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

Gerenciamento de Proveniência de Dados de Workflows de Bioinformática em Ambiente de Nuvens Federadas

Polyane Wercelens de Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Coorientadora
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo

Brasília
2019

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

Gerenciamento de Proveniência de Dados de Workflows de Bioinformática em Ambiente de Nuvens Federadas

Polyane Wercelens de Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

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

Prof.ª Dr.ª Célia Ghedini Ralha
CIC/UnB

Prof. Dr. Sérgio Lifschitz
PUC/Rio

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
Coordenador do Programa de Pós-graduação em Informática

Brasília, 03 de Julho de 2019

Dedicatória

Dedico este trabalho a Deus, essencial durante toda a minha trajetória. Dedico também aos meus amados pais, irmãos e sobrinhos.

Agradecimentos

Após cumprir esta jornada de estudos, gostaria de agradecer primeiramente e acima de tudo, à Deus. Em meus dias mais difíceis Ele foi a minha força, meu lugar seguro e me capacitou para esta caminhada. Este trabalho é um sonho que foi construído com muita dedicação e apoio de pessoas que acreditaram que era possível.

Agradeço em especial ao meu pai, Anísio Andrade da Silva e à minha mãe, Martha Wercelens Pinheiro Andrade, por estarem ao meu lado incondicionalmente durante esta jornada, pelos ensinamentos e suporte, por acreditarem no meu potencial, pelo amor e carinho dedicados em todos os momentos, principalmente nos momentos mais difíceis.

Agradeço à minha irmã e melhor amiga Karen Wercelens, uma das maiores dádivas da minha vida, uma das pessoas que mais me ensinou sobre amor incondicional, amizade e companheirismo, exemplo de força e fé. Aos meus irmãos Dene Wercelens e Marco Antônio, companheiros de vida, pelo cuidado, amizade, amor e carinho. Agradeço também por terem me dado os melhores presentes que já recebi: meus amados sobrinhos, Isabella Wercelens, Emily Wercelens e Emanuel Wercelens. À minha prima Mirelle Pinheiro, minha irmã do coração, por me acolher nos momentos mais tristes e por estar ao meu lado nos momentos mais felizes.

Agradeço também ao Juarez Paulino, pelos conselhos sábios, por acreditar na minha recuperação e superação, pelo suporte incondicional, pela paciência, dedicação, compreensão e carinho. Ao meu amigo Luiz Alves por me incentivar a lutar pelos meus sonhos, por todos os conselhos e risadas.

Agradeço à minha amiga Mariana Dantas, pelo apoio sempre que precisei de uma palavra amiga, por entender a minha ausência em virtude do tempo dedicado aos estudos e por todo incentivo durante o processo. Agradeço também à minha amiga Fernanda Hondo por tudo que vivemos e compartilhamos, pelas risadas, por todo apoio, amizade e parceria. Foram muitos dias pesquisando e trabalhando juntas em nossos artigos e diversas histórias vividas.

Também agradeço aos amigos e colegas que sempre me incentivaram, por cada palavra amiga e pelo suporte nos trabalhos acadêmicos: Rener Garcia, Klayton Castro, Lucas Maurício, Aurélio Costa, Rafael Rabelo e Waldeyr Silva.

Agradeço às minhas orientadoras, professora Dra. Maristela Holanda e professora Dra. Aletéia Araújo, pelo tempo dedicado a me auxiliar, pelas orientações que conduziram os estudos e elaboração desta dissertação. À professora Maristela, pela compreensão e apoio nos momentos difíceis, pela sabedoria e todo o aprendizado que levarei para a vida.

Agradeço também à professora Dra. Célia Ghedini, que durante um período complicado da minha vida pessoal me acolheu, me ouviu e me incentivou a focar nos estudos para completar esta jornada. Ao professor MSc. Evandro Lorens por ser meu incentivador desde a época da graduação.

Agradeço a todos que participaram direta e indiretamente desta trajetória e que contribuíram de alguma forma para que este trabalho pudesse chegar até aqui.

Resumo

Workflows de Bioinformática prioritariamente visam tratar, processar e analisar dados oriundos de sequenciamento de DNA/RNA. A diversidade desses *workflows* é dependente da questão biológica que se pretende responder, e por isso podem ser bastante complexos. O uso de nuvem federada em *workflows* de Bioinformática, ao mesmo tempo que oferece flexibilidade para o usuário, pode aumentar o trabalho de configuração do ambiente quando comparado a um ambiente de nuvem computacional. Independentemente da questão biológica, e considerando o ambiente computacional como parte do experimento *in silico*, a documentação do *workflow* tem particularidades a serem preservadas com vistas à sua reprodutibilidade. Os modelos de proveniência de dados proveem uma estrutura de armazenamento e recuperação dos dados de proveniência, mantendo seus significados. A maneira com a qual os dados de proveniência são armazenados é outra característica, cujos aspectos tecnológicos influenciam o resultado final. Neste contexto, este trabalho propõe uma solução que permita o gerenciamento de dados de proveniência de *workflows* de Bioinformática em um ambiente de nuvem federada, armazenando os dados de proveniência de forma distribuída em esquemas de dados baseados no PROV-DM, utilizando sistemas de banco de dados NoSQL. Nos resultados, foram explorados aspectos relacionados à federação de nuvens, o que proporcionou menos dependência de um único provedor para os serviços hospedados. Em relação às bases de dados, este trabalho traz três opções de tecnologias de banco de dados para armazenar a proveniência de dados usando o modelo de dados PROV-DM, incluindo o esquema de dados específico de cada banco de dados, que pode ser usado de acordo com a preferência do pesquisador ou integrado aos sistemas de gerenciamento de *workflows*. Por fim, a solução proposta demonstrou ser adequada para o gerenciamento dos dados de proveniência para *workflows* de Bioinformática em nuvem federada.

Palavras-chave: Proveniência de Dados, Bioinformática, Banco de Dados, NoSQL, Nuvem Federada

Abstract

Bioinformatics workflows essentially aim to treat, process, and analyze data from DNA or RNA sequencing. The diversity of these workflows is dependent on the biological question to be answered, which therefore can be quite complex. The computational environment is part of the *in silico* experiment, and regardless of biological questions, the workflow's documentation has particularities to be preserved to promote its reproducibility. Data provenance models address this problem providing a storage and query structure of data provenance while maintaining their meanings. Moreover, technological aspects can influence how data provenance is stored. Using federated cloud in Bioinformatics workflows can provide both flexibility for the user and increase the environment configuring work compared to a cloud computing environment. In this context, this work proposes a solution to data provenance management for Bioinformatics workflows using NoSQL database systems in a federated cloud environment, storing data provenance in distributed databases using data schemas based on PROV-DM. The results report aspects related to cloud federation providing less dependence on a single provider for the hosted services. Concerning the databases, this work draws three options of database technologies to store data provenance using the PROV-DM data model. Specific database data schemas are provided and can be used according to the researcher's preference and can be integrated into workflow management systems. Finally, it is proposed a suitable solution for the data provenance management for Bioinformatics workflows in a federated cloud.

Keywords: Data Provenance, Bioinformatics, Database, NoSQL, Federated Cloud

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Bioinformática é uma área interdisciplinar que busca resolver problemas da Biologia Molecular utilizando ferramentas e métodos de Computação, Matemática e Estatística, cujas soluções estão frequentemente associadas à execução de *workflows* (GEORGAKOPOULOS; HORNICK; SHETH, 1995).

Experimentos em Bioinformática, em geral, são realizados por meio de *workflows*, que podem ser vistos como uma cadeia de execuções de *software*, onde ocorre a automação dos processos e tarefas para atingir um objetivo conforme um conjunto de regras previsto. Os programas que compõem um *workflow* combinam parâmetros e dados de entrada para resolver problemas específicos (LEIPZIG, 2017), (MATTOSO *et al.*, 2014).

O desenvolvimento de experimentos envolve a produção de um grande volume de dados e deve ser acompanhado por abordagens que permitam sua reprodutibilidade possibilitando a verificação e validação dos resultados produzidos (OLIVEIRA; OLIVEIRA; MATTOSO, 2017). Em Bioinformática, a reprodutibilidade dos experimentos é um princípio fundamental para o qual a proveniência dos dados contribui significativamente por meio da aquisição e gerência de conhecimento sobre a trajetória dos dados em um determinado *workflow*.

A proveniência é classificada em categorias que se referem à especificação de um *workflow*, aos dados de execução de um *workflow* e qualquer informação necessária para uso futuro definida pelo usuário, proveniência prospectiva, retrospectiva e dados definidos pelo usuário respectivamente (FREIRE *et al.*, 2008).

Todavia, antes de definir quais informações serão gerenciadas, é necessário definir a forma de organizá-las, com intuito de tornar possível sua recuperação e análise posterior. Com essa finalidade, encontra-se na literatura diversos modelos de proveniência tais como OPM (OPM, 2019), PROV-DM (W3C, 2019), entre outros. O PROV-DM já foi aplicado no contexto da Bioinformática em (PAULINO *et al.*, 2010), (PAULA *et al.*, 2013), (HONDO *et al.*, 2017) e (ALMEIDA *et al.*, 2019). Neste trabalho, a escolha de utilizá-lo é baseada

em (HONDO *et al.*, 2017) onde é apresentada uma modelagem para dados de proveniência de execução de experimentos em nuvem.

O PROV-DM possibilita diferentes sistemas importarem e exportarem suas representações da proveniência (W3C, 2019). Além da trajetória dos dados, vários aspectos devem ser considerados para a reprodutibilidade de experimentos e podem ser representados através da proveniência, abrangendo as configurações dos programas e todo o ambiente computacional.

