* que podem podem ser obtidos por meio do *European Nucleotive Archive* [59].

Por meio da Figura 3.10 pode-se notar que alguns arquivos servem de entrada, saída ou entrada e saída no decorrer do *workflow* e ao considerar cada um dos comandos a serem executados. Por meio da separação de cores, é possível ver também quais comandos representam cada fase com rosa representando a fase de filtragem e verde a fase de montagem.

Capítulo 4

Análise dos Resultados

Esse capítulo expõe a análise dos resultados obtidos com os experimentos realizados. O capítulo é dividido nas seguintes seções: a Seção 4.1 apresenta os resultados obtidos no ambiente local para o *workflow* 1 e os recolhidos no ambiente de nuvem para os *workflows* 1 e 2; a Seção 4.2 expõe graficamente os tempos médios de inserção de dados calculados e, por fim, a Seção 4.3 apresenta resultados acadêmicos atingidos durante as pesquisas que foram realizadas para este trabalho.

4.1 Resultado da execução dos *Workflows*

Com auxílio das informações contidas no arquivo de log gerado pelo *ProvTriplé*, anotou-se o tempo que cada execução de cada modelo demorou para ser concluída. O *ProvTriplé* foi executado pelo menos quatro vezes para cada modelo e, antes de cada vez, limpou-se completamente o banco de dados para evitar sobreposição de dados. Além disso, um conjunto de indagações foi executado para cada modelo visando-se obter a mesma informação de cada um. Algumas das perguntas foram baseadas nas interrogações realizadas por Hondo *et al* [54]. Em Hondo *et al* [54], os questionamentos feitos sobre a base de dados foram:

1. Quais são os nomes e as versões dos programas usados para executar as atividades do *workflow*?
2. Qual é a configuração da nuvem usada no *workflow* (máquinas, processadores, memória, etc)?
3. Qual o custo de executar o *workflow* na nuvem?

Ao se considerar as indagações que seriam feitas nos modelos propostos desse trabalho, julgou-se que seria interessante utilizar os questionamentos propostos por Hondo *et al* [54] como base de inspiração. Por conseguinte, as perguntas aplicadas nesse trabalho foram:

1. Quais são os nomes e as versões dos programas usados para executar as atividades do *workflow*?
2. Quantas atividades foram executadas pelo primeiro experimento?
3. Qual a quantidade de máquinas e a região no *Google Cloud*?

Enquanto a primeira interrogação feita é a equivalente à pergunta inicial realizada no trabalho em Hondo *et al* [54], as demais são diferentes e incluem pelo menos um questionamento por dados referentes ao ambiente de nuvem computacional. Além disso, exclusivamente no ambiente de nuvem, optou-se por executar um pequeno conjunto de buscas por dados brutos inseridos no banco de dados do MongoDB. A exclusividade de execução no ambiente de nuvem deve-se simplesmente ao fato de que o ambiente local, no momento em que se optou por incluir estes novos testes, não estar mais com a capacidade física e lógica de executar nenhum outro teste devido à mal funcionamentos do computador utilizado.

Faz-se importante lembrar que, visando facilitar o vocabulário, nesse trabalho, optou-se por designar os modelos de armazenagem de dados de proveniência em documentos referenciados como *Proveniência Referencial*, os modelos de armazenagem de dados de proveniência em documentos embutidos, como *Proveniência Embutida* e os modelos de armazenagem de dados de proveniência em documentos híbridos, como *Proveniência Híbrida*.

4.1.1 Ambiente Local

A seguir são apresentados os resultados obtidos com a execução do *workflow* 1 no ambiente local. Apenas o *workflow* 1 foi executado nesse ambiente pois, primeiramente, um dos objetivos visados por essa etapa tornou-se o aprimoramento do *ProvTriplé*. Ademais, a baixa quantidade de RAM disponível no ambiente local se tornou um grande dificultador que, geralmente, era capaz de inviabilizar o prosseguimento das análises. Dessa forma, o ambiente de nuvem surgiu como opção posteriormente para melhor obtenção e observação dos dados.

***Workflow* 1**

Os resultados obtidos de cada execução do *workflow* 1 em um ambiente local podem ser vistos a seguir na seguinte ordem: proveniência referencial, proveniência embutida e proveniência híbrida. Para cada caso são inseridas também tabelas e imagens com os dados obtidos relativos a inserção e busca. A quantidade de tempo que cada busca

Tabela 4.1: Tempos de Inserção da Proveniência Referencial para *workflow* 1.

Parâmetro	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	16 : 25 : 01	16 : 56 : 07	1866
Execução 2	17 : 45 : 15	18 : 10 : 54	1539
Execução 3	19 : 07 : 50	19 : 33 : 09	1519
Execução 4	20 : 27 : 08	21 : 01 : 25	2057
Duração Média 1	–	–	1745,25
Duração Média 2	–	–	1702,5

levou para ser realizada não foi calculado para nenhum caso uma vez que esta se mostrou demasiadamente pequena para ser comparada.

Armazenagem de Proveniência Referencial

A armazenagem de proveniência referencial aqui explorada foi construída segundo a exibida na Figura 3.2. Os tempos de execução para inserção de dados da proveniência referencial podem ser vistos em Tabela 4.1 enquanto as buscas realizadas e os resultados destas podem ser vistas na Figura 4.1.

A Tabela 4.1 é composta por quatro colunas. A primeira coluna contém os parâmetros para identificação dos dados contidos em cada linha sendo eles divididos em: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Duração Média 1* e *Duração Média 2*. Enquanto as Execuções 1 a 4 enumeram e representam cada vez que o *ProvTriplé* foi processado, considerou-se que, *Duração Média 1* simboliza a média considerando todos os resultados e que *Duração Média 2* constitui a média desconsiderando-se o maior e o menor resultado. A segunda coluna representa o horário do relógio do computador no momento que teve início a inserção de dados no MongoDB. De forma análoga, a terceira coluna expõe o horário, segundo o relógio do computador, em que se teve fim a introdução dos dados do modelo de proveniência referencial. Ambos os horários são representados no formato hh:mm:ss onde 'h' representa *hora*, 'm' simboliza *minutos* e 's' os *segundos*. A última coluna salienta o tempo de duração total em segundos de cada execução considerando-se a diferença entre o tempo inicial (coluna dois) e o tempo final (coluna três). Esses dados também são os usados no cálculo de média para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

Na Figura 4.1 é possível observar os questionamentos feitos, o comando inserido no MongoDB para obter a resposta e os resultados obtidos com cada comando. Ao se realizar a primeira indagação *"Quais são os nomes e as versões dos programas usados no Workflow 1?"*, a busca realizada retornou as respostas esperadas exibindo as versões dos programas usados: *HTseq*, *samtools*, *hisat2* e *sickle*. O mesmo ocorreu ao questionar o modelo no MongoDB sobre *"Quantas atividades foram executadas pelo primeiro experimento?"*.

Nesse caso, assim como esperado, a conclusão obtida mostra as seis etapas existentes no *Workflow* 1. Deve-se salientar que a terceira indagação proposta nessa pesquisa, *"Qual o custo de executar o workflow na nuvem?"* não foi realizada uma vez que o modelo proposto para a fase de análises em ambiente local não contém dados referentes a nuvem e, por isso, não seria possível realizar tal questionamento neste dado momento.

Quais são os nomes e as versões dos programas usados no *workflow* 1?

```
db.activity3.aggregate([{$match:{"experiment_id":"1"}},
{$unwind:"$_id"},{$group:{_id:{program:"$program_name",
version:'$program_version'}}}])
```

resultados:

```
{"_id": {"program": "htseq", "version": "0.7.2"}}
{"_id": {"program": "samtools", "version": "1.4.4"}}
{"_id": {"program": "hisat2", "version": "2.0.5"}}
{"_id": {"program": "sickle", "version": "1.33"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.activity3.count({experiment_id: "1"})
```

resultados:

```
6
```

Figura 4.1: Buscas executadas no *workflow* 1 para Proveniência Referencial.

Armazenagem de Proveniência Embutida

A armazenagem de proveniência embutida analisada foi construída segundo a exibida na Figura 3.3. Os tempos de execução para inserção de dados de proveniência no modelo embutido podem ser visualizados em Tabela 4.2. Semelhantemente, as buscas realizadas e os resultados destas podem ser vistas na Figura 4.2.

A Tabela 4.2 é formada por quatro colunas onde a primeira, denominada *parâmetros*, é responsável pela identificação dos dados contidos em cada linha. Tais dados são distribuídos entre: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Duração Média 1* e *Duração Média 2*. Ao passo que as Execuções 1 a 4 enumeram e representam cada vez que o *ProvTriplé* foi processado para o ambiente considerado, declarou-se que, *Duração Média 1* equivale à média entre todos os resultados obtidos e que *Duração Média 2* simboliza a média desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna. A segunda coluna contém o horário do relógio do computador no momento que se teve início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira coluna expõe o horário, segundo o relógio do computador, em que se teve fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna salienta o tempo de duração total em segundos de cada execução por meio

Tabela 4.2: Tempos de Inserção da Proveniência Embutida para *workflow* 1.

Parâmetros	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	15 : 52 : 46	16 : 25 : 01	1935
Execução 2	17 : 22 : 15	17 : 45 : 15	1380
Execução 3	18 : 42 : 34	19 : 07 : 50	1516
Execução 4	20 : 00 : 05	20 : 27 : 08	1623
Duração Média 1	–	–	1613,5
Duração Média 2	–	–	1569,5

do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.2 demonstra os questionamentos feitos para esta etapa atual, os comandos inseridos no MongoDB para responder à cada questão e os respectivos resultados. Ao se realizar a primeira indagação *"Quais são os nomes e as versões dos programas usados no Workflow 1?"*, a busca realizada retornou as respostas esperadas exibindo as versões dos programas usados: *HTseq*, *samtools*, *hisat2* e *sickle*. Embora as buscas tenham sido construídas visando responder às mesmas perguntas em todos os modelos, é possível observar que existem algumas pequenas diferenças na construção do comando proposto para responder a mesma pergunta ao se comparar os comandos presentes na Figura 4.2 e na Figura 4.1. Isso se deve às dessemelhanças presentes ao se comparar cada modelo entre si. De forma análoga, ao questionar o MongoDB sobre *"Quantas atividades foram executadas pelo primeiro experimento?"*, obteve-se o resultado desejado que representa as seis etapas existentes no *Workflow* 1 mesmo considerando-se que a concepção da instrução de busca foi diferente da observada na Figura 4.1. Deve-se salientar que a terceira indagação proposta, *"Qual o custo de executar o workflow na nuvem?"* não foi realizada uma vez que, no momento, trata-se da fase de execução em ambiente local, não existem campos de nuvens no modelo o que impossibilita o questionamento por tal informação.