O provisionamento, instalação e configuração de ambientes computacionais para a execução dos *workflows* requer recursos, tempo e profissionais qualificados. Tais restrições técnicas podem ser minimizadas utilizando a computação em nuvem como uma alternativa, considerando que esse paradigma disponibiliza serviços web (GROZEV; BUYYA, 2014). O termo computação em nuvem foi cunhado como um termo genérico para descrever uma categoria de serviços de computação sob demanda. Ele denota um modelo no qual uma infraestrutura de computação é vista como uma nuvem, onde as empresas e usuários acessam de qualquer lugar do mundo sob demanda. O principal princípio por trás desse modelo é oferecer computação, armazenamento e software como serviço (VOORSLUYS; BROBERG; BUYYA, 2011).

Assim, usuários com diferentes perfis conseguem utilizar uma variedade de funcionalidades de infraestrutura, software e hardware sem a necessidade de lidar com detalhes de configuração e desenvolvimento. Uma das vantagens das nuvens computacionais para os experimentos é o acesso a uma grande variedade de recursos que podem ser alocados apenas para aquela demanda, evitando a aquisição e configuração da infraestrutura computacional (PAULINO *et al.*, 2010).

O gerenciamento de proveniência de *workflows* em ambientes distribuídos é uma atividade complexa. Em ambiente de nuvem computacional, aumenta-se consideravelmente o volume de recursos e ferramentas utilizadas em sua execução e, consequentemente, a quantidade de dados a serem gerenciados (MARINHO *et al.*, 2009).

Para armazenar esses dados, existem diversos Sistemas Gerenciadores de Banco de Dados (SGBD) - relacionais e não relacionais. Os sistemas de bancos de dados não relacionais, também chamados *Not Only SQL* (NoSQL), são conhecidos por serem mais flexíveis em relação ao armazenamento de dados estruturados e não estruturados, à facilidade de escalar e pela capacidade de armazenar e recuperar grandes volumes de dados com alto desempenho (CORBELLINI *et al.*, 2017).

Os sistemas de banco de dados NoSQL ganharam popularidade em ambientes de nuvem computacional, devido a demanda de processamento e armazenamento eficiente de grandes volumes de dados. Especialmente em aplicações de grande escala e processamento simultâneo que exigem alto desempenho durante a leitura e gravação dos dados (HAN *et*

al., 2011). Esses sistemas de banco de dados têm sido utilizados também como soluções alternativas para o armazenamento de dados de proveniência.

Com a diversidade de nuvens computacionais existentes, o provisionamento e a portabilidade de máquinas entre nuvens, demanda tempo de configuração do ambiente que será utilizado para a execução dos experimentos, além de tornar mais complexo o gerenciamento de dados de proveniência.

Todavia, essa diversidade de nuvens computacionais é interessante, pois possibilita a execução de experimentos em mais de um provedor de nuvem simultaneamente de forma distribuída, surgindo o conceito de federação de nuvens. A federação de nuvens baseia-se no uso de diferentes provedores que operam simultaneamente de forma distribuída, minimizando a dependência de um único provedor, melhorando a disponibilidade do serviço e podendo reduzir custo (TOOSI; CALHEIROS; BUYYA, 2014), (ASSIS; BITTENCOURT, 2016).

Neste contexto, gerenciamento de proveniência de dados de *workflows* de Bioinformática em um ambiente de nuvem federada é um desafio, assim como também é o armazenamento de dados distribuídos.

1.1 Descrição do Problema

Workflows de Bioinformática podem ser executados inúmeras vezes por diversos pesquisadores, utilizando várias ferramentas e parâmetros, gerando uma variedade de dados como resultado. O armazenamento da proveniência dos dados contribui para a reprodutibilidade dos experimentos e análise dos dados. Todavia, em um ambiente de nuvem federada, considerando a diversidade de provedores existentes e que o mesmo experimento pode ser executado e armazenado de forma distribuída em vários provedores simultaneamente, torna-se mais complexo o gerenciamento desses dados.

Conforme o cenário apresentado, pode-se citar como questão de pesquisa: como fazer a gestão de dados de proveniência de *workflows* de Bioinformática em ambiente de nuvem federada?

Neste sentido, este trabalho assume como hipótese que o gerenciamento de dados de proveniência de *workflows* de Bioinformática em ambiente de nuvem federada, pode ser feito por meio de uma solução que contemple as ferramentas e softwares necessários para execução de *workflows* de Bioinformática, armazenando os dados de forma distribuída em esquemas de dados baseados no modelo PROV-DM e persistidos em sistemas de banco de dados NoSQL.

1.2 Objetivos

O objetivo geral desta dissertação é desenvolver uma solução que permita o gerenciamento de dados de proveniência de *workflows* de Bioinformática, armazenando de forma distribuída em esquemas de dados baseados no PROV-DM, utilizando sistemas de banco de dados NoSQL em um ambiente de nuvem federada.

1.2.1 Objetivos Específicos

Para que o objetivo geral seja atingido, foram definidos os seguintes objetivos específicos:

- Definir os esquemas de dados de proveniência de dados de *workflows* de bioinformática para nuvem federada;
- Definir uma arquitetura em um ambiente de nuvem federada que possibilite a gestão de dados de proveniência de forma distribuída;
- Desenvolver um serviço de captura, armazenamento e consulta dos dados de proveniência;
- Avaliar a adequação do modelo de dados proposto e validar o funcionamento da solução em ambiente de nuvem federada.

1.3 Metodologia

Este trabalho tem natureza empírica com propósito de pesquisa exploratória, abordagem qualitativa e quantitativa. Foram realizadas pesquisas bibliográficas por meio de um processo empírico e exploratório com o objetivo de levantar o estado da arte a respeito dos temas relacionados a este trabalho. O estudo abrangeu uma investigação sobre *workflows* científicos e suas áreas de aplicação, bem como os elementos e informações que os compõem.

A partir dessa investigação, o escopo do trabalho foi delimitado a área de Bioinformática. O modelo de dados de proveniência PROV-DM foi escolhido com base nos trabalhos relacionados que o validaram para o uso na área de Bioinformática (PAULA *et al.*, 2013), (HONDO *et al.*, 2017) e (ALMEIDA *et al.*, 2019). Dessa forma, foi realizado um estudo mais profundo sobre o PROV-DM visando identificar elementos e padrões para serem aplicados aos esquemas de dados de proveniência para ambiente de nuvem federada.

Com relação aos sistemas de banco de dados NoSQL, o estudo identificou as principais famílias que categorizam esses bancos de dados (CORBELLINI *et al.*, 2017). Foi possível

constatar que a família chave-valor não representa as relações entre os dados, associado apenas uma chave ao valor armazenado, ficando a cargo das aplicações gerenciar possíveis relacionamentos entre os dados. Dessa forma, a família chave-valor foi desconsiderada para o escopo deste trabalho. O estado da arte apresentou os principais sistemas de banco de dados NoSQL utilizados nas famílias de colunas, documentos e grafos (TALREJA *et al.*, 2019), (CHIGURUPATI *et al.*, 2019), (CHANDRABABU; BASTOLA, 2019). Com base nisso foram escolhidos, inicialmente, respectivamente o Cassandra, o MongoDB e o Neo4J para estudos e testes preliminares. Durante o estudo sobre a arquitetura dos sistemas de banco de dados NoSQL foi possível constatar que a versão gratuita do Neo4J não possibilita a criação de *cluster* e armazenamento de dados brutos, como por exemplo, os arquivos de entrada e saída de cada fase do *workflow*. Dessa forma, foi necessário escolher outro NoSQL da família de grafos. O OrientDB foi escolhido para testes preliminares por permitir a criação de *cluster* em sua versão gratuita e armazenar dados brutos. Por fim, foi realizado um estudo sobre nuvem federada, com objetivo de identificar especificidades desse tipo de ambiente para subsidiar a proposta de modelagem dos esquemas de dados para cada família de NoSQL.

Para o desenvolvimento e implementação experimental, foi proposta uma arquitetura contendo as ferramentas necessárias para armazenar dados de proveniência em cada banco de dados de forma distribuída. Os esquemas de dados foram construídos para armazenar dados de proveniência prospectiva, retrospectiva e dados definidos pelo usuário em um ambiente de nuvem federada. A proveniência prospectiva refere-se à especificação de uma tarefa computacional, a retrospectiva é referente às etapas executadas e o ambiente utilizado, e os dados de usuário são referentes à informações definidas pelo usuário para uso futuro (FREIRE *et al.*, 2008). Para simplificar o provisionamento e possibilitar a portabilidade dos nós do *cluster* de banco de dados, foi utilizada uma ferramenta baseada em contêineres chamada Docker (Docker Inc., 2019). O uso dessa ferramenta mostrou-se eficaz do ponto de vista do provisionamento e portabilidade da arquitetura proposta entre diferentes provedores de nuvem.

Na sequência foram escolhidos os provedores de nuvem, sendo um provedor de nuvem privada chamado CloudJus do Supremo Tribunal Federal e três provedores de nuvem pública (Google Cloud, Microsoft Azure e Digital Ocean). A escolha foi baseada no custo, considerando que os provedores disponibilizam *vouchers* gratuitos para utilização dos serviços. Para o provisionamento da plataforma para execução dos *workflows* e gerenciamento dos dados de proveniência foram utilizadas imagens construídas na ferramenta Docker e executadas nas máquinas virtuais provisionadas nos provedores de nuvem. Em paralelo foi construído um tutorial disponível no Github (WERCELENS, 2019) com intuito de documentar as etapas necessárias para o provisionamento dessas imagens na nuvem.

Após o provisionamento da plataforma, foram executados três *workflows* de Bioinformática e os dados de proveniência armazenados em provedores de nuvem distintos para cada um dos sistemas de banco de dados NoSQL escolhidos em seus respectivos esquemas de dados. Foi utilizada a metodologia empírica para avaliar os resultados considerando o armazenamento e consulta dos dados, e geração do grafo de proveniência no formato do PROV-DM. Os dados foram armazenados e o grafo de proveniência gerado por meio de consultas. Dessa forma, a proposta foi avaliada e considerada viável para responder a questão de pesquisa.