Armazenagem de Proveniência Híbrida

A armazenagem de proveniência híbrida aqui explorada foi construída segundo a exibida na Figura 3.4. Os tempos de execução para inserção de dados de proveniência de forma híbrida podem ser vistos em Tabela 4.3. Além disso, também é possível observar as buscas realizadas e seus respectivos resultados na Figura 4.3.

A tabela Tabela 4.3 é constituída por quatro colunas: *Parâmetros*, *Tempo Inicial*, *Tempo Final* e *Duração*. A coluna *Parâmetros* identifica os dados contidos em cada linha. Tais dados podem ser repartidos entre as seguintes denominações: *Execução 1*, *Execução*

Quais são os nomes e a versão dos programas usados no *workflow* 1?

```
db.default.aggregate([{$match:{$and:[{id:"1"},
{"experiment.id":"1"},]}}, {$unwind:"$experiment.activity"},
{$group:{_id:{program:"$experiment.activity.program_name",
version:'$experiment.activity.program_version'}}}])
```

resultados:

```
{"_id": {"program": "htseq", "version": "0.7.2"}}
{"_id": {"program": "samtools", "version": "1.4.4"}}
{"_id": {"program": "hisat2", "version": "2.0.5"}}
{"_id": {"program": "sickle", "version": "1.33"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.default.aggregate([{$match:{"experiment_id": "1"}},
{$project: {numberOfActivities:
{$size:"$experiment.activity"}}}])
```

resultados:

```
{"_id": "1", "numberOfActivities": 6}
```

Figura 4.2: Buscas executadas no *workflow* 1 para Proveniência Embutida.

2, *Execução 3*, *Execução 4*, *Duração Média 1* e *Duração Média 2*. Cada uma das quatro execuções do *ProvTriplé* consideradas possuem seus dados representados pelos parâmetros *Execução 1*, *Execução 2*, *Execução 3* e *Execução 4*. Os parâmetros *Duração Média 1* e *Duração Média 2* representam as médias padrões calculadas usando os dados obtidos na coluna *Duração* como base. Ao passo que *Duração Média 1* equivale à média entre todos os resultados obtidos, a *Duração Média 2* simboliza a média desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna. A segunda coluna, *Tempo Inicial*, contém o horário que, pelo relógio do computador, teve-se início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira, *Tempo Final*, expõe o horário que, segundo o relógio do computador, teve-se fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna contém o tempo de duração total em segundos de cada execução por meio do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.3 apresenta questionamentos feitos para esta etapa atual e os resultados obtidos para cada um. Primeiramente é exibido a pergunta feita, em seguida, o comando inserido no MongoDB para obter a resposta à indagação e, logo após, os resultados obtidos. As interrogações *"Quais são os nomes e as versões dos programas usados no Workflow 1?"* e *"Quantas atividades foram executadas pelo primeiro experimento?"* obtiveram os mesmos

Tabela 4.3: Tempos de Inserção da Proveniência Híbrida para *workflow* 1.

Parâmetros	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	15 : 32 : 19	15 : 52 : 46	1227
Execução 2	16 : 59 : 06	17 : 22 : 15	1389
Execução 3	18 : 15 : 04	18 : 42 : 34	1650
Execução 4	19 : 35 : 57	20 : 00 : 05	1448
Duração Média 1	–	–	1428,5
Duração Média 2	–	–	1418,5

resultados dos exibidos de forma equivalente na Figura 4.1 e Figura 4.2. Deve-se salientar que a terceira indagação proposta, *"Qual o custo de executar o workflow na nuvem?"* não foi realizada por se tratar da fase de execução em ambiente local, não existem campos de nuvens no modelo o que impossibilita o questionamento por tal informação. Embora as buscas tenham sido construídas visando responder às mesmas perguntas em todos os modelos, é possível observar que existem algumas pequenas diferenças na construção de cada um dos comandos proposto para responder às perguntas ao se comparar a Figura 4.1, a Figura 4.2 e a Figura 4.3. Isso se deve às dessemelhanças presentes ao se comparar cada modelo entre si.

Quais são os nomes e a versão dos programas usados no Workflow 1?

```
db.project1.aggregate([{$match:{$and:[{id:"1"},
{"experiment.id":"1"},]}}, {$unwind:"$experiment.activity"},
{$group:{_id:{program:"$experiment.activity.program_name",
version:'$experiment.activity.program_version'}}}])
```

resultados:

```
{"_id": {"program": "htseq", "version": "0.7.2"}}
{"_id": {"program": "samtools", "version": "1.4.4"}}
{"_id": {"program": "hisat2", "version": "2.0.5"}}
{"_id": {"program": "sickle", "version": "1.33"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.project1.aggregate([{$match:{"experiment_id": "1"}},
{$project: {numberOfActivities: {$size:"$experiment.activity"}}}])
```

resultados:

```
{"_id": "1", "numberOfActivities": 6}
```

Figura 4.3: Buscas executadas *workflow* 1 para Proveniência Híbrida.

4.1.2 Ambiente de Nuvem

A seguir são apresentados os resultados obtidos após a execução do *ProvTriplé* para o *workflow* 1 e *workflow* 2 no ambiente de nuvem construído com a *Google Cloud Platform*

conforme as especificações de máquina mencionadas no Capítulo 3.

Workflow 1

Os resultados obtidos de cada execução do *workflow* 1 em um ambiente de nuvem podem ser vistos a seguir na seguinte ordem: proveniência referencial, proveniência embutida e proveniência híbrida. A quantidade de tempo que cada busca levou para ser realizada não foi calculado para nenhum caso uma vez que esta se mostrou demasiadamente pequena para ser comparada.

Armazenagem de Proveniência Referencial

A armazenagem de proveniência referencial explorada para este *workflow* foi construída segundo a exibida na Figura 3.4. Os tempos de execução para inserção de dados de proveniência de forma referencial podem ser vistos em Tabela 4.4 enquanto as buscas realizadas e seus resultados podem ser vistos em Figura 4.4.

A tabela Tabela 4.4 é composta por quatro colunas denominadas *Parâmetros*. *Tempo Inicial*, *Tempo Final* e *Duração*. A primeira coluna contém os parâmetros para identificação dos dados contidos em cada linha sendo eles divididos em: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Aqueles que representam execuções (de *Execução 1* a *Execução 6*) enumeram e representam cada vez que o *ProvTriplé* foi processado. Por outro lado, considerou-se que, *Duração Média 1* simboliza a média considerando todos os resultados e que *Duração Média 2* constitui a média desconsiderando-se o maior e o menor resultado. A segunda coluna representa o horário do relógio da máquina da nuvem da *Google* no momento que teve início a inserção de dados no MongoDB. De forma análoga, a terceira coluna expõe o horário, segundo o relógio da máquina da nuvem da *Google*, em que se teve fim a introdução dos dados do modelo de proveniência referencial. Ambos os horários são representados no formato hh:mm:ss onde 'h' representa *hora*, 'm' simboliza *minutos* e 's' os *segundos*. A última coluna salienta o tempo de duração total em segundos de cada execução considerando-se a diferença entre o tempo inicial (coluna dois) e o tempo final (coluna três). Esses dados também são os usados no cálculo de média para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

Na Figura 4.4 é possível observar os questionamentos feitos, o comando inserido no MongoDB para obter a resposta e os resultados obtidos. Ao analisar a Figura 4.4, pode-se constatar que os comandos da primeira indagação, *"Quais são os nomes e as versões dos programas usados no workflow 1?"*, e da segunda, *"Quantas atividades foram executadas pelo primeiro experimento?"*, assim como seus resultados, são análogos aos observados na Figura 4.1. Isso se dá devido à mesma estrutura do modelo de dados ao se tratar dos

Tabela 4.4: Tempos de Inserção da Proveniência Referencial para *workflow* 1.

Parâmetros	Tempo Inicial	Tempo Final	Duração (segundos)
Execução 1	19 : 08 : 34	19 : 18 : 39	605
Execução 2	16 : 27 : 56	16 : 37 : 30	574
Execução 3	20 : 15 : 35	20 : 25 : 13	578
Execução 4	17 : 44 : 04	17 : 53 : 38	574
Execução 5	21 : 00 : 11	21 : 09 : 46	575
Execução 6	21 : 30 : 39	21 : 40 : 14	575
Duração Média 1	–	–	580,16
Duração Média 2	–	–	575,5

campos utilizados nessa busca, uma vez que, a diferença entre os modelos é observada apenas nos campos referentes à armazenagem em nuvem. De forma equivalente, por nessa etapa o modelo conter os dados refentes a nuvem computacional, tornou-se possível executar a terceira indagação, *"Qual o custo de executar o workflow na nuvem?"*. Seu comando e resposta podem ser vistos na imagem da mesma forma. A região obtida foi a *"região"*, uma vez que esse foi o valor padrão inicializado para a região ao inserir o modelo no MongoDB, e a quantidade de máquinas mostrou-se igual à 1.

Quais são os nomes e as versões dos programas usados no *workflow* 1?

```
db.activity3.aggregate([{$match:{"experiment_id":"1"}},
{$unwind:"$_id"},{$group:
{_id:{program:"$program_name", version:'$program_version'}}}])
```

resultados:

```
{"_id": {"program": "htseq", "version": "0.7.2"}}
{"_id": {"program": "samtools", "version": "1.4.4"}}
{"_id": {"program": "hisat2", "version": "2.0.5"}}
{"_id": {"program": "sickle", "version": "1.33"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.activity3.count({experiment_id: "1"})
```

resultados:
6

Qual a quantidade de máquinas e região no Google Cloud?

```
db.provider3.aggregate([{$lookup: {from : "cluster3",
localField: "_id", foreignField: "provider_id", as: "provider"}},
{$project:{"localization_region": 1, "provider.number_machines": 1}}])
```

resultados:

```
{"_id": "1", "localization_region": "region","provider": [{ "number_machines": "1"}]}
```

Figura 4.4: Buscas executadas no *workflow* 1 para Proveniência Referencial.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Para este workflow, estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.5. Por meio da Figura 4.5, nota-se que não foi possível criar um comando de

busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Embora fosse possível montar uma estrutura que provavelmente fosse capaz de obter tais resultados, o tamanho dos dados mostrou-se como um dificultador, pois foi o responsável pela exibição de uma mensagem de erro informando que o tamanho do documento que passaria pelas operações propostas passava dos 16MB limites. Logo, optou-se por propor duas buscas onde a primeira seria responsável por obter dados genéricos do arquivo, dentre eles, o id deste arquivo na coleção *fs.files*. Em seguida, seria realizada uma segunda busca usando este valor encontrado de forma a se obter os dados contidos nesse arquivo e demais dados presentes na coleção *fs.chunks*. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido.

Quais são os dados do arquivo de id igual à 1?

```
db.data3.aggregate([{$match:{"_id": "1"}},{$lookup:{from: "fs.files",
localField: "name", foreignField: "filename",as: "files"}}])
```

resultados:

```
{ "_id": "1", "name": "SRR5181508.fastq",
"description": "description", "localization": "localization",
"annotation": "annotation", "size": "0",
"insertion_date": "insertion_date", "machine_id": "1",
"type": "type", "oid": "5b7c9628d4fdab0783684dd2",
"files": [ { "_id": ObjectId("5b7c9628d4fdab0783684dd2"),
"chunkSize": 261120, "filename": "SRR5181508.fastq",
"length": NumberLong("6949291022"),
"uploadDate": ISODate("2018-08-21T22:46:00Z") } ] }
```

Quais os *chunks* desse arquivo?

```
db.fs.chunks.find({"files_id":ObjectId("5b7c9628d4fdab0783684dd2")}, {"_id":"1"})
```

resultados:

```
{ "_id": ObjectId("5b7c9628c93847720f9802ca") }
{ "_id": ObjectId("5b7c9628c93847720f9802cd") }
{ "_id": ObjectId("5b7c9628c93847720f9802d0") }
...
```

Figura 4.5: Buscas por dados brutos executadas no *workflow* 1 para Proveniência Referencial.

Armazenagem de Proveniência Embutida

A armazenagem de proveniência embutida explorada para este *workflow* foi construída segundo a exibida na Figura 3.5. Os tempos de execução para inserção de dados de proveniência de forma embutida podem ser vistos em Tabela 4.5 enquanto as buscas realizadas e seus resultados podem ser vistos na Figura 4.6.

Tabela 4.5: Tempos de Inserção da Proveniência Embutida para *workflow* 1.

Parâmetros	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	18 : 58 : 30	19 : 08 : 34	604
Execução 2	15 : 48 : 47	15 : 58 : 20	573
Execução 3	22 : 18 : 08	22 : 27 : 47	579
Execução 4	22 : 56 : 21	23 : 05 : 58	577
Execução 5	23 : 42 : 53	23 : 52 : 30	577
Execução 6	00 : 21 : 35	00 : 31 : 11	576
Duração Média 1	–	–	581
Duração Média 2	–	–	577,25

A tabela Tabela 4.5 é formada por quatro colunas onde a primeira, denominada *parâmetros*, é responsável pela identificação dos dados contidos em cada linha. Tais dados são distribuídos entre: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Ao passo que as Execuções 1 a 6 enumeram e representam cada vez que o *ProvTriplé* foi processado para o ambiente considerado, declarou-se que, *Duração Média 1* equivale à média padrão entre todos os resultados obtidos na última coluna e que *Duração Média 2* simboliza a média padrão desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna. A segunda coluna contém o horário da máquina da nuvem da *Google* no momento que se teve início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira coluna expõe o horário, segundo o relógio da máquina da nuvem da *Google*, em que se teve fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna salienta o tempo de duração total em segundos de cada execução do *ProvTriplé* para esse *workflow* por meio do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.6 demonstra os questionamentos feitos para esta etapa atual e seus resultados além de conter o comando que foi inserido no MongoDB para obter a resposta ao questionamento. Ao ponderar à respeito da Figura 4.6, pode-se constatar que os comandos das indagações *"Quais são os nomes e as versões dos programas usados no workflow 1?"* e *"Quantas atividades foram executadas pelo primeiro experimento?"*, assim como seus resultados, são iguais aos observados na Figura 4.2. Isso se dá devido à mesma estrutura do modelo de dados ao se tratar dos campos utilizados nessa busca, uma vez que, a diferença entre os modelos é observada apenas nos campos referentes à armazenagem em nuvem. Considerando o questionamento que visa os dados de nuvem, *"Qual o custo de executar o workflow na nuvem?"*, observa-se que a resposta obtida é análoga à equivalente exibida na Figura 4.4. Comparando-se a Figura 4.4 e a Figura 4.6, observa-se pequenas

diferenças na construção do comando proposto para responder as mesmas perguntas. Isso se deve às dessemelhanças presentes ao se comparar cada modelo entre si.

Quais são os nomes e a versão dos programas usados no *workflow* 1?

```
db.default.aggregate([{$match:{$and:[{_id:'1'},
{'experiment._id':'1'},]}}, {$unwind:'$experiment.provider.cluster.machine.activity'},
{$group:{_id:{program:'$experiment.provider.cluster.machine.activity.program_name',
version:'$experiment.provider.cluster.machine.activity.program_version'}}}])
```

resultados:

```
{'_id': {'program': 'htseq', 'version': '0.7.2'}}
{'_id': {'program': 'samtools', 'version': '1.4.4'}}
{'_id': {'program': 'hisat2', 'version': '2.0.5'}}
{'_id': {'program': 'sickle', 'version': '1.33'}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.default.aggregate([{$match:{'experiment._id': '1'}},
{$project: {numberOfActivities:
{$size:'$experiment.provider.cluster.machine.activity'}}}])
```

resultados:

```
{'_id': '1', 'numberOfActivities': 6}
```

Qual a quantidade de máquinas e região no Google Cloud?

```
db.default.find({'experiment.provider.name': 'Google Cloud',
'experiment.provider.cluster.name': 'ProvBio-BIBM'},
{_id:1, 'experiment.provider.localization_region': '1',
'experiment.provider.cluster.number_machines':1, machine:1})
```

resultados:

```
{'_id': '1', 'experiment': { 'provider': {
'localization_region': 'region','cluster': { 'number_machines': '1'}}}}
```

Figura 4.6: Buscas executadas no *workflow* 1 para Proveniência Embutida.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Para este workflow, estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.7. Por meio da Figura 4.7, nota-se que não foi possível criar um comando de busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Uma vez que o modelo de armazenagem de proveniência embutida é composto de uma única grande coleção e das coleções referentes ao GridFS, tornou-se um desafio a filtragem pelas informações que seriam exibidas no final da execução de cada comando. Sendo assim, optou-se por criar um comando que fosse capaz de encontrar qual seria o valor de *oid* de cada arquivo. Esse seria importante para, em seguida, relizar uma segunda busca de forma a se obter os dados contidos nesse arquivo e demais dados presentes na coleção *fs.chunks*. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido.

Quais são os ids e oids de cada um dos arquivos?

```
db.default.aggregate([{$unwind:'
$experiment.provider.cluster.machine.activity'}, {$group:{_id: {id:'
$experiment.provider.cluster.machine.activity.inputFiles._id',
oid:'$experiment.provider.cluster.machine.activity.inputFiles.oid' }}}])
```

resultados:

```
{ '_id': { 'id': [ '14', '15'],
'oid': [ '5c016e6cd4fdab07a932fb0f', '5c016ea8d4fdab07a932fb10'] } }
{ '_id': { 'id': [ '1'], 'oid': [ '5c016c79d4fdab07a932fb02'] } }
{ '_id': { 'id': [ '3'], 'oid': [ '5c016dced4fdab07a932fb04'] } }
{ '_id': { 'id': [ '13'], 'oid': [ '5c016e4dd4fdab07a932fb0e'] } }
{ '_id': { 'id': [ '2', '2', '4', '5', '6', '7', '8', '9', '11'],
'oid': [ '5c016d3cd4fdab07a932fb03', '5c016d3cd4fdab07a932fb03',
'5c016dd2d4fdab07a932fb05', '5c016dd2d4fdab07a932fb06', '5c016dd2d4fdab07a932fb07',
'5c016dd2d4fdab07a932fb08', '5c016dd2d4fdab07a932fb09', '5c016dd2d4fdab07a932fb0a',
'5c016dd3d4fdab07a932fb0c'] } }
{ '_id': { 'id': [ '12'], 'oid': [ '5c016dd3d4fdab07a932fb0d'] } }
```

Quais os *chunks* do arquivo 1?

```
db.fs.chunks.find({'files_id':ObjectId('5c016c79d4fdab07a932fb02')}, {'_id':'1'})
```

resultados:

```
{ '_id': ObjectId('5c016c79316caf0e9d93bd58') }
{ '_id': ObjectId('5c016c79316caf0e9d93bd5b') }
{ '_id': ObjectId('5c016c79316caf0e9d93bd5e') }
...
```

Figura 4.7: Buscas por dados brutos executadas no *workflow* 1 para Proveniência Embutida.

Armazenagem de Proveniência Híbrida

A armazenagem de proveniência referencial híbrida para este *workflow* foi construída segundo a exibida na Figura 3.6. Os tempos de execução para inserção de dados de proveniência no modelo híbrido podem ser vistos em Tabela 4.6. As buscas realizadas e seus resultados podem ser vistos em Figura 4.8.

A tabela Tabela 4.6 é constituída por quatro colunas: *Parâmetros*, *Tempo Inicial*, *Tempo Final* e *Duração*. A coluna *Parâmetros* identifica os dados contidos em cada linha. Tais dados podem ser repartidos entre as seguintes denominações: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Cada uma das seis execuções do *ProvTriplé* consideradas possuem seus dados representados pelos parâmetros *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5* e *Execução 6*. Os parâmetros *Duração Média 1* e *Duração Média 2* representam as médias padrões calculadas usando os dados obtidos na coluna *Duração* como base. Ao passo que *Duração Média 1* equivale à média padrão entre todos os resultados obtidos, a *Duração Média 2* simboliza a média padrão desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna marcados em segundos. A segunda coluna, *Tempo*

Tabela 4.6: Tempos de Inserção da Proveniência Híbrida para *workflow* 1.