As principais contribuições desta dissertação são:

- Um esquema de dados conceitual de proveniência de *workflows* da Bioinformática baseado no PROV-DM com objetivo de viabilizar a reprodutibilidade dos experimentos;
- Mapeamento deste esquema conceitual para esquemas físicos específicos para três sistemas banco de dados NoSQL, capazes de armazenar dados de proveniência baseado no PROV-DM a partir de execuções de experimentos em um ambiente de nuvem federada;
- Uma arquitetura capaz de gerenciar os dados de proveniência prospectiva, retrospectiva e dados definidos pelo usuário, e armazenar de forma distribuída em ambiente de nuvem federada.

1.4 Estrutura do Trabalho

Este documento está estruturado nos seguintes capítulos:

- O Capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento deste projeto, como *workflows* científicos, *workflows* de Bioinformática, proveniência de dados, sistemas de banco de dados NoSQL, nuvem federada, e alguns trabalhos relacionados a esta pesquisa;
- O Capítulo 3 especifica a arquitetura e esquemas de dados propostos seguindo o modelo PROV-DM para este trabalho e o ambiente de nuvem federada utilizado;
- O Capítulo 4 apresenta os estudos de casos aplicados para validação da arquitetura e esquemas de dados, e as contribuições;
- O Capítulo 5 apresenta as conclusões deste trabalho e os trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo são tratados os conceitos pertinentes a *workflows* científicos, *workflows* de Bioinformática, proveniência de dados, computação em nuvem, federação de nuvens e bancos de dados NoSQL. Assim, este capítulo foi dividido nas seguintes seções: a Seção 2.1 trata sobre os conceitos de *workflows* científicos, sua utilização na área de Bioinformática e suas principais características; a Seção 2.2 descreve a proveniência de dados, seus modelos e utilização na área de Bioinformática; a Seção 2.3 trata os conceitos de computação em nuvem, suas principais características, arquitetura e modo de funcionamento; a Seção 2.4 aborda o conceito de federação de nuvens e suas principais características; a Seção 2.5 aborda os conceitos de bancos de dados NoSQL; e por fim, a Seção 2.6 mostra um resumo dos trabalhos que são relacionados ao trabalho proposto.

2.1 *Workflows* Científicos

Um *workflow* é a automação de um processo de negócio, no todo ou em parte, onde os documentos, informações e tarefas passam de um participante para outro, com intuito de que uma ação seja tomada segundo um conjunto de regras (BARGA; GANNON, 2007).

Experimentos científicos computacionais usam técnicas de computação integradas a metodologias e programas científicos para apoiar o desenvolvimento da ciência. Experimentos em diferentes domínios do conhecimento são frequentemente dependentes de métodos computacionais orientados a dados (OLIVEIRA; OLIVEIRA; MATTOSO, 2017). Eles podem ser modelados como *workflows* científicos e normalmente são realizados por meio de combinações de programas, cada um com um conjunto de parâmetros e dados de entrada, para resolver um problema específico (MATTOSO *et al.*, 2014).

Um *workflow* científico é composto por um conjunto de atividades que envolvem a execução coordenada de tarefas múltiplas realizadas por diferentes entidades de processamento permitindo modelar, gerenciar e coordenar a execução de experimentos científicos

que envolvem diversas fases, cada uma com características, propósitos e ordem de execução particulares (GEORGAKOPOULOS; HORNICK; SHETH, 1995), (MATTOSO *et al.*, 2008), (ROSA *et al.*, 2016).

As atividades de cada fase de um *workflow* científico são normalmente executadas por um ou mais programas que recebem dados de entrada e produzem um conjunto de dados de saída. O conjunto de dados resultante de cada fase pode ser utilizado como entrada para a próxima fase e a execução das fases pode ser sequencial ou paralela, respeitando a ordem de execução e as dependências entre as tarefas.

A Figura 2.1 mostra um exemplo de um *workflow* composto por n fases, onde as fases são executadas sequencialmente. A primeira fase utiliza um conjunto de dados de entrada, gerando um conjunto de dados de saída que, por sua vez, é utilizado pela próxima fase e assim por diante.

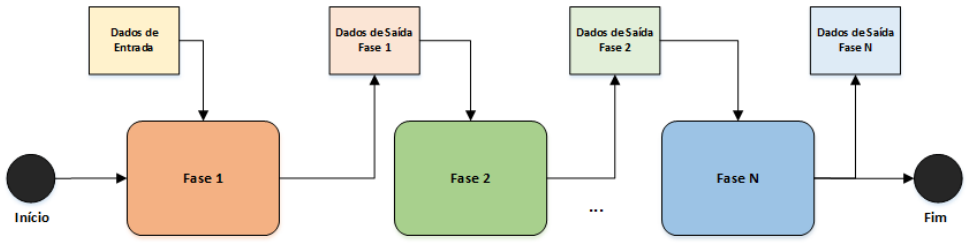


Figura 2.1: Exemplo de um *workflow* sequencial.

2.1.1 *Workflows* de Bioinformática

Diversas áreas da ciência utilizam *workflows* para modelar seus experimentos científicos, dentre elas, a Bioinformática. Os *workflows* de Bioinformática, de modo geral, processam dados biológicos oriundos de processos de sequenciamento de ácido desoxirribonucleico (DNA) ou ácido ribonucleico (RNA).

Os dados produzidos por sequenciadores de alto desempenho *High Throughput Sequencing* (HTS) necessitam de análises como controle de qualidade e identificação de funções biológicas. O processo de análise pode ocorrer em fases diferentes e utilizando ferramentas diferentes que compõem os *workflows* (SALDANHA, 2012).

Um *workflow* de Bioinformática tipicamente possui as fases de filtragem, montagem ou mapeamento e análise (HAAS *et al.*, 2013), conforme mostra a Figura 2.2. Incialmente, o material biológico (DNA ou RNA) coletado e replicado em processos laboratoriais é sequenciado gerando pequenas sequencias chamadas *reads*. As *reads*, geralmente possuem um indicador de qualidade, que pode ser utilizado para filtrá-las (PAULA *et al.*, 2013).

Para filtrar as *reads*, os arquivos gerados pelos sequenciadores são utilizados como entrada na fase de filtragem do *workflow*. Nessa fase são aplicados filtros por meio de

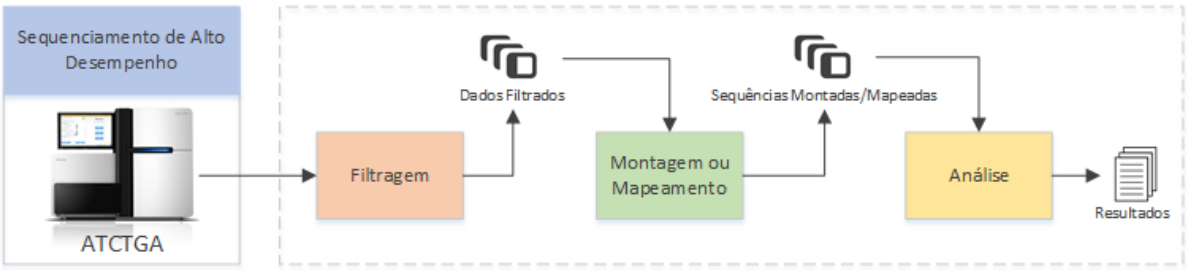


Figura 2.2: Exemplo de *workflow* de projetos para genoma e transcritoma.

parâmetros como, a qualidade das *reads*, regiões de mapeamento que podem ser aproveitadas para as fases seguintes do *workflow*, dentre outros. Esses parâmetros podem variar de acordo com os objetivos do projeto, a espécie estudada e experiência do pesquisador (HUACARPUMA, 2012).

Os dados resultantes da fase de filtragem são utilizados como entrada na fase de montagem. Na montagem as *reads* são agrupadas com o objetivo de montar a sequência original. Existem dois tipos de montagem (EKBLOM; WOLF, 2014), *ab initio* e com referência. Na montagem com referência, é feito mapeamento das *reads* para um genoma de referência que normalmente é um organismo filogeneticamente próximo. Na montagem *ab initio* não é usado um genoma de referência e as *reads* são alinhadas entre si dando origem a *contigs*.

A última fase, representada na Figura 2.2, é a fase de análise. Nessa fase, as porções de DNA ou RNA mapeadas/montadas são analisadas, com a finalidade de obter informações biológicas para validar ou refutar a hipótese inicial de pesquisa do experimento (CAMACHO *et al.*, 2013).

2.2 Proveniência de Dados

A execução de *workflows* de Bioinformática gera dados de proveniência, que permite a avaliação e reprodução dos experimentos. A reprodutibilidade é uma maneira de permitir que o conhecimento fique disponível para o público em geral, é uma questão fundamental na produção do conhecimento científico. Uma contribuição científica é considerada valiosa se, entre outras coisas, outros pesquisadores são capazes de reproduzir seus resultados com sucesso (OLIVEIRA; OLIVEIRA; MATTOSO, 2017).

A proveniência, ou linhagem, de um experimento está relacionada aos metadados associados aos dados gerados por uma execução específica do experimento. Ela deve fornecer uma linhagem ou histórico de como os dados foram criados, utilizados e modificados, preservando a fonte de dados e o processo empregado para transformá-los em um produto final (BUNEMAN; KHANNA; WANG-CHIEW, 2001), (LI *et al.*, 2014), (OLIVEIRA; OLIVEIRA;

MATTOSO, 2017). Por esse motivo, a proveniência dos dados está intimamente relacionada à reprodutibilidade. Dessa forma, os cientistas têm a possibilidade de estudar detalhes dos seus experimentos e executá-los diversas vezes de maneira mais controlada e planejada (PAULA *et al.*, 2013).