Parâmetro	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	18 : 48 : 55	18 : 58 : 30	575
Execução 2	15 : 00 : 23	15 : 09 : 59	576
Execução 3	21 : 35 : 18	21 : 44 : 53	575
Execução 4	00 : 56 : 02	01 : 05 : 40	578
Execução 5	12 : 37 : 56	12 : 47 : 34	578
Execução 6	01 : 33 : 23	01 : 42 : 58	575
Duração Média 1	–	–	576,16
Duração Média 2	–	–	576

Inicial, contém o horário que, pelo relógio da máquina da nuvem da *Google*, teve-se início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira coluna, *Tempo Final*, expõe o horário que, segundo o relógio da máquina da nuvem da *Google*, teve-se fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna contém o tempo de duração total em segundos de cada execução por meio do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.8 apresenta questionamentos feitos para esta etapa atual e os resultados obtidos para cada um. Primeiramente é exibido a pergunta feita, em seguida, o comando inserido no MongoDB para obter a resposta à indagação e, logo após, os resultados obtidos. Ao ponderar à respeito da Figura 4.8, pode-se constatar que os comandos das indagações *"Quais são os nomes e as versões dos programas usados no workflow 1?"* e *"Quantas atividades foram executadas pelo primeiro experimento?"*, assim como seus resultados, são iguais aos observados na Figura 4.3. Isso se dá devido à mesma estrutura do modelo de dados ao se tratar dos campos utilizados nessa busca, uma vez que, a diferença entre os modelos é observada apenas nos campos referentes ao armazenamento em nuvem. Considerando o questionamento que visa os dados de nuvem, *"Qual o custo de executar o workflow na nuvem?"*, observa-se que a resposta obtida é análoga à equivalente exibida na Figura 4.4 e Figura 4.6. Comparando-se a Figura 4.4, Figura 4.6 e a Figura 4.8, observa-se pequenas diferenças na construção do comando proposto para responder as mesmas perguntas. Isso se deve às dessemelhanças presentes ao se comparar cada modelo entre si.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.9. Através da Figura 4.9, nota-se que não foi possível criar um comando de busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Embora fosse possível montar uma estrutura que provavelmente fosse capaz de obter tais

Quais são os nomes e a versão dos programas usados no *workflow* 1?
db.project1.aggregate([{$match:{$and:[{_id:'1'}, {"experiment._id":'1'},]}}, {$unwind:"$experiment.activity"}, {$group:{_id: {program:"$experiment.activity.program_name", version:'$experiment.activity.program_version'}}}])
resultados:
{"_id": {"program": "htseq", "version": "0.7.2"}}
{"_id": {"program": "samtools", "version": "1.4.4"}}
{"_id": {"program": "hisat2", "version": "2.0.5"}}
{"_id": {"program": "sickle", "version": "1.33"}}
Quantas atividades foram executadas pelo primeiro experimento?
db.project1.aggregate([{$match:{$and:[{id:'1'}, {"experiment.id":'1'},]}}, {$unwind:"$experiment.activity"}, {$group:{_id:{program:"$experiment.activity.program_name", version:'$experiment.activity.program_version'}}}])
resultados:
{"_id": "1", "numberOfActivities": 6}
Qual a quantidade de máquinas e região no Google Cloud?
db.provider1.find({"name": "Google Cloud", "cluster.name": "ProvBio-BIBM"}, {_id: "1", "localization_region": 1 , "cluster.number_machines":1, machine:1})
resultados:
{"_id": "1", "localization_region": "region","cluster": { "number_machines": "1"}}

Figura 4.8: Buscas executadas *workflow* 1 para Proveniência Híbrida.

resultados, o tamanho dos dados mostrou-se como um dificultador, pois foi o responsável pela exibição de uma mensagem de erro informando que o tamanho do documento que passaria pelas operações propostas passava dos 16MB. Logo, optou-se por propor duas buscas onde a primeira seria responsável por obter dados do arquivo, dentre eles, o id deste arquivo na coleção fs.files. Em seguida, seria realizada uma segunda busca usando este valor encontrado de forma a se obter os dados contidos nesse arquivo. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido.

4.1.3 *workflow* 2

Os resultados obtidos de cada execução do *workflow* 2 em um ambiente de nuvem podem ser vistos a seguir na seguinte ordem: proveniência referencial, proveniência embutida e proveniência híbrida. A quantidade de tempo que cada busca levou para ser realizada não foi calculado para nenhum caso uma vez que esta se mostrou demasiadamente pequena para ser comparada.

Quais são os dados do arquivo de id igual à 1?

```
db.data1.aggregate([{$match:{"_id": "1"}},{$lookup:{from: "fs.files",
localField: "name", foreignField: "filename",as: "files"}}])
```

resultados:

```
{ "_id": "1", "name": "SRR5181508.fastq",
"description": "description", "localization": "localization",
"annotation": "annotation", "size": "0",
"insertion_date": "insertion_date", "machine_id": "1",
"type": "type", "oid": "5b7c8c02d4fdab06342c3a22",
"files": [ { "_id": ObjectId("5b7c8c02d4fdab06342c3a22"),
"chunkSize": 261120, "filename": "SRR5181508.fastq",
"length": NumberLong("6949291022"),
"uploadDate": ISODate("2018-08-21T22:46:00Z") } ] }
```

Quais os *chunks* desse arquivo?

```
db.fs.chunks.find({"files_id":ObjectId("5b7c8c02d4fdab06342c3a22")}, {"_id":"1"})
```

resultados:

```
{ "_id": ObjectId("5b7c8c02c93847720f94c3f7") }
{ "_id": ObjectId("5b7c8c02c93847720f94c3fa") }
{ "_id": ObjectId("5b7c8c02c93847720f94c3fd") }
...
```

Figura 4.9: Buscas por dados brutos executadas no *workflow* 1 para Proveniência Híbrida.

Armazenagem de Proveniência Referencial

A armazenagem de proveniência referencial explorada para este *workflow* foi construída segundo a exibida na Figura 3.4. Os tempos de execução para inserção de dados de proveniência de forma referencial podem ser vistos em Tabela 4.7. Do mesmo modo como no caso do *workflow* anterior, as buscas realizadas e seus resultados podem ser vistas em Figura 4.10.

A tabela Tabela 4.7 é composta por quatro colunas denominadas *Parâmetros*. *Tempo Inicial*, *Tempo Final* e *Duração*. A primeira coluna contém os parâmetros para identificação dos dados contidos em cada linha sendo eles divididos em: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Aqueles que representam execuções (de *Execução 1* a *Execução 6*) enumeram e representam cada vez que o *ProvTriplé* foi processado. Por outro lado, considerou-se que, *Duração Média 1* simboliza a média considerando todos os resultados e que *Duração Média 2* constitui a média desconsiderando-se o maior e o menor resultado. A segunda coluna representa o horário do relógio da máquina da nuvem da *Google* no momento que teve início a inserção de dados no MongoDB. De forma análoga, a terceira coluna expõe o horário, segundo o relógio da máquina da nuvem da *Google*, em que se teve fim a introdução dos dados do modelo de proveniência referencial. Ambos os horários são representados no formato hh:mm:ss onde 'h' representa *hora*, 'm' simboliza *minutos* e 's'

Tabela 4.7: Tempos de Inserção da Proveniência Referencial para *workflow* 2.

Parâmetro	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	17 : 55 : 57	17 : 57 : 07	70
Execução 2	19 : 57 : 02	19 : 58 : 11	69
Execução 3	20 : 12 : 25	20 : 15 : 24	179
Execução 4	20 : 27 : 03	20 : 28 : 14	71
Execução 5	20 : 46 : 50	20 : 48 : 02	72
Execução 6	21 : 03 : 56	21 : 05 : 06	70
Duração Média 1	–	–	88,5
Duração Média 2	–	–	70,75

os *segundos*. A última coluna salienta o tempo de duração total em segundos de cada execução considerando-se a diferença entre o tempo inicial (coluna dois) e o tempo final (coluna três). Esses dados também são os usados no cálculo de média para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

Na Figura 4.10 é possível observar os questionamentos feitos, o comando inserido no MongoDB para obter a resposta e os resultados obtidos. Ao analisar a Figura 4.10, pode-se constatar que embora os comandos utilizados para responder aos questionamentos sejam semelhantes aos usados na Figura 4.4, os resultados não o são uma vez que eles representam as informações relativas ao *workflow* 2. A construção similar dos comandos se dá devido à mesma estrutura do modelo de dados. Para a primeira indagação, *"Quais são os nomes e as versões dos programas usados no Workflow 1?"*, obtêm-se a informação de que foram usados os programas *sickle* e *abyss* nas versões 1.33 e 1.3.6 respectivamente. Na segunda pergunta, *"Quantas atividades foram executadas pelo primeiro experimento?"*, o resultado alcançado indica que o *workflow* em questão possui 3 atividades, ou seja, etapas. Por último, uma vez que as configurações de nuvem não foram alteradas, ao visar a adquirir a resposta do quesionamento, *"Qual o custo de executar o workflow na nuvem?"*, o resultado é o mesmo obtido para os casos anteriores ilustrados na Figura 4.4, Figura 4.6 e Figura 4.8.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Para este workflow, estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.11. Por meio da Figura 4.11, nota-se que não foi possível criar um comando de busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Embora fosse possível montar uma estrutura que provavelmente fosse capaz de obter tais resultados, o tamanho dos dados mostrou-se como um dificultador, pois foi o responsável pela exibição de uma mensagem de erro informando que o tamanho do documento que passaria pelas operações propostas passava dos 16MB limites. Logo, optou-se por propor duas buscas onde a primeira seria responsável por obter dados genéricos do arquivo, dentre eles, o id deste arquivo na coleção *fs.files*. Em seguida, seria realizada

Quais são os nomes e as versões dos programas usados no *workflow* 2?

```
db.activity3.aggregate([{$match:{"experiment_id":"1"}},
{$unwind:"$_id"},{$group:
{_id:{program:"$program_name", version:'$program_version'}}}])
```

resultados:

```
{"_id": {"program": "sickle", "version": "1.33"}}
{"_id": {"program": "abyss", "version": "1.3.6"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.activity3.count({experiment_id: "1"})
```

resultados:

```
3
```

Qual a quantidade de máquinas e região no Google Cloud?

```
db.provider3.aggregate([{$lookup: {from : "cluster3",
localField: "_id", foreignField: "provider_id", as: "provider"}},
{$project:{"localization_region": 1, "provider.number_machines": 1}}])
```

resultados:

```
{"_id": "1", "localization_region": "region","provider": [{ "number_machines": "1"}]}
```

Figura 4.10: Buscas executadas no *workflow* 2 para Proveniência Referencial.

uma segunda busca usando este valor encontrado de forma a se obter os dados contidos nesse arquivo e demais dados presentes na coleção *fs.chunks*. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido. As buscas e os resultados encontrados, exibidos na Figura 4.11 se diferem dos exibidos na Figura 4.5 apenas nos dados encontrados nos resultados, pois eles são particulares de cada arquivo considerado como exemplo de utilização em cada *workflow*.

Armazenagem de Proveniência Embutida

A armazenagem de proveniência embutida explorada para este *workflow* foi construída segundo a exibida na Figura 3.5. Os tempos de execução para inserção de dados de proveniência de forma embutida podem ser vistos em Tabela 4.8 enquanto as buscas realizadas podem ser vistas em Figura 4.12.

A tabela Tabela 4.8 é formada por quatro colunas onde a primeira, denominada *parâmetros*, é responsável pela identificação dos dados contidos em cada linha. Tais dados são distribuídos entre: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Ao passo que as Execuções 1 a 6 enumeram e representam cada vez que o *ProvTriplé* foi processado para o ambiente considerado, declarou-se que, *Duração Média 1* equivale à média padrão entre todos os resultados obtidos na última coluna e que *Duração Média 2* simboliza a média padrão desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna. A segunda coluna

Quais são os dados do arquivo de id igual à 1?

```
db.data3.aggregate([{$match:{"_id": "1"}},{$lookup:{from: "fs.files",
localField: "name", foreignField: "filename",as: "files"}}])
```

resultados:

```
{ "_id": "1", "name": "ERR885455_1.fastq",
"description": "description", "localization": "localization",
"annotation": "annotation", "size": "0", "insertion_date": "insertion_date",
"machine_id": "1", "type": "type", "oid": "5c018144d4fdab0ac8334af3",
"files": [ { "_id": ObjectId("5c018144d4fdab0ac8334af3"),
"chunkSize": 261120, "filename": "ERR885455_1.fastq",
"length": NumberLong(542058524),
"uploadDate": ISODate("2018-11-30T18:28:20Z") } ] }
```

Quais os *chunks* desse arquivo?

```
db.fs.chunks.find({"files_id":ObjectId("5c018144d4fdab0ac8334af3")}, {"_id":"1"})
```

resultados:

```
{ "_id": ObjectId("5c018144316caf0e9d97b41b") }
{ "_id": ObjectId("5c018144316caf0e9d97b41e") }
{ "_id": ObjectId("5c018144316caf0e9d97b421") }
...
```

Figura 4.11: Buscas por dados brutos executadas no *Workflow* 2 para Proveniência Referencial.

contém o horário da máquina da nuvem da *Google* no momento que se teve início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira coluna expõe o horário, segundo o relógio da máquina da nuvem da *Google*, em que se teve fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna salienta o tempo de duração total em segundos de cada execução do *Prov-Triplé* para esse *workflow* por meio do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.12 demonstra os questionamentos feitos para esta etapa atual e seus resultados além de conter o comando que foi inserido no MongoDB para obter a resposta ao questionamento. Ao analisar a Figura 4.10, pode-se constatar que embora os comandos utilizados para responder aos questionamentos sejam semelhantes aos usados na Figura 4.6, os resultados não o são uma vez que eles representam as informações relativas ao *workflow* 2. A construção similar dos comandos se dá devido à mesma estrutura do modelo de dados. Também é possível observar uma pequena diferença na construção dos comandos ao comparar-se com a Figura 4.10. Tal distinção é ocasionada devido as dessemelhanças entre os modelos de proveniência relacional e embutida. Para as indagações, *"Quais são os nomes e as versões dos programas usados no Workflow 1?"*, *"Quantas*

Tabela 4.8: Tempos de Inserção da Proveniência Embutida para *workflow* 2.

#	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	17 : 54 : 54	17 : 55 : 57	63
Execução 2	19 : 56 : 00	19 : 57 : 02	62
Execução 3	20 : 11 : 27	20 : 12 : 25	58
Execução 4	20 : 26 : 02	20 : 27 : 03	61
Execução 5	20 : 45 : 49	20 : 46 : 50	61
Execução 6	21 : 02 : 51	21 : 03 : 56	65
Duração Média 1	–	–	61,67
Duração Média 2	–	–	61,75

atividades foram executadas pelo primeiro experimento?" e *"Qual o custo de executar o workflow na nuvem?"*, foram atingidos resultados análogos aos exibidos naFigura 4.10.

Quais são os nomes e a versão dos programas usados no *workflow* 2?

```
db.default.aggregate([{$match:{$and:[{_id:"1"},
{"experiment._id":"1"},]}}, {$unwind:"$experiment.provider.cluster.machine.activity"},
{$group:{_id:{program:"$experiment.provider.cluster.machine.activity.program_name",
version:'$experiment.provider.cluster.machine.activity.program_version'}}}])
```

resultados:

```
{"_id": {"program": "sickle", "version": "1.33"}}
{"_id": {"program": "abyss", "version": "1.3.6"}}
```

Quantas atividades foram executadas pelo primeiro experimento?

```
db.default.aggregate([{$match:{"experiment._id": "1"}},
{$project: {numberOfActivities:
{$size:"$experiment.provider.cluster.machine.activity"}}}])
```

resultados:

```
{"_id": "1", "numberOfActivities": 3}
```

Qual a quantidade de máquinas e região no Google Cloud?

```
db.default.find({"experiment.provider.name": "Google Cloud",
"experiment.provider.cluster.name": "ProvBio-BIBM"},
{_id:1, "experiment.provider.localization_region": "1",
"experiment.provider.cluster.number_machines":1, machine:1})
```

resultados:

```
{"_id": "1","experiment": { "localization_region": "region","cluster": { "number_machines": "1"}}}
```

Figura 4.12: Buscas executadas no *workflow* 2 para Proveniência Embutida.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Para este workflow, estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.13. Por meio da Figura 4.13, nota-se que não foi possível criar um comando de busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Uma vez que o modelo de armazenagem de proveniência embutida é composto de uma única grande coleção e das coleções referentes ao GridFS, tornou-se um desafio a filtragem pelas informações que seriam exibidas no final da execução de cada comando. Sendo assim, optou-se por criar um comando que fosse capaz de encontrar qual seria o valor de *oid* de cada arquivo. Esse seria importante para, em seguida, relizar uma segunda

busca de forma a se obter os dados contidos nesse arquivo e demais dados presentes na coleção *fs.chunks*. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido. Os resultados exibidos na Figura 4.13 se diferem dos que podem ser vistos na Figura 4.7 apenas nas características únicas dos arquivos usados como exemplos nas buscas. De maneira semelhante, os comandos são os mesmos.

Quais são os ids e oids de cada um dos arquivos?

```
db.default.aggregate([{$unwind:"
$experiment.provider.cluster.machine.activity"}, {$group:{_id: {id:"
$experiment.provider.cluster.machine.activity.inputFiles._id",
oid:"$experiment.provider.cluster.machine.activity.inputFiles.oid"
}}}])
```

resultados:

```
{ "_id": { "id": [ "2", "4"],
"oid": [ "5c018114d4fdab0ac8334a76", "5c018130d4fdab0ac8334a78"] } }
{ "_id": { "id": [ "3"],
"oid": [ "5c01811dd4fdab0ac8334a77"] } }
{ "_id": { "id": [ "1"],
"oid": [ "5c018107d4fdab0ac8334a75"] } }
```

Quais os *chunks* do arquivo 1?

```
db.fs.chunks.find({"files_id":ObjectId("5c018107d4fdab0ac8334a75")}, {"_id":"1"})
```

resultados:

```
{ "_id": ObjectId("5c018107316caf0e9d9758d4") }
{ "_id": ObjectId("5c018107316caf0e9d9758d7") }
{ "_id": ObjectId("5c018107316caf0e9d9758da") }
...
```

Figura 4.13: Buscas por dados brutos executadas no *Workflow* 2 para Proveniência Embutida.

Armazenagem de Proveniência Híbrida

A armazenagem de proveniência híbrida explorada para este *workflow* foi construída segundo a exibida na Figura 3.6. Os tempos de execução para inserção de dados de proveniência na modelagem híbrida podem ser vistos em Tabela 4.9.Enquanto as buscas realizadas e seus resultados podem ser vistos em Figura 4.14.

A tabela Tabela 4.9 é constituída por quatro colunas: *Parâmetros*, *Tempo Inicial*, *Tempo Final* e *Duração*. A coluna *Parâmetros* identifica os dados contidos em cada linha. Tais dados podem ser repartidos entre as seguintes denominações: *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5*, *Execução 6*, *Duração Média 1* e *Duração Média 2*. Cada uma das seis execuções do *ProvTriplé* consideradas possuem seus dados representados pelos parâmetros *Execução 1*, *Execução 2*, *Execução 3*, *Execução 4*, *Execução 5* e

Tabela 4.9: Tempos de Inserção da Proveniência Híbrida para *workflow* 2.

Parâmetro	**Tempo Inicial**	**Tempo Final**	**Duração (segundos)**
Execução 1	17 : 54 : 10	17 : 54 : 54	44
Execução 2	19 : 55 : 18	19 : 56 : 00	42
Execução 3	20 : 10 : 43	20 : 11 : 27	44
Execução 4	20 : 25 : 19	20 : 26 : 02	43
Execução 5	20 : 45 : 06	20 : 45 : 49	43
Execução 6	21 : 02 : 09	21 : 02 : 51	42
Duração Média 1	–	–	43
Duração Média 2	–	–	43

Execução 6. Os parâmetros *Duração Média 1* e *Duração Média 2* representam as médias padrões calculadas usando os dados obtidos na coluna *Duração* como base. Ao passo que *Duração Média 1* equivale à média padrão entre todos os resultados obtidos, a *Duração Média 2* simboliza a média padrão desconsiderando-se o maior e o menor resultado dentre os valores obtidos na última coluna marcados em segundos. A segunda coluna, *Tempo Inicial*, contém o horário que, pelo relógio da máquina da nuvem da *Google*, teve-se início a etapa de inserção dos dados de proveniência no MongoDB. Semelhantemente, a terceira coluna, *Tempo Final*, expõe o horário que, segundo o relógio da máquina da nuvem da *Google*, teve-se fim a introdução dos dados. Ambos os horários são representados no formato hh:mm:ss. A última coluna contém o tempo de duração total em segundos de cada execução por meio do cálculo da diferença entre o tempo inicial, marcado na coluna dois, e o tempo final, representado na coluna três. Esses dados, assim como explicado anteriormente, são os usados no cálculo de média padrão para obtenção dos parâmetros *Duração Média 1* e *Duração Média 2* da primeira coluna.

A Figura 4.14 apresenta questionamentos feitos para esta etapa atual e os resultados obtidos para cada um. Primeiramente é exibido a pergunta feita, em seguida, o comando inserido no MongoDB para obter a resposta à indagação e, logo após, os resultados obtidos. Ao ponderar-se à respeito da Figura 4.10, pode-se perceber que, embora os comandos utilizados para responder aos questionamentos sejam semelhantes aos usados na Figura 4.14, os resultados não o são uma vez que eles representam as informações relativas ao *workflow* 2. A construção similar dos comandos dá-se devido à mesma estrutura do modelo de dados. Também é possível observar uma pequena diferença na construção dos comandos ao comparar-se com a Figura 4.10 e Figura 4.12. Tal distinção é ocasionada devido as dessemelhanças entre os modelos de proveniência relacional e embutida. Para as indagações, *"Quais são os nomes e as versões dos programas usados no Workflow 1?"*, *"Quantas atividades foram executadas pelo primeiro experimento?"* e *"Qual o custo de executar o workflow na nuvem?"*, foram atingidos resultados análogos aos exibidos naFigura 4.10 e na Figura 4.12.