Além da reprodutibilidade, existem outras utilidades para a proveniência de dados. A procedência do dado é a informação que ajuda a determinar o histórico de derivação de dados, a partir de suas fontes originais, permitindo diversas análises de dados por parte dos cientistas, possibilitando aferir a qualidade dos dados gerados, com base nas referências ancestrais determinando sua confiabilidade (MARINHO *et al.*, 2009), (OLIVEIRA; OLIVEIRA; MATTOSO, 2017).

Segundo FREIRE *et al.* (2008), a proveniência é classificada em categorias: prospectiva, retrospectiva e dados definidos pelo usuário. A proveniência prospectiva captura a especificação de uma tarefa computacional (um *script* ou um *workflow*), em outras palavras, a proveniência prospectiva captura os passos que devem ser seguidos para a geração de um dado. A proveniência retrospectiva captura os passos que foram executados para gerar um dado, ou seja, as etapas executadas e as informações sobre o ambiente utilizado. Dados definidos pelo usuário são referentes a qualquer informação necessária para uso futuro, conforme definido pelo usuário. Uma não depende da outra, é possível capturar informações sobre as etapas executadas, como foi executada, quem o executou e quanto tempo levou sem ter conhecimento prévio da sequência de etapas computacionais envolvidas. Os dados prospectivos e retrospectivos de proveniência e definidos pelo usuário podem ser modelados eficientemente usando modelos de proveniência de dados compatíveis (GUEDES *et al.*, 2018).

2.2.1 Modelos de Proveniência de Dados

O principal objetivo dos modelos de proveniência de dados é prover uma estrutura de armazenamento e recuperação dos dados de proveniência, mantendo seus significados e potencializando seus benefícios (ALMEIDA, 2016). As principais características dos modelos são: fornecimento de ferramentas que facilitem a utilização dos mesmos, representação gráfica simplificada, bem desenvolvido, capacidade de propiciar intercâmbio de informações entre diferentes sistemas, entre outras (PAULA, 2013). Existem diversos modelos de proveniência de dados, com diferentes formatos e objetivos, entre os quais pode-se citar:

- *W7* (RAM; LIU, 2007): é um modelo baseado na ontologia de Bunge (BUNGE, 1977), a qual tem como objetivo descrever as propriedades de um objeto. A partir dessa ontologia, o modelo *W7* estruturou a proveniência de dados através de 7 perguntas: O que?, Quem?, Quando?, Onde?, Como?, Qual? e Por quê?;

- *Provenancy Vocabulary* (HARTIG; ZHAO, 2010): voltado para solucionar problemas de proveniência de dados publicados na web. Fornece classes e propriedades para que os dados e metadados publicados possam ser armazenados com informações úteis sobre a proveniência dos mesmos;
- *Provenir Ontology* (SAHOO; SHETH, 2009): modelo de proveniência de dados genérico, o qual prioriza a interoperabilidade entre sistemas e adaptação para quaisquer aplicações;
- *Open Provenance Model* (OPM) (MOREAU *et al.*, 2009): modelo aberto, com foco na caracterização de qualquer "coisa", material ou imaterial. Demonstra a relação entre eventos que afetam objetos e descreve a relação entre eles por meio de um grafo acíclico direcionado;
- *Provenance Data Model* (PROV-DM) (MOREAU; MISSIER, 2013): descreve pessoas, entidades e atividades envolvidas na produção de um determinado dado. Permite que a proveniência seja demonstrada e trocada entre diferentes sistemas.

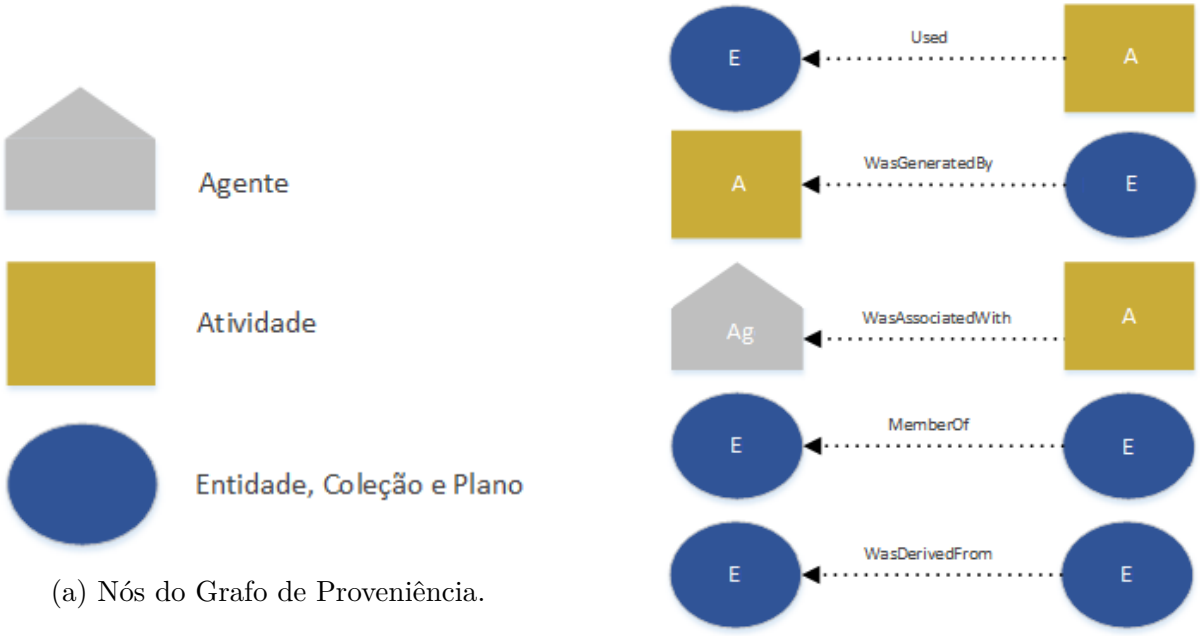
2.2.2 *Provenance Data Model* (PROV-DM)

O PROV-DM é um modelo de dados genérico para proveniência que permite que representações específicas de domínio e de aplicação da proveniência sejam traduzidas em um modelo de dados tornando possível a interoperabilidade de informações entre diferentes sistemas (W3C, 2019).

A principal função do PROV-DM é descrever as pessoas, entidades e atividades que estão envolvidas na produção de um dado ou de um objeto qualquer (W3C, 2019). Dessa forma, o modelo viabiliza a demonstração e troca de proveniência entre diferentes sistemas. Sua principal característica é demonstrar a proveniência de qualquer objeto por meio de um grafo direcionado (PAULA *et al.*, 2013).

O modelo possui um *design* modular, estruturado em oito componentes (W3C, 2019):

- Agentes (*Agents*): são entidades que influenciam, direta ou indiretamente, as execuções das atividades;
- Anotações (*Annotation*): mecanismos para inclusão de anotações para os elementos do modelo;
- Atividades (*Activities*): representam os processos que usam e geram as entidades;
- Coleções (*Collections*): são coleções de entidades que podem ter a sua proveniência demonstrada coletivamente;



(a) Nós do Grafo de Proveniência.

(b) Arestas do Grafo de Proveniência.

Figura 2.3: Representação gráfica dos nós e arestas do grafo de proveniência do modelo PROV-DM (W3C, 2019).

- Conta (*Account*): representa um conjunto de informações que são parte de um grafo de proveniência;
- Derivações (*Derivations*): são transformações de uma entidade em outra, permitindo demonstrar a dependência entre as entidades usadas e geradas;
- Entidades (*Entities*): representam qualquer objeto;
- Plano (*Plan*): representa um conjunto de ações ou etapas pretendidas por um ou mais agentes para atingir alguns objetivos.

Tipos básicos que originam os possíveis nós do grafo de proveniência, são definidos no modelo: Atividade e Entidade. A Entidade possui quatro subtipos: Agente, Coleção, Conta e Plano. A Figura 2.3a ilustra os nós do grafo de proveniência, que são representados por diferentes símbolos. Os subtipos de Entidade – Coleção e Plano – são representados pelo símbolo "Entidade". O subtipo Conta representa o próprio grafo de proveniência, ou seja, não há símbolo específico para representá-lo (W3C, 2019).

A Figura 2.3b representa graficamente os relacionamentos entre os componentes do modelo. As arestas do grafo de proveniência representam os relacionamentos entre os nós (componentes). Existem diversos tipos de relacionamentos, o que permite expressar mais precisamente a proveniência. Entretanto, os mais relevantes para o contexto deste trabalho são:

- *memberOf*: indica que uma Entidade particular é membro de uma Coleção;
- *used*: indica o uso de uma Entidade por uma Atividade;
- *wasAssociatedWith*: atribui responsabilidade a um Agente sobre uma Atividade;
- *wasAttributedTo*: atribui responsabilidade a um Agente sobre uma Entidade;
- *wasDerivedFrom*: indica que uma Entidade original foi utilizada, direta ou indiretamente, para gerar outra Entidade derivada;
- *wasGeneratedBy*: indica que uma Entidade foi gerada por uma Atividade.

2.2.3 Proveniência de Dados de *Workflows* Bioinformática

A Bioinformática é uma área multidisciplinar que utiliza, de forma intensiva, ferramentas computacionais, com intuito de coletar, organizar, armazenar, recuperar e analisar dados biológicos, permitindo inferir ou descobrir informações a respeito da biologia e/ou evolução dos organismos (MATTOSO *et al.*, 2008).

Os projetos da bioinformática processam um grande volume de dados, executam diversos processos com várias opções de programas e parâmetros de configuração, e um fluxo de dados entre diversos arquivos de diferentes formatos. Essas características afetam de forma significativa a análise dos resultados.

Dessa forma, manter a proveniência de dados neste contexto demanda uma solução capaz de armazenar a ligação entre as execuções dos processos, seus resultados e os dados processados por eles. Ademais, em laboratórios onde diversos usuários podem trabalhar em projetos distintos, é importante registrar o que cada usuário executou (ALMEIDA *et al.*, 2019). Os dados, os processos executados e as pessoas envolvidas formam os três pilares do PROV-DM. Além disso, a aderência do modelo PROV-DM ao propósito de armazenar dados de proveniência de *workflows* de Bioinformática, foi confirmada nos trabalhos de PAULA *et al.* (2013), HONDO *et al.* (2017) e ALMEIDA *et al.* (2019).