Quais são os nomes e a versão dos programas usados no *workflow* 2?
db.project1.aggregate([{$match:{$and:[{_id:"1"}, {"experiment._id":"1"},]}}, {$unwind:"$experiment.activity"}, {$group:{_id: {program:"$experiment.activity.program_name", version:'$experiment.activity.program_version'}}}])
resultados: {"_id": {"program": "sickle", "version": "1.33"}} {"_id": {"program": "abyss", "version": "1.3.6"}}
Quantas atividades foram executadas pelo primeiro experimento?
db.project1.aggregate([{$match:{$and:[{id:"1"}, {"experiment.id":"1"},]}}, {$unwind:"$experiment.activity"}, {$group:{_id:{program:"$experiment.activity.program_name", version:'$experiment.activity.program_version'}}}])
resultados: {"_id": "1", "numberOfActivities": 3}
Qual a quantidade de máquinas e região no Google Cloud?
db.provider1.find({"name": "Google Cloud", "cluster.name": "ProvBio-BIBM"}, {_id: "1", "localization_region": 1 , 'cluster.number_machines':1, machine:1})
resultados: {"_id": "1", "localization_region": "region","cluster": { "number_machines": "1"}}

Figura 4.14: Buscas executadas no *workflow* 2 para Proveniência Embutida.

Adicionalmente, optou-se por executar também buscas que envolvessem os dados brutos. Estas podem ser vistas com seus respectivos comandos e resultados na Figura 4.15. Através da Figura 4.15, nota-se que não foi possível criar um comando de busca único que pudesse, desde da coleção de dados, alcançar o nível dos *chunks* dentro do GridFS. Embora fosse possível montar uma estrutura que provavelmente fosse capaz de obter tais resultados, o tamanho dos dados mostrou-se como um dificultador, pois foi o responsável pela exibição de uma mensagem de erro informando que o tamanho do documento que passaria pelas operações propostas passava dos 16MB. Logo, optou-se por propor duas buscas onde a primeira seria responsável por obter dados do arquivo, dentre eles, o id deste arquivo na coleção fs.files. Em seguida, seria realizada uma segunda busca usando este valor encontrado de forma a se obter os dados contidos nesse arquivo. Por questões de visualização, o resultado foi filtrado de modo à mostrar apenas o id dos *chunks*. O tempo que as buscas levaram para serem concluídas foi pequeno demais para ser medido. Os comandos exibidos na Figura 4.15 são os mesmos exibidos na Figura 4.9 embora os resultados se diferenciem de acordo com as características únicas de cada arquivo usado como exemplo dos testes.

Quais são os dados do arquivo de id igual à 1?

```
db.data1.aggregate([{$match:{"_id": "1"}},{$lookup:{from: "fs.files",
localField: "name", foreignField: "filename",as: "files"}}])
```

resultados:


```
{ "_id": "1", "name": "ERR885455_1.fastq",
"description": "description",
"localization": "localization", "annotation": "annotation", "size": "0",
"insertion_date": "insertion_date", "machine_id": "1", "type": "type",
"oid": "5c0180d9d4fdab0ac8334a22",
"files": [ { "_id": ObjectId("5c0180d9d4fdab0ac8334a22"), "chunkSize": 261120,
"filename": "ERR885455_1.fastq", "length": NumberLong(542058524),
"uploadDate": ISODate("2018-11-30T18:26:33Z") } ] }
```


Quais os *chunks* desse arquivo?

```
db.fs.chunks.find({"files_id":ObjectId("5c0180d9d4fdab0ac8334a22")}, {"_id":"1"})
```

resultados:


```
{ "_id": ObjectId("5c0180d9316caf0c9d96fc0f") }
{ "_id": ObjectId("5c0180d9316caf0c9d96fc12") }
{ "_id": ObjectId("5c0180d9316caf0c9d96fc15") }
...
```


Figura 4.15: Buscas por dados brutos executadas no *Workflow* 2 para Proveniência Híbrida.

4.2 Comparação de armazenagens

Os tempos médios de inserção de cada modelo podem ser observados para comparação em Figura 4.16, Figura 4.17 e Figura 4.18. Primeiramente são expostas as médias considerando a execução do *workflow* 1 em ambiente local. Em seguida, a média para o mesmo *workflow* em ambiente de nuvem e, posteriormente, a média do *workflow* 2 em ambiente de nuvem. Na Figura 4.16,Figura 4.17 e Figura 4.18, o eixo x do gráfico especifica se o valor demonstrado é referente ao modelo de armazenagem NoSQL baseado em documento utilizado de forma referencial (Modelo Referencial), ou empregado de maneira embutida (Modelo Embutido) ou híbrida (Modelo Híbrido). O eixo y representa, em segundos, o tempo de Duração Média (1 e 2) de acordo com a respectiva tabela do caso em questão.

Na Figura 4.16, compara-se o tempo médio para criação dos modelos de proveniência embutida, referencial e híbrida do *workflow* 1 quando executado localmente. Para a coluna *Proveniência Referencial*, os valores da Duração Média 1 e 2 são aqueles apresentados na Tabela 4.1. De forma análoga, os dados expostos na coluna *Proveniência Embutida* são os expressos na Tabela 4.2 e os mostrados na coluna *Proveniência Híbrida*, os provenientes da duração média 1 e 2 da Tabela 4.3.

Em Figura 4.17, compara-se o tempo médio para criação dos modelos de proveniência embutida, referencial e híbrida do *workflow* 1 quando executado na nuvem. A clara dife-

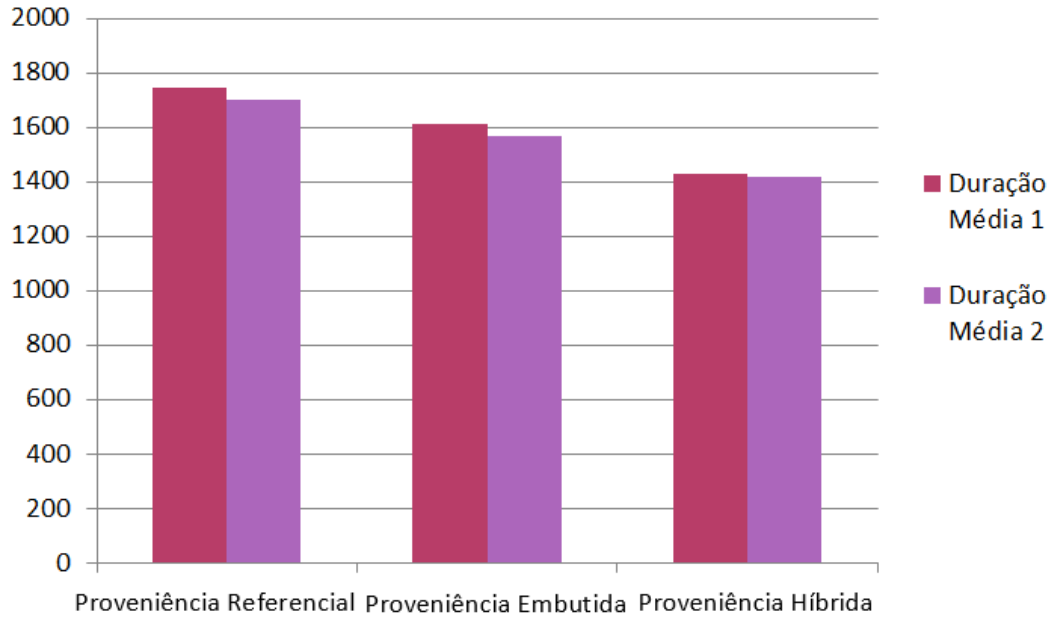

Figura 4.16: Comparação de Médias de Tempo de Inserção entre modelos para *workflow* 1 em ambiente local.

rença de performance entre a execução local e na nuvem deve-se a quantidade diferenciada de RAM entre o ambiente de execução local (1,8 GB) e do ambiente de nuvem (3GB). Tal discrepância não é salientada como forma de comparar a performance entre os modelos, mas apenas como forma de explicar o motivo de sua ocorrência especialmente considerando que a baixa performance em ambiente local, decorrente da menor quantidade de memória RAM, tornou-se um problema e, por isso, foram encontradas e implementadas alternativas para evitar empecilhos decorrentes deste detalhe uma vez que permaceu claro que uma grande quantidade de memória do sistema estava sendo utilizada ocasionando em dificuldades para chegar ao término de cada execução do *ProvTriplé*. A Figura 4.17 apresenta para a coluna *Proveniência Referencial* os valores da Duração Média 1 e 2 de acordo com os dados recolhidos na Tabela 4.4. De forma análoga, os dados expostos na coluna *Proveniência Embutida* são os expressos na Tabela 4.5 e os mostrados na coluna *Proveniência Híbrida*, os provenientes dos cálculos das Duração Média 1 e 2 da Tabela 4.6.

Em Figura 4.18, compara-se o tempo médio para criação dos modelos de proveniência embutida, referencial e híbrida do *workflow* 2 quando executado na nuvem. Para a coluna *Proveniência Referencial*, os valores da Duração Média 1 e 2 são aqueles apresentados na Tabela 4.7. De forma análoga, os dados expostos na coluna *Proveniência Embutida* são os expressos na Tabela 4.8 e os mostrados na coluna *Proveniência Híbrida*, os provenientes da duração média 1 e 2 da Tabela 4.9.

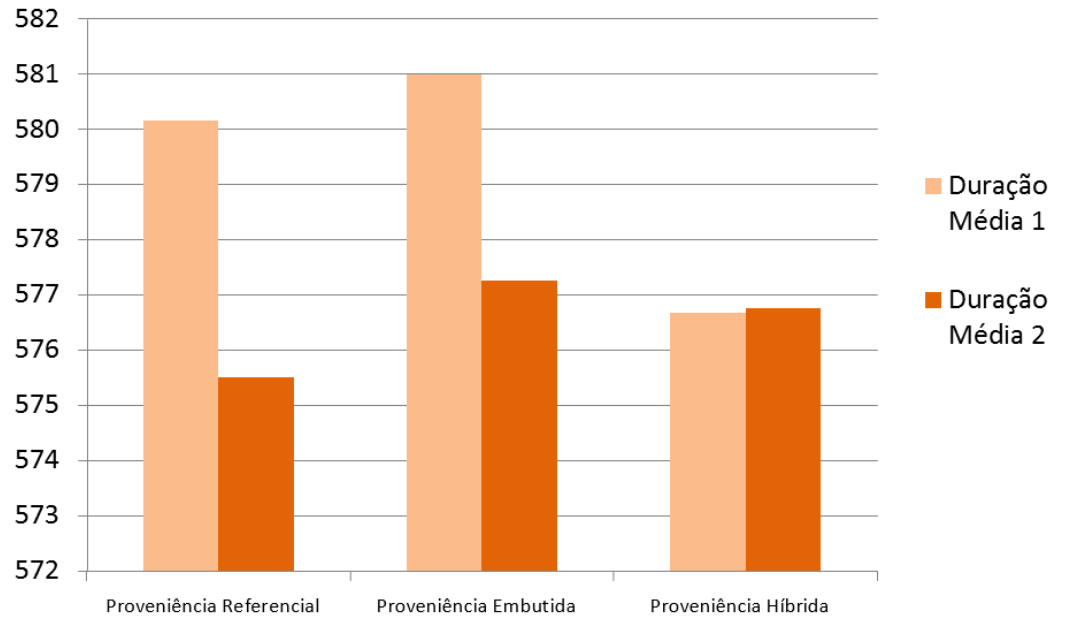

Figura 4.17: Comparação de Médias de Tempo de Inserção entre modelos para *workflow* 1 em nuvem.

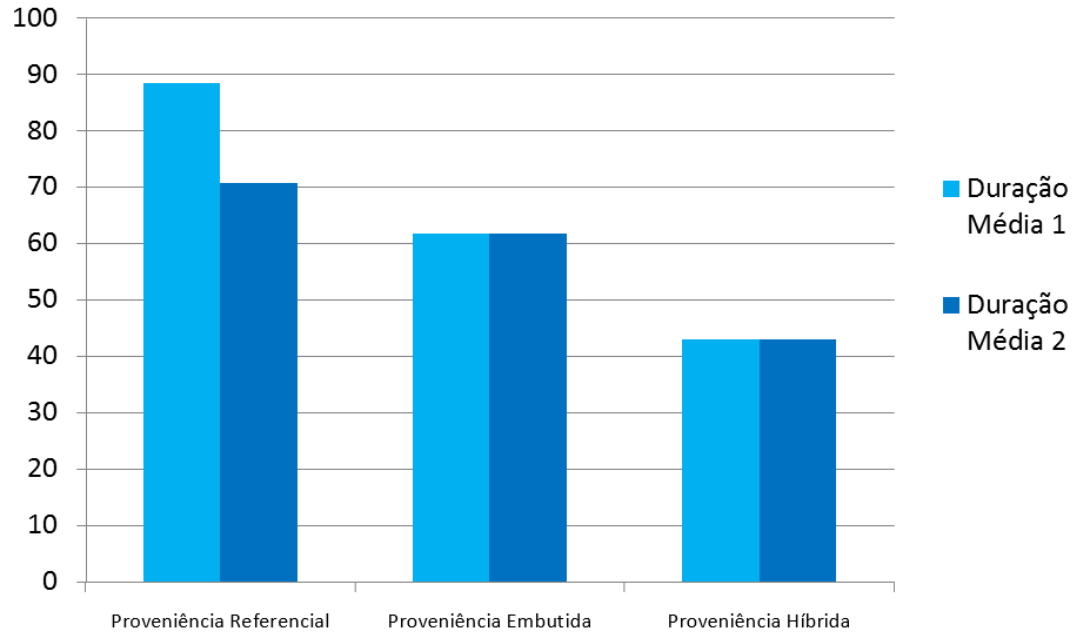

Figura 4.18: Comparação de Médias de Tempo de Inserção entre modelos para *workflow* 2 em nuvem.

Por meio dos gráficos é mais claro observar-se que, em média, de acordo com os resultados obtidos para os *Workflows* testados, a modelagem híbrida insere os dados em menos tempo embora exista uma flutuação dos valores calculados para a Duração Média da *Proveniência Referencial* para o *workflow* 1 quando executado na nuvem. Contudo, a diferença de tempo entre as execuções em ambiente de nuvem para ambos os *workflows* é pequena, pois os dados que ocupam mais espaço e teriam maior potencial de causar uma maior discrepância no tempo, ou seja, os dados brutos, são todos armazenados da mesma maneira, com o *GridFS*. Tal inserção, além de não alterar a forma que os dados brutos estão presentes em cada um dos modelos, faz com que as maiores diferenças sejam causadas pelos dados realmente de proveniência que são muito pequenos quando comparados aos dados brutos.

4.3 Resultados Acadêmicos

No decorrer do desenvolvimento deste trabalho surgiram oportunidades de submete-lo para conferências intercionais em formato de *full-paper*, obtendo os seguintes aceites:

- LOPES, INGRID SANTANA; HOLANDA, MARISTELA. Biological database in NoSQL document-oriented database. 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018, Cáceres. DOI: 10.23919/CISTI.2018.8399168 [62]
- LOPES, INGRID SANTANA; HOLANDA, MARISTELA. NoSQL Model Evaluation With Bioinformatic Provenance Data. International Conference on Bioinformatics and Biomedicine 2018 (BIBM), 2018, Madrid.

Capítulo 5

Conclusão e Trabalhos Futuros

A grande quantidade de dados gerados pela Bioinformática, especialmente no campo da biologia molecular, apresentou novos desafios e demandas. Tais desafios e demandas criados, não só pela bioinformática, mas também por outras áreas, tornaram importante o uso do NoSQL. Sendo o conceito de proveniência de dados tradicionalmente aplicado no contexto de *workflows*, este trabalho apresentou uma análise de implementações de modelos de dados baseados em documento com base em *workflows* de Bioinformática implementando o conceito de proveniência de dados. Foi possível criar e analisar três variações de modelos de armazenagem de proveniência no MongoDB utilizando os conceitos de documentos referenciados, embutidos e híbridos. A inserção de dados teve o auxílio da utilização do ambiente de nuvem computacional e teve o amparo do programa criado neste trabalho.

O tamanho dos arquivos de bioinformática podem representar uma complicação ao se considerar a inserção de dados brutos em algum banco de dados. Entretanto, a inserção de dados brutos de bioinformática tornou-se viável ao utilizar o conceito do MongoDB de GridFS. Dessa forma, foi possível ultrapassar a problemática do limite de 16MB para os documentos do MongoDB apesar de alguns arquivos de bioinformática se mostrarem maiores do que isso.

Observou-se que, em relação aos resultados de análises de tempo de inserção e busca entre os modelos, em média, de acordo com as conclusões obtidas para os *workflows* testados, a modelagem híbrida insere os dados em menos tempo. Contudo, a variação é pequena, pois os dados que poderiam ocasionar uma diversidade mais acenturada no tempo calculado devido ao seu tamanho, ou seja, os dados brutos, são todos armazenados da mesma maneira, com o GridFS. Tal inserção, além de não alterar a forma que os dados brutos estão presentes em cada um dos modelos, faz com que as maiores diferenças sejam causadas pelos dados realmente de proveniência, que são muito pequenos quando comparados aos dados brutos. Ademais, a distinção entre os tempos de inserção

também é causada devido a maneira pela qual os documentos associam os dados que se correlacionam, ou seja, de maneira relacional, embutida ou híbrida.

Quanto as buscas, pode-se notar que, existem pequenas alterações que se fazem necessárias na elaboração do comando que será utilizado para responder as perguntas propostas de acordo com qual modelagem está sendo usada. Dependendo da modelagem utilizada, seja ela referencial, embutida ou híbrida, é possível que a construção de algum comando se mostre mais fácil do que do outro. Do mesmo modo, também é possível que aconteça o oposto, isso é, comandos de concepção mais difícil. Em contra partida, por mais simples que as construções de buscas no modelo embutido se mostrem em alguns casos, elas também podem se tornar rapidamente longas ao ter que descrever no comando todo o caminho até a parte mais profundamente embutida do documento caso ela contenha a resposta do questionamento em questão.

As análises feitas nesse trabalho podem servir para propor modelos de armazenagem ideais para cada caso e também para sugerir alternativas de métodos e implementações capazes de facilitar a execução de um *workflows* de bioinformática e obtenção de seus dados relevantes.

Trabalhos futuros incluirão a execução de outros *workflows* visando analisar os resultados obtidos para as três armazenagens propostas com intuíto de considerar se os comportamentos são parecidos, podendo também conter o aperfeiçoamento das funcionalidades e comportamento do programa criado, além de aperfeiçoamento dos comandos executados nas buscas, especialmente aqueles aplicados sobre os dados brutos e também questionamentos mais complexos. Demais trabalhos futuros poderão considerar também estudos de novas propostas de modelagens e até mesmo análises semelhantes quanto ao armazenamento de dados de proveniência em bancos de dados NoSQL em outros sistemas de gerenciamento de banco de dados baseados em documento que não sejam o MongoDB ou até mesmo em sistemas baseados em chave-valor, coluna ou grafo.

Referências

[1] Han, Jing, E Haihong, Guan Le e Jian Du: *Survey on nosql database*. Em *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, páginas 363–366. IEEE, 2011. 1, 4, 5, 7, 8, 9

[2] Erturk, Emre e Kamal Jyoti: *Perspectives on a big data application: What database engineers and it students need to know*. Engineering, Technology & Applied Science Research, 5(5):850–853, 2015. 1, 5, 9, 10

[3] Abdrabo, Mai, Mohammed Elmogy, Ghada Eltaweel e Sherif Barakat: *Enhancing big data value using knowledge discovery techniques*. IJ Information Technology and Computer Science, 8:1–12, 2016. 1, 4, 5, 17

[4] Moniruzzaman, ABM e Syed Akhter Hossain: *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*. arXiv preprint arXiv:1307.0191, 2013. 1, 4, 6

[5] Sempéré, G., F. Philippe, A.Dereeper, M. Ruiz e G. Sarah: *Gigwa - genotype investigator for genomewide analyses*. 2016. 1, 6, 17, 18

[6] Nance, Cory, Travis Losser, Reenu Iype e Gary Harmon: *Nosql vs rdbms-why there is room for both*. 2013. 1, 7, 8, 11

[7] Grolinger, K., W. A. Higashino, A. Tiwari e M. A. Capretz: *Data management in cloud environments: Nosql and newsql data stores*. 2013. 1, 8

[8] Gessert, F. e N. Ritter: *Scalable data management: Nosql datastores in research and practice*. 2016. 1, 8, 9

[9] Reis, Debora G, Fabio S Gasparoni, Maristela Holanda, Marcio Victorino, Marcelo Ladeira e Edward O Ribeiro: *An evaluation of data model for nosql document-based databases*. Em *World Conference on Information Systems and Technologies*, páginas 616–625. Springer, 2018. 1, 6, 8, 9, 10, 11, 12, 19, 20