2.3 Computação em Nuvem

A computação em nuvem emergiu como um modelo de computação em que os serviços disponíveis na web permitem que diferentes tipos de usuários obtenham uma grande variedade de recursos, como software e hardware (OLIVEIRA; OLIVEIRA; MATTOSO, 2017).

As nuvens computacionais podem desempenhar um papel fundamental na reprodutibilidade de experimentos. A computação em nuvem tem sido aplicável a uma ampla gama de problemas em vários domínios, incluindo a Bioinformática. Isso porque os ambientes de nuvem computacional são uma alternativa para processar o grande volume de

dados dos *workflows* de Bioinformática (OLIVEIRA; OLIVEIRA; MATTOSO, 2017). Isso porque uma das vantagens das nuvens computacionais para a execução dos experimentos é o acesso a uma grande variedade de recursos que podem ser alocados de acordo com a demanda, evitando a aquisição e a configuração da infraestrutura computacional por parte do usuário (PAULINO *et al.*, 2010).

Em um modelo tradicional de infraestrutura de Tecnologia da Informação e Comunicação (TIC), existem dificuldades de consolidação arquitetural de um ambiente complexo e em crescimento, associadas ao aumento de demandas por novas funcionalidades. A necessidade de aprimorar o desempenho e a agilidade relativos à disponibilização de ambientes mais aderentes aos atributos de usabilidade, flexibilidade, escalabilidade e robustez, traduzem o aumento das taxas de execução de trabalho fora de um contexto de planejamento (KIM *et al.*, 2016).

De acordo com FOSTER *et al.* (2008), computação em nuvem é um paradigma computacional altamente distribuído que fornece infraestrutura de armazenamento, computação, plataformas abstratas e serviços aos clientes, por meio de virtualização, e são gerenciadas e dinamicamente escaláveis sob demanda, por meio da internet.

Assim sendo, as seguintes premissas estabelecem uma nuvem computacional (MELL; GRANCE *et al.*, 2011), (VAQUERO *et al.*, 2008):

- Facilidade de Acesso: mecanismos que facilitem o acesso a nuvem de forma padronizada;
- Serviço sob Demanda: o cliente consome os recursos computacionais sob demanda, respeitando os limites previamente acordados;
- Agrupamento de Recursos: os recursos de um determinado provedor são agrupados para atender a múltiplos consumidores e facilitar os ajustes de demanda;
- Elasticidade: caso seja necessário aumentar ou liberar recurso, seja ele qual for, conforme a demanda, este aumento deve ocorrer de forma fácil ou automática;
- Medição de Serviço: a utilização dos recursos deve ser monitorada, garantindo o cumprimento do contrato de qualidade de serviço firmado entre o provedor de serviço e o usuário.

2.3.1 Modelos de Hospedagem

Os modelos de hospedagem referem-se ao acesso e disponibilidade do ambiente de computação em nuvem. As nuvens computacionais podem ser categorizadas como pública, privada, comunitária ou híbrida (SOTOMAYOR *et al.*, 2009), (VOORSLUYS; BROBERG; BUYYA, 2011):

- Nuvem Pública: a infraestrutura é mantida por uma organização empresarial, acadêmica, governamental ou uma combinação entre elas. Os usuários, conhecendo a localização do serviço, acessam essa infraestrutura utilizando mecanismos adequados de controle de acesso;
- Nuvem Privada: o acesso e utilização da infraestrutura é exclusivo de uma organização, podendo ser mantida pela própria organização, por terceiros ou uma combinação deles;
- Nuvem Comunitária: a infraestrutura é compartilhada por diversas organizações de maneira colaborativa, podendo ser mantida por uma ou mais destas organizações;
- Nuvem Híbrida: é composta por duas ou mais nuvens de modelos de hospedagem distintos.

2.3.2 Modelos de Serviços

Os modelos de serviços que são oferecidos no paradigma de computação em nuvem podem ser divididos em categorias de acordo com o nível de abstração da capacidade fornecida e o modelo de serviço dos provedores. Na literatura é possível encontrar propostas com mais de três categorias. Entretanto, o mais comum é a divisão dos serviços em Infraestrutura como Serviço, Plataforma como Serviço, e Software como Serviço, descritos a seguir (VOORSLUYS; BROBERG; BUYYA, 2011):

- Infraestrutura como Serviço (IaaS): em IaaS o usuário controla os sistemas operacionais, recursos de armazenamento e aplicativos. Eventualmente, é possível selecionar componentes de rede. Dessa forma, são oferecidos ao consumidor recursos computacionais essenciais para a construção de um ambiente de aplicação sob demanda, como por exemplo, armazenamento, rede, entre outros. Para propiciar a interação com dispositivos computacionais, a IaaS disponibiliza uma interface única para administração da infraestrutura, armazenamento e comunicação. Além disso, a IaaS disponibiliza suporte à adição de novos componentes de maneira simplificada e transparente. Entretanto, a administração ou o controle da infraestrutura da nuvem não fica a cargo do usuário;
- Plataforma como Serviço (PaaS): neste modelo o usuário possui o controle das aplicações implantadas e suas configurações. Todavia, diferente do IaaS, o usuário não administra ou controla a infraestrutura subjacente, tais como sistemas operacionais, recursos de armazenamento e rede;

- Software como Serviço (SaaS): o usuário não administra ou controla a infraestrutura subjacente nem as características individuais da aplicação, exceto configurações muito específicas.

2.4 Federação de Nuvens

Ao longo dos anos a utilização de nuvens de maneira isolada, no contexto de um único provedor, passou a não ser mais suficiente para algumas aplicações. Com o intuito de aumentar a disponibilidade dos serviços, e diminuir o tempo de resposta das aplicações, as empresas começaram a criar *datacenters* ao redor do mundo. Assim, a integração de nuvens tornou-se necessária para que os serviços continuem sendo fornecidos de forma rápida, escalável e eficiente. Dessa forma, surge o conceito de federação de nuvens computacionais, que pode ser definido como um conjunto de provedores de nuvens públicos, privados e comunitários, conectados através da internet (SALDANHA, 2012).

A federação de nuvens é um modelo computacional que, por meio da integração de nuvens, possibilita que os provedores de nuvem estendam e aumentem seus recursos mediante acordos estabelecidos. Assim, os provedores obtêm economia de escala, aumento da capacidade e uso eficiente dos seus recursos (CELESTI *et al.*, 2010).

Na literatura existem alguns termos utilizados para definir ambientes de computação em nuvem. Tais termos estão em evolução e é fácil confundi-los, uma vez que eles quase se sobrepõem. Assim, é importante esclarecer os seguintes conceitos, segundo (TOOSI; CALHEIROS; BUYYA, 2014), (CELESTI *et al.*, 2010), (BARRIL; RUYTER; TAN, 2016), (KOGIAS; XEVGENIS; PATRIKAKIS, 2016):

- Federação de Nuvens: as nuvens voluntariamente estabelecem conexão com outras nuvens com o objetivo de estender seus recursos automaticamente sem a necessidade de mediadores. Assim, não há um terceiro ator responsável pelo gerenciamento de recursos dessas diferentes nuvens. Nesse paradigma os provedores visam superar a limitação de recursos em sua infraestrutura local, o que pode resultar na rejeição de solicitações de clientes, terceirizando as solicitações para outros membros da federação. Além disso, a federação de nuvem permite que os provedores que operam em baixa utilização concedam parte de seus recursos a outros membros da federação, a fim de evitar o desperdício de recursos computacionais não utilizados;
- *Multi-cloud*: diversas nuvens são utilizadas para fornecer recursos. Todavia, as nuvens utilizadas neste contexto não participam de forma voluntária no estabelecimento de conexão com outras nuvens. Um terceiro ator é responsável pelo gerenciamento de recursos das diferentes nuvens. Esse gerenciamento pode ser por meio de

um *cloud-broker*, que é um agente intermediário entre o usuário final e as nuvens, ou por meio do próprio usuário alocando recursos diretamente nas nuvens;

- *Intercloud*: os recursos de determinada nuvem podem ser estendidos utilizando qualquer tipo de relação existente, incluindo as duas anteriores.

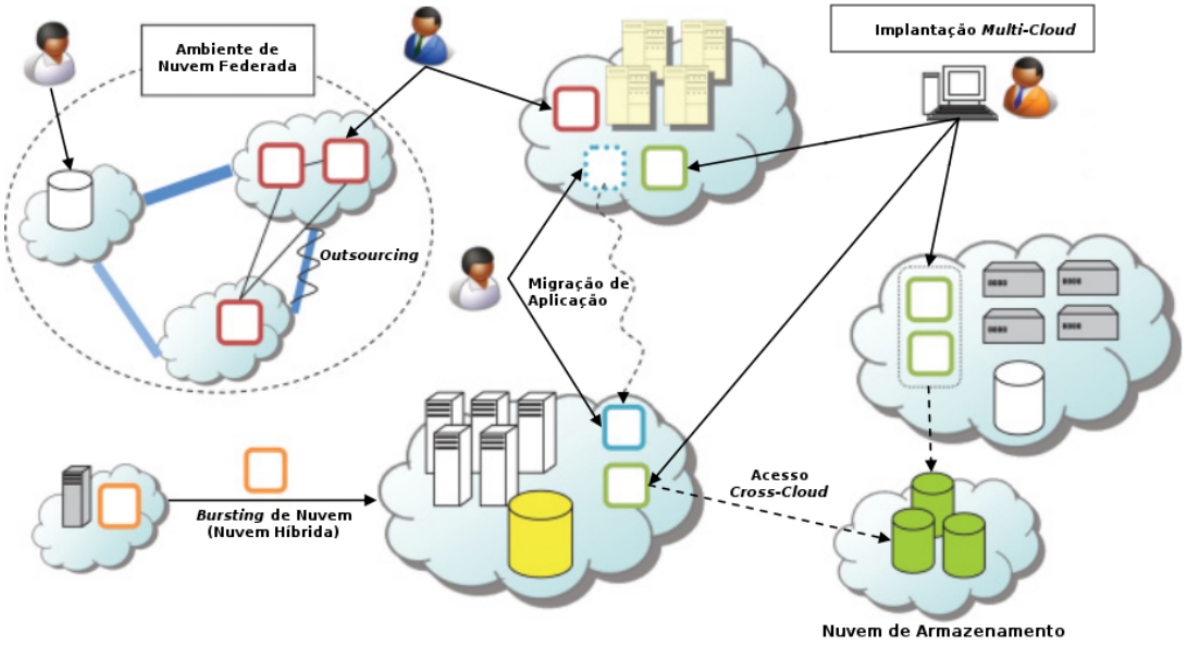


Figura 2.4: Diferentes Interações entre Nuvens (TOOSI; CALHEIROS; BUYYA, 2014).