[10] Li, Tao, Ling Liu, Xiaolong Zhang, Kai Xu e Chao Yang: *Provenancelens: Service provenance management in cloud*. 2014. 1, 13, 18

[11] Cheah, You Wei, Richard Canon, Beth Plale e Lavanya Ramakrishnan: *Milieu: Lightweight and configurable big data provenance for science*. 2013. 1, 13, 18

[12] Mattoso, Marta, Cláudia Werner, Guilherme Horta Travassos, Vanessa Braganholo e Leonardo Murta: *Gerenciando experimentos científicos em larga escala.* 2008. 1, 13, 18

[13] Bellazzi, Riccardo: *Big data and biomedical informatics: a challenging opportunity.* Yearbook of medical informatics, 9(1):8, 2014. 2, 5, 6, 17

[14] Brevern, Alexandre G de, Jean Philippe Meyniel, Cécile Fairhead, Cécile Neuvéglise e Alain Malpertuy: *Trends in it innovation to build a next generation bioinformatics solution to manage and analyse biological big data produced by ngs technologies.* BioMed research international, 2015, 2015. 2, 17

[15] Stephens, Zachary D, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha e Gene E Robinson: *Big data: astronomical or genomical?* PLoS biology, 13(7):e1002195, 2015. 4

[16] Gao, Song, Linna Li, Wenwen Li, Krzysztof Janowicz e Yue Zhang: *Constructing gazetteers from volunteered big geo-data based on hadoop.* Computers, Environment and Urban Systems, 61:172–186, 2017. 5

[17] Goli-Malekabadi, Z., M. Sargolzaei-Javan e M. Akbari: *An effective model for store and retrieve big health data in cloud computing.* páginas 72–82, 2016. 5, 17

[18] Strauch, Christof, Ultra Large Scale Sites e Walter Kriha: *Nosql databases.* Lecture Notes, Stuttgart Media University, 20, 2011. 5, 6

[19] Hecht, Robin e Stefan Jablonski: *Nosql evaluation: A use case oriented survey.* Em *Cloud and Service Computing (CSC), 2011 International Conference on*, páginas 336–341. IEEE, 2011. 6, 8, 9

[20] Corbellini, A., C. Mateos, A. Zunino e D. Godoy: *Information systems.* 2017. 6

[21] Padhy, Rabi Prasad, Manas Ranjan Patra e Suresh Chandra Satapathy: *Rdbms to nosql: reviewing some next-generation non-relational database's.* International Journal of Advanced Engineering Science and Technologies, 11(1):15–30, 2011. 6

[22] Toth, Renato Molina: *Abordagem nosql-uma real alternativa.* Sorocaba, São Paulo, Brasil: Abril, 13, 2011. 6, 8

[23] Carniel, Anderson Chaves, Aried de Aguiar Sá, Marcela Xavier Ribeiro, Renato Bueno, Cristina Dutra de Aguiar Ciferri e Ricardo Rodrigues Ciferri: *Análise experimental de bases de dados relacionais e nosql no processamento de consultas sobre data warehouse.* Em *SBBD (Short Papers)*, páginas 113–120, 2012. 7

[24] Brewer, Eric e Armando Fox: *Harvest, yield, and scalable tolerant systems.* 1999. 7

[25] *Mongodb architeture guide*, 2016. `https://www.mongodb.com/collateral/mongodb-architecture-guide?jmp=hero`. 9, 10, 12, 27

[26] *Db-engine ranking of document stores.* https://db-engines.com/en/ranking_trend/document+store, Accessed: 2018-06-30. 10

[27] *Databases and collections.* https://docs.mongodb.com/manual/core/databases-and-collections/, Accessed: 2018-06-23. 10

[28] *Gridfs.* https://docs.mongodb.com/manual/core/gridfs/, Accessed: 2018-06-23. 12, 27, 28

[29] *Glossary.* https://docs.mongodb.com/manual/reference/glossary/, Accessed: 2018-06-30. 12

[30] *Mongodb c driver.* http://mongoc.org/, Accessed: 2018-06-19. 12

[31] *mongo-c-driver.* https://github.com/mongodb/mongo-c-driver, Accessed: 2018-06-19. 12

[32] Yang, Chaowei, Qunying Huang, Zhenlong Li, Kai Liu e Fei Hu: *Big data and cloud computing: innovation opportunities and challenges.* International Journal of Digital Earth, 10(1):13–53, 2017. 12, 13

[33] Mohamed, Mohamed, Mourad Amziani, Djamel Belaïd, Samir Tata e Tarek Melliti: *An autonomic approach to manage elasticity of business processes in the cloud.* Future Generation Computer Systems, 50:49–61, 2015. 12

[34] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg e Ivona Brandic: *Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility.* Future Generation computer systems, 25(6):599–616, 2009. 13

[35] Foster, Ian, Yong Zhao, Ioan Raicu e Shiyong Lu: *Cloud computing and grid computing 360-degree compared.* Em *Grid Computing Environments Workshop, 2008. GCE'08*, páginas 1–10. Ieee, 2008. 13

[36] Buyya, Rajkumar, Rajiv Ranjan e Rodrigo N Calheiros: *Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services.* Em *International Conference on Algorithms and Architectures for Parallel Processing*, páginas 13–31. Springer, 2010. 13

[37] *Visão geral google cloud plataform.* https://cloud.google.com/docs/overview/, Accessed: 2018-08-07. 13

[38] Buneman, Peter, Sanjeev Khanna e Tan Wang-Chiew: *Why and where: A characterization of data provenance.* Em *International conference on database theory*, páginas 316–330. Springer, 2001. 13

[39] Paula, Renato de, Maristela Holanda, Luciana SA Gomes, Sergio Lifschitz e Maria Emilia MT Walter: *Provenance in bioinformatics workflows.* BMC bioinformatics, 14(11):S6, 2013. 14, 17

[40] Wong, Peter P Chen Leah Y: *Active conceptual modeling of learning.* 2007. 14

[41] Hartig, Olaf e Jun Zhao: *Publishing and consuming provenance metadata on the web of linked data*. Em *International Provenance and Annotation Workshop*, páginas 78–90. Springer, 2010. 14

[42] *Prov-dm: The prov data model*. `https://www.w3.org/TR/prov-dm/`, Accessed: 2018-06-23. 14

[43] Bivar, Bárbara, Lucas Santos, Troy C Kohwalter, Anderson Marinho, Marta Mattoso e Vanessa Braganholo: *Uma comparação entre os modelos de proveniência opm e prov*. Proocedings of BRESCi, 2013. 14

[44] Belhajjame, Khalid, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker *et al.*: *Prov-dm: The prov data model*. W3C Recommendation, 2013. 14, 15, 16

[45] Luscombe, Nicholas M, Dov Greenbaum e Mark Gerstein: *What is bioinformatics? a proposed definition and overview of the field*. Methods of information in medicine, 40(04):346–358, 2001. 17

[46] Lewis, C., S. McQuais, P. W. Hamilton, M. Salto-Tellez, D. McArt e J. A. James: *Building a 'repository of science': The importance of integrating biobanks within molecular pathology programmes*. 2016. 17

[47] Aniceto, R., R. Xavier, M. Holanda, M. E. Walter e S. Lifschitz: *Genomic data persistency on a nosql database*. 2014. 17

[48] Mattoso, Marta, Jonas Dias, Flavio Costa, Daniel de Oliveira e Eduardo Ogasawara: *Experiences in using provenance to optimize the parallel execution of scientific workflows steered by users*. Em *Workshop of Provenance Analytics*, volume 1, 2014. 17

[49] Kanwal, Sehrish, Farah Zaib Khan, Andrew Lonie e Richard O Sinnott: *Investigating reproducibility and tracking provenance–a genomic workflow case study*. BMC bioinformatics, 18(1):337, 2017. 17, 18

[50] Leipzig, Jeremy: *A review of bioinformatic pipeline frameworks*. Briefings in bioinformatics, 18(3):530–536, 2017. 17

[51] Chacko, Anu Mary, Ajeeb M Basheer e Dr. S D Madhu Kumar: *Capturing provenance for big data analytics done using sql interface*. 2015. 18

[52] Costa, Flavio, Vítor Silva, Daniel De Oliveira, Kary Ocaña, Eduardo Ogasawara, Jonas Dias e Marta Mattoso: *Capturing and querying workflow runtime provenance with PROV: a practical approach*. Em *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, páginas 282–289. ACM, 2013. 18

[53] Hondo, Fernanda, Polyane Wercelens, Waldeyr da Silva, Iasmini Lima, Ingrid Santana, Gabriel de Araujo, Aleteia Araujo, Maria Emília Walter, Maristela Holanda e Sergio Lifschitz: *Uso de bancos de dados nosql para gerenciamento de dados em workflow de bioinformática*. Em *PROCEEDINGS OF THE SATELLITE EVENTS OF THE 32nd BRAZILIAN SYMPOSIUM ON DATABASES*, páginas 310–317, 2017. 18

[54] Hondo, Fernanda, Polyane Wercelens, Waldeyr da Silva, Klayton Castro, Ingrid Santana, Maria Emilia Walter, Aleteia Araujo, Maristela Holanda e Sergio Lifschitz: *Data provenance management for bioinformatics workflows using nosql database systems in a cloud computing environment.* Em *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, páginas 1929–1934. IEEE, 2017. 18, 20, 21, 22, 33, 34

[55] Vukicevic, M., S. Radovanovic, M. Milovanovic e M.. Minovic: *Cloud based metalearning system for predictive modeling of biomedical data.* 2014. 18

[56] Guimaraes, Valeria, Fernanda Hondo, Rodrigo Almeida, Harley Vera, Maristela Holanda, Aleteia Araujo, Maria Emilia Walter e Sergio Lifschitz: *A study of genomic data provenance in nosql document-oriented database systems.* Em *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, páginas 1525–1531. IEEE, 2015. 20

[57] *Hisat.* `https://ccb.jhu.edu/software/hisat/index.shtml`, Accessed: 2018-11-16. 30

[58] *Ensembl.* `http://www.ensembl.org/index.html`, Accessed: 2018-11-16. 30

[59] *European nucleotide archive.* `https://www.ebi.ac.uk/ena`, Accessed: 2018-11-16. 30, 32

[60] *samtools.* `http://samtools.sourceforge.net/`, Accessed: 2018-11-16. 30

[61] *Htseq.* `https://htseq.readthedocs.io/en/release_0.10.0/`, Accessed: 2018-11-16. 30

[62] Lopes, Ingrid Santana e Maristela Holanda: *Biological database in nosql document-oriented database.* Em *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–6. IEEE, 2018. 59