.

A Figura 2.4 apresenta as diversas interações existentes dentro do universo da computação em nuvem, no qual é possível observar que em um ambiente de *multi-cloud* há um terceiro ator responsável pelo gerenciamento dos recursos das nuvens distintas, e em um ambiente de nuvem federada não há esse ator gerenciando recursos.

Em federação de nuvem, é preciso que exista a cooperação entre as diferentes nuvens, pelo menos duas, sendo possível observar os seguintes comportamentos entre elas (KURZE *et al.*, 2011):

- *Outsourcing*: ao perceber que seus recursos estão esgotando, o provedor de nuvem procura recursos em outros provedores que possuem recursos disponíveis. O uso dessa estratégia é interessante para que o provedor não faça a negação de serviço ao usuário por falta de recursos, evitando a perda de consumidores;

- *Insourcing*: o provedor que possui recursos disponíveis e não estão sendo muito demandados, fornece seus recursos a outros provedores externos, diminuído o alto custo com a sua infraestrutura.

A evolução do paradigma de computação em nuvem foi dividida em três fases por BITTMAN (2008). A primeira fase é chamada de monolítica e é caracterizada por ilhas proprietárias com serviços fornecidos por empresas de grande porte. Na segunda fase, chamada de cadeia vertical de fornecimento, apesar do foco ainda estar nos ambientes proprietários, tem início uma integração, na qual as empresas começam a utilizar os serviços de outras nuvens. Atualmente, é vivida a terceira fase, que é a federação horizontal, na qual pequenos provedores se juntam para aumentar sua escalabilidade e eficiência na utilização dos recursos, e começam a ser discutidos padrões de interoperabilidade.

Uma federação de nuvens possibilita a migração cruzada de recursos para implementação de redundância e complementação de funcionalidades de forma cooperativa (ASSIS; BITTENCOURT, 2016). A cooperação é atrativa por quebrar a dependência de um único provedor, aprimorando a disponibilidade do serviço, reduzindo custos e aumentando o número de combinações de instâncias.

Contudo, as diferenças entre as nuvens, não é uma tarefa simples realizar uma federação. Existem particularidades de um provedor para outro tanto em relação ao software como em relação do hardware (MOURA, 2017).

De acordo com (CELESTI *et al.*, 2010), para existir uma federação os requisitos de automatismo e escalabilidade, bem como segurança interoperável devem ser atendidos:

- Automatismo e Escalabilidade: a escolha de nuvens que atendam às necessidades do consumidor de serviço de nuvem, deve ser feita por meio de mecanismos de descoberta entre as nuvens existentes;
- Segurança Interoperável: as diversas tecnologias de segurança devem ser integradas, de forma que não seja necessário alterar as políticas de segurança da nuvem que se juntar à federação.

A diversidade e a flexibilidade das funcionalidades (crescimento e encolhimento dinâmico dos sistemas de computação) previstas pelo modelo de computação em nuvem federada, combinadas com as magnitudes e incertezas de seus componentes (servidores de computação, serviços e carga de trabalho), apresentam problemas difíceis no provisionamento e na entrega eficazes dos serviços de aplicações. Em particular, encontrar soluções eficientes para acompanhar os desafios é essencial para explorar o potencial das infraestruturas de nuvem federada (BUYYA; RANJAN; CALHEIROS, 2010):

- Previsão de Comportamento da Aplicação: é essencial que o sistema seja capaz de prever as demandas e o comportamento das aplicações, de forma que ele possa tomar decisões inteligentes relacionadas à alocação de recursos, dimensionamento dinâmico, armazenamento ou largura de banda. Para tentar realizar essa previsão,

um modelo deve ser construído levando em consideração estatísticas de padrões de utilização dos serviço, e ajustar as variáveis sempre que for necessário, para que o modelo seja o mais próximo possível da realidade;

- Mapeamento Flexível de Recursos e Serviços: mapear os recursos necessários para os serviços é uma tarefa complexa. Entretanto, o custo é algo que sempre é considerado em qualquer projeto. Dessa forma, é muito importante que o sistema seja o mais eficiente possível, sempre tentando encontrar a melhor configuração de hardware e de software que atenda às demandas de qualidade de serviço que foram estabelecidas entre o usuário e o provedor;

- Modelo Econômico Impulsionado por Técnicas de Otimização: o problema da tomada de decisão orientado pelo mercado é um problema de otimização combinatória, que busca encontrar a melhor combinação entre serviços e planos. Os modelos de otimização visam melhorar tanto os recursos centralizados (utilização, disponibilidade, confiabilidade e incentivo), quanto os centrados nos usuários (tempo de resposta, gasto orçamentário e equidade);

- Integração e Interoperabilidade: diversas empresas possuem dados confidenciais e não se sentirão confortáveis de colocá-los na nuvem por receio de problemas relacionados a privacidade e a segurança. Esse receio é pela questão da segurança, ou seja, medo de que alguma pessoa não autorizada acesse dados confidenciais, e também pela forma como as aplicações que já existem na empresa irão interagir com os dados e as aplicações que estão na nuvem;

- Monitoramento Escalável dos Componentes do Sistema: atualmente as técnicas que são utilizadas para o monitoramento e o gerenciamento dos componentes da nuvem utilizam uma abordagem centralizada. Em sistemas distribuídos manter uma singularidade sempre é um problema, principalmente, por este poder se tornar um gargalo para o sistema, caso o volume de requisições seja muito alto, como também por questões de segurança, visto que algum problema pode prejudicar toda a federação de nuvens. É importante que este monitoramento seja feito de forma distribuída, para que não haja problema de escalabilidade, de desempenho e de confiabilidade.

Um novo paradigma para federação de nuvens é proposto por BIRAN *et al.* (2017), no qual as duas partes envolvidas, os provedores de serviços e os provedores de serviços de nuvem, são beneficiadas tanto do ponto de vista da otimização dos recursos utilizados, quanto do ponto de vista de redução do consumo de energia. Assim, surge o conceito

de sistemas-de-sistemas (SoS), onde um conjunto de sistemas pode disponibilizar outros sistemas e implantá-los em diferentes provedores de nuvem criando uma federação.

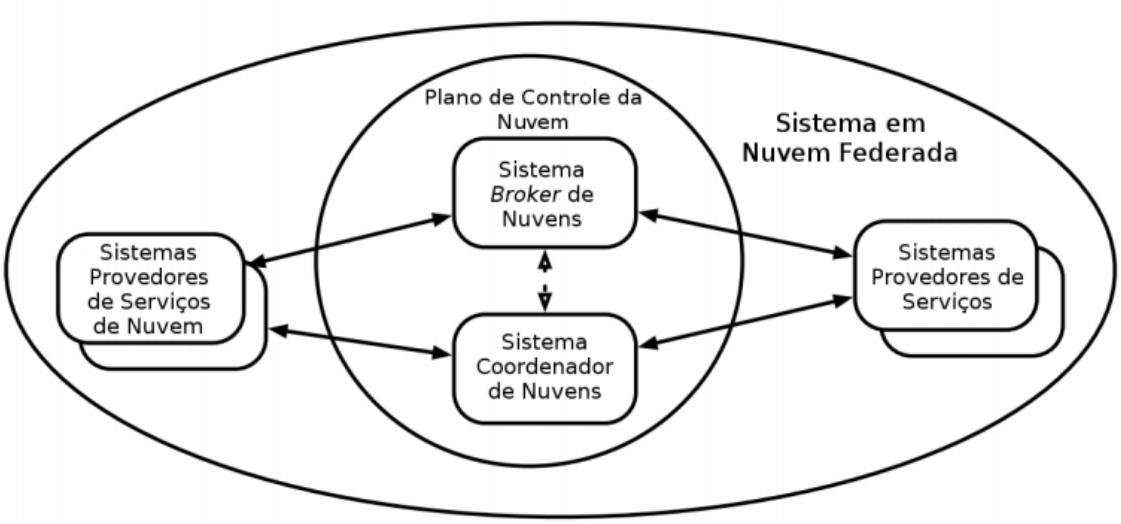


Figura 2.5: Sistemas-de-Sistemas (BIRAN *et al.*, 2017).

.

A Figura 2.5 mostra o plano de controle que estabelece meios para criar uma federação, no qual os provedores de serviços obtêm recursos dos provedores de serviços de nuvens. O coordenador de nuvens é responsável por registrar os tipos de recursos e o preço das nuvens disponíveis. Para que os recursos sejam alocados, o *broker* pesquisa os preços e os recursos solicitados pelos provedores de serviços.

Diante do exporto, é importante destacar que esta dissertação utiliza a definição de SoS proposta por (BIRAN *et al.*, 2017), possibilitando o gerenciamento de dados de proveniência de *workflows* de Bioinformática em um ambiente provisionado em diferentes provedores de nuvem, por meio de um sistema gerenciador de *workflows*.

2.5 Banco de Dados NoSQL

Os bancos de dados NoSQL (*Not only SQL*) surgiram como uma alternativa aos Sistemas Gerenciadores de Banco de Dados Relacionais (SGBDR). Em 1998, Carlo Strozzi cunhou o termo NoSQL para nomear o banco de dados baseado em arquivo desenvolvido por ele. O nome surgiu pelo fato de que o banco de dados utilizava *scripts shell* ao invés de *Structured Query Language* (SQL), linguagem utilizada pelos bancos de dados relacionais como linguagem de consulta. Entretanto, atualmente o termo não refere-se mais ao banco de dados desenvolvido por Carlos Strozzi (MOHAMED; ALTRAFI; ISMAIL, 2014), passando a ser utilizado para referenciar os bancos de dados não relacionais.

O esquema de dados de um NoSQL geralmente é flexível e fluido. Os bancos de dados NoSQL manipulam dados estruturados, semiestruturados e não estruturados e são propostos como soluções escaláveis, proporcionando alta disponibilidade e processamento distribuído. CORBELLINI *et al.* (2017), define alguns fatores para orientar a adoção de um sistema de banco de dados NoSQL para armazenamento de dados:

- Capacidade de executar rotinas em grandes volumes de dados para extrair conhecimento;
- Escalabilidade;
- Esquema flexível;
- Facilidade de implantar o banco de dados em um *cluster*;
- Afinidade de Dados: em geral, como os bancos de dados NoSQL são projetados para serem distribuídos, a localização dos dados no *cluster* é aproveitada para melhorar o uso da rede, normalmente armazenando dados remotos em *cache* e fazendo consultas a esses nós mais próximos.

Os sistemas de banco de dados NoSQL são capazes de lidar com grandes volumes de dados e são muito utilizados devido a sua flexibilidade e facilidade de escalar. Na área de Bioinformática o uso de sistemas de banco de dados NoSQL é um campo emergente e parece ser útil para lidar com os problemas e limitações que os bancos de dados relacionais encontram no campo (BREVERN *et al.*, 2015), (SHAO; CONRAD, 2015), (CHIGURUPATI *et al.*, 2019).

2.5.1 Famílias de Bancos de Dados NoSQL

Os sistemas de banco de dados NoSQL são categorizados em quatro tipos de famílias principais (CORBELLINI *et al.*, 2017), (GESSERT *et al.*, 2017), (HAN *et al.*, 2011), (HECHT; JABLONSKI, 2011):

- Chave-valor: armazenam dados em duas partes, onde cada valor está associado a uma chave única. Os valores são isolados e independentes uns dos outros, enquanto os relacionamentos são tratados pela lógica da aplicação. Alguns exemplos de NoSQL da família chave-valor são Voldemort (Project Voldemort, 2019) e Redis (Redis, 2019);
- Família colunas: armazenam dados por coluna, diferente de um banco de dados relacional, que armazena dados por linha. Assim, algumas linhas podem não conter parte das colunas, oferecendo flexibilidade na definição de dados e permitindo aplicar

algoritmos de compactação de dados por coluna. Além disso, as colunas que não são frequentemente consultadas juntas podem ser distribuídas entre nós diferentes. O HBase (HBase, 2019) e o Cassandra (Cassandra, 2019) são alguns exemplos de sistemas de bancos de dados orientados por colunas;

- Orientados a documentos: armazenam documentos como coleções de atributos e valores e podem conter atributos de vários valores. Eles também usam a noção de chaves e valores, onde cada documento é identificado por uma chave que é exclusiva dentro de uma coleção. Os dados geralmente são armazenados em formatos como *JavaScript Object Notation* (JSON) ou *eXtensible Markup Language* (XML). Alguns exemplos de sistemas de banco de dados orientados a documento são MongoDB (MongoBD, 2019) e CouchDB (CouchDB, 2019);
- Grafos: o esquema é representado por grafos compostos por nós e arestas. Os dados podem ser armazenados nos nós e nas arestas entre eles. Um exemplo de sistemas de banco de dados baseados em grafos é Neo4J (Neo4J, 2019);

Alguns sistemas de bancos de dados NoSQL são híbridos e implementam mais de uma família de colunas, como é o caso do OrientDB (OrientDB, 2019), que implementa a família de grafo, documento e chave-valor.

2.5.2 Provisionamento de Bancos de Dados NoSQL

Recentemente, o uso de contêineres Docker têm sido adotado de forma crescente, com objetivo de simplificar a implantação de software. A configuração e manutenção dos bancos de dados de um software distribuído é uma tarefa desafiadora. Existem diversas configurações e parâmetros com interdependências (TRUYEN *et al.*, 2018).

Uma abordagem baseada no Docker funciona de maneira similar a uma imagem de máquina virtual ao abordar o problema de dependência, fornecendo a outros pesquisadores uma imagem binária na qual todo o software já foi instalado, configurado e testado. A imagem também pode incluir todos os arquivos de dados necessários para a pesquisa, o que pode simplificar a distribuição de dados (BOETTIGER, 2015).

Os aplicativos podem ser empacotados em uma imagem do Docker, independente e executável, incluindo as ferramentas necessárias para o sistema, bibliotecas, configurações e código. Imagens se tornam contêineres em tempo de execução no mecanismo Docker. Ele está disponível para aplicativos baseados em Linux e Windows que suportam software empacotado para serem executados da mesma maneira, independentemente da infraestrutura (Docker Inc., 2019).

A abordagem do Docker tem potencial para oferecer uma solução de configuração e consistência de parâmetros simples para os bancos de dados NoSQL por meio da utili-

zação das imagens. O uso de contêineres permite que o software e os bancos de dados de Bioinformática sejam executados em diversas plataformas de computação, reduzindo significativamente o tempo e o esforço necessários para configurar o ambiente. Portanto, o software não necessita de configurações prévias ao ser implantado em cada laboratório (KIM *et al.*, 2017).

2.6 Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados ao projeto proposto que abordam de maneira semelhante a proveniência de dados. PAULA *et al.* (2013) propôs o uso do PROV-DM para gerenciar os dados de proveniência de projetos de Bioinformática. É feita uma análise comparativa entre diversas abordagens de modelagem de proveniência com objetivo de validar se a proposta é vantajosa. A validação foi feita por meio de um simulador de proveniência, onde o cientista é capaz de criar seus experimentos, mapear e armazenar os dados de proveniência. Seus resultados mostraram que o modelo PROV-DM permite o armazenamento de propriedades para cada execução de um *workflow* de Bioinformática. Isso inclui a representação gráfica dos grandes volumes de dados gerados pelos projetos do genoma usando as coleções de entidades.

O trabalho proposto por FERREIRA; FILIPE JR.; OLIVEIRA (2014) apresenta uma análise de desempenho entre um SGBD relacional e um sistema de banco de dados NoSQL da família de colunas, PostgreSQL e Cassandra, respectivamente em um ambiente de nuvem computacional. No que tange a gerência de dados de proveniência o modelo de proveniência utilizado foi o PROV-Wf. Os resultados mostram que o Cassandra obteve um desempenho mais satisfatório quando comparado ao PostgreSQL.

LI *et al.* (2014) sugere uma estrutura, chamada *ProvenanceLens*, que fornece gerenciamento dados de proveniência em ambientes de nuvem e compara seu desempenho ao usar MySQL, o MongoDB e o Neo4J, SGBD relacional, sistema de banco de dados NoSQL baseado em documentos e sistema de banco de dados NoSQL baseado em grafos, respectivamente. Os pesquisadores elencaram várias vantagens em utilizar um modelo de dados baseado em documentos quando comparado ao modelo de dados relacional, dentre elas a facilidade de incluir novos valores na estrutura de dados de forma dinâmica e ressaltam a possibilidade de melhorar o desempenho das consultas e inserções de dados.

O MongoDB também foi utilizado por SEMPÉRÉ *et al.* (2016) para a camada de armazenamento de dados de uma ferramenta *web* chamada *Gigwa*. Os pesquisadores destacam a capacidade do MongoDB de lidar com grandes conjuntos de dados, seu suporte a consultas complexas por meio agregações e a facilidade em distribuir os dados.

COSTA *et al.* (2017) apresentam o GeNNET, uma plataforma que é capaz de unificar *workflows* científicos com sistemas de bancos de dados baseados em grafos para integrar análise transcricional e selecionar genes relevantes. A plataforma utiliza contêineres de software provisionados pela ferramenta Docker para permitir a portabilidade e reprodutibilidade. O estudo inclui o rastreamento de proveniência e o sistema de banco de dados utilizado para armazená-la é o Neo4J. Os resultados permitem testar hipóteses prévias sobre o experimento, bem como explorar novas por meio do ambiente de banco de dados de grafo.

O trabalho proposto por HONDO *et al.* (2017) apresenta uma modelagem de dados conceitual para gerenciamento de dados de proveniência em ambiente de nuvem computacional baseada no modelo PROV-DM. Além disso, também fornece modelagens de dados específicas para três famílias de NoSQL: grafo, documentos e colunas. HONDO *et al.* (2017) utiliza os sistemas de banco de dados OrientDB, MongoDB e Cassandra. Em suas conclusões relata ter obtido bons resultados com o uso de bancos de dados NoSQL para gerenciamento da proveniência nos moldes do PROV-DM. Dentre os sistemas de banco de dados utilizados, destaca o NoSQL OrientDB pela facilidade com relação a modelagem de dados e armazenamento da proveniência.

Em ALMEIDA *et al.*, (2019) é proposto um simulador web chamado aProvBio capaz de capturar, de forma centralizada e automática, diferentes tipos de proveniência (prospectiva, retrospectiva e definida pelo usuário). O modelo de proveniência considerado para o trabalho foi o PROV-DM, juntamente com o banco de dados Neo4J para o armazenamento dos dados de proveniência em forma de grafo. Os dados brutos utilizados pelos *workflows* não são armazenados no banco de dados. Para validar a modelagem e arquitetura referida foram utilizados dois estudos de casos diferentes - o primeiro envolvendo *workflow* de RNA-Seq de rim e fígado e o segundo envolvendo um *workflow* da bactéria do tipo *Bacillus*.

O trabalho proposto por HASHAM; MUNIR; MCCLATCHEY (2018) apresenta uma estrutura para reproduzir a execução de *workflows* científicos utilizando abordagens de mapeamento para auxiliar na captura das informações de proveniência. Os dados de proveniência são armazenados em banco de dados relacional. Seus resultados mostram que as abordagens de mapeamento propostas podem capturar informações da nuvem em vários cenários durante seu uso sem causar sobrecarga de desempenho e também podem habilitar o reprovisionamento de recursos na nuvem.

HASHAM; MUNIR, (2018) apresenta uma abordagem de coleta de proveniência em três níveis: estrutura do *workflow*, infraestrutura de execução e saída. Os dados são armazenados em um banco de dados relacional. A representação da proveniência é feita por meio de um grafo direcionado acíclico (DAG). Seus resultados mostram que a abordagem im-

plementada pode detectar mudanças nos rastros de proveniência e nas saídas produzidas pelo *workflow*.

MENDES, (2018) apresenta o BioNimbuZ 2, uma plataforma de federação de nuvens em uma arquitetura orientada a microsserviços. A plataforma possui uma arquitetura paralela e distribuída de federação de nuvens com a utilização de microsserviços que facilita a execução de tarefas para o usuário final, de forma transparente onde ele não precise se preocupar com o fato de suas tarefas executarem em uma plataforma de federação.

Esta dissertação conceitualmente difere do modelo de serviço de propostas como o Galaxy (AFGAN *et al.*, 2016), já que ele lida com Infraestrutura como Serviço (IaaS) ao invés de Software como Serviço (SaaS). Também apresenta uma proposta de serviço de gerenciamento de proveniência que pode ser utilizado livremente e acoplado em outras ferramentas de gerenciamento de *workflows*, como por exemplo o BioNimbuZ (MENDES, 2018), que na sua versão atual não armazena dados de proveniência. O gerenciamento de proveniência será disponibilizado para três opções de NoSQL: OritentDB, MongoDB e Cassandra.

A Tabela 2.1 apresenta um resumo dos trabalhos relacionados apresentados anteriormente. Nenhum dos trabalhos tem como objetivo o gerenciamento de dados de proveniência de *workflows* de Bioinformática em ambiente de nuvem federada utilizando sistemas de banco de dados NoSQL para armazenar os dados de forma distribuída. O armazenamento da proveniência é baseado no modelo PROV-DM, que utiliza como padrão um grafo acíclico como estrutura dos dados.

Tabela 2.1: Trabalhos Relacionados.

Autores	Modelo de Proveniência	Banco de Dados	Nuvem Federada
PAULA *et al.* (2013)	PROV-DM	-	-
FERREIRA; FILIPE JR.; OLIVEIRA (2014)	PROV-Wf	PostgreSQL, Cassandra	-
LI *et al.* (2014)	-	MySQL, MongoDB, Neo4J	-
SEMPÉRÉ *et al.* (2016)	-	MongoDB	-
COSTA *et al.* (2017)	-	Neo4J	-
(AFGAN *et al.*, 2016)	OPM	-	-
HONDO *et al.* (2017)	PROV-DM	Cassandra, MongoDB, Orientdb	-
ALMEIDA *et al.* (2019)	PROV-DM	Neo4J	-
HASHAM; MUNIR; MCCLATCHEY (2018)	-	Relacional	-
HASHAM; MUNIR (2018)	DAG	Relacional	-
(MENDES, 2018)	-	-	X
Este trabalho	PROV-DM	Cassandra, MongoDB, Orientdb	X

É importante ressaltar que nenhum dos trabalhos tem como objetivo o gerenciamento de dados de proveniência de*workflows* de Bioinformática para um ambiente de nuvem federada. Para isso é proposto um esquema de dados conceitual de proveniência baseado no PROV-DM, bem como o mapeamento para três modelagens de dados específicas das famílias de colunas, documento e grafo.

Capítulo 3

Modelagem de dados e Arquitetura de Gerenciamento de Dados de Proveniência Propostos

Este capítulo apresenta a descrição das informações sobre a modelagem de dados e arquitetura para o gerenciamento de dados de proveniência de *workflows* de Bioinformática em ambiente de nuvem federada propostos para esta dissertação. Este capítulo está divido da seguinte maneira: a Seção 3.1 trata sobre a definição dos elementos do ambiente de nuvem federada; a Seção 3.2 define a modelagem de dados para cada sistema de banco de dados NoSQL OrientDB, MongoDB e Cassandra; na Seção 3.3 é definida a arquitetura de sistema do serviço de gerenciamento de proveniência em ambiente de nuvem federada; na Seção 3.4 é especificada a arquitetura de infraestrutura da Plataforma de execução dos *workflows* de Bioinformática e armazenamento dos dados de proveniência no ambiente de nuvem federada.

3.1 Elementos do Ambiente de Nuvem Federada

Experimentos executados em nuvem federada possuem características que foram tratadas nesta dissertação. Segundo BIRAN *et al.* (2017), uma federação ocorre quando sistemas (provedores de serviços) alocam recursos dos provedores de nuvens (provedores de serviços de nuvem), armazenar os dados referentes ao provisionamento do ambiente é relevante para a proveniência de dados. Assim, a proveniência mostra a rastreabilidade do processamento tornando possível identificar se a execução do experimento ocorreu em um ambiente de nuvem computacional ou em um ambiente de nuvem federada.

HONDO (2018) identificou quatro elementos relevantes para a proveniência de dados em um ambiente de nuvem computacional: Provedor, *Cluster*, Local e Máquina. Em

um ambiente de nuvem federada, tais elementos também devem ser considerados. Porém, novos elementos devem ser adicionados. Para adicionar os novos elementos de proveniência em um ambiente de nuvem federada, foi realizado um estudo nos provedores de nuvem. A seguir, esses elementos são definidos.

A escolha dos provedores de nuvem para executar um experimento, pode ser feita tanto pelo usuário (agente) como pelo próprio sistema de gerenciamento de *workflows* (provedor de serviço). Em ambos os casos, é necessário escolher a localização onde se deseja provisionar as máquinas virtuais. A localização é composta pela região geográfica específica do *datacenter* que irá hospedar as máquinas e a zona de disponibilidade dos recursos. O preço e a latência de comunicação entre as máquinas podem ser influenciados diretamente pela localização escolhida. A Figura 3.1 e a Figura 3.2 mostram a diferença de preço de uma máquina virtual sendo provisionada pela Google Cloud Platform na região asia-northeast1(Tóquio) e outra máquina com as mesmas configurações sendo provisionada na região southamerica-east1(São Paulo), onde o preço mensal é de US$35.99 e US$44.20 respectivamente.

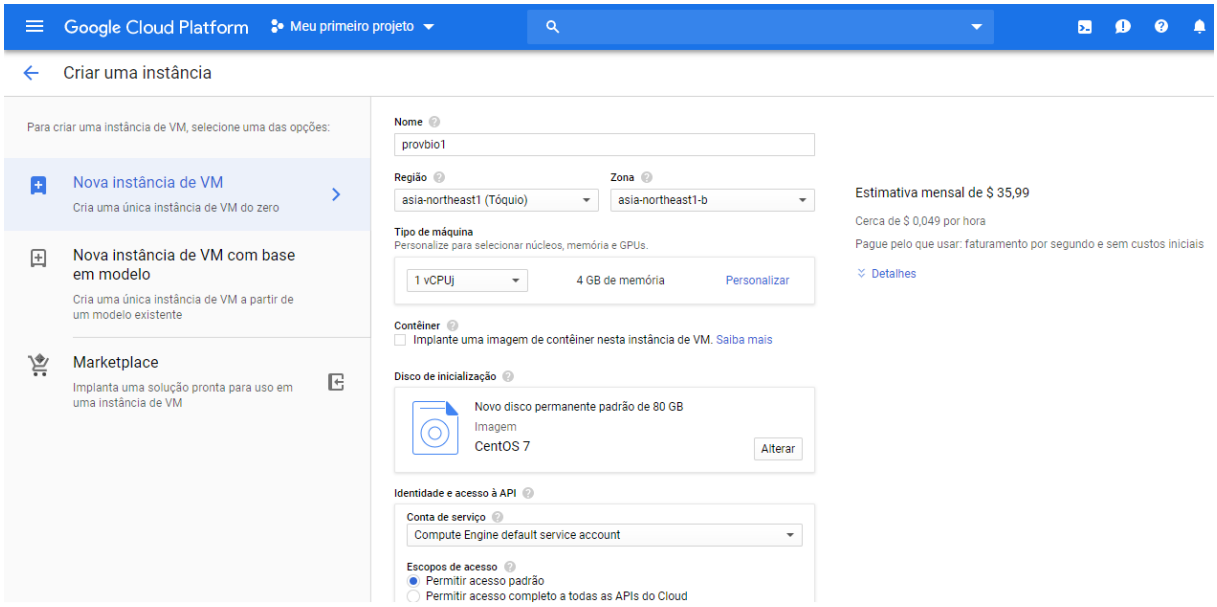


Figura 3.1: Google Cloud Região Tóquio (Google, 2019).

.

Outro exemplo de tarifação pode ser observado nas Figura 3.3 e Figura 3.4, onde o preço por hora é de US$0,0104 se a máquina for hospeda na região de São Paulo e US$0,089 se for hospedada na região de Toronto. Ainda com relação à tarifação, nos provedores de nuvem utilizados por esta dissertação, observa-se que a cobrança pode ser mensal ou por hora.

Ao configurar uma máquina virtual, o tipo de autenticação da conta de administrador pode ser por meio de nome de usuário e senha ou chaves *Secure Shell* (SSH). Também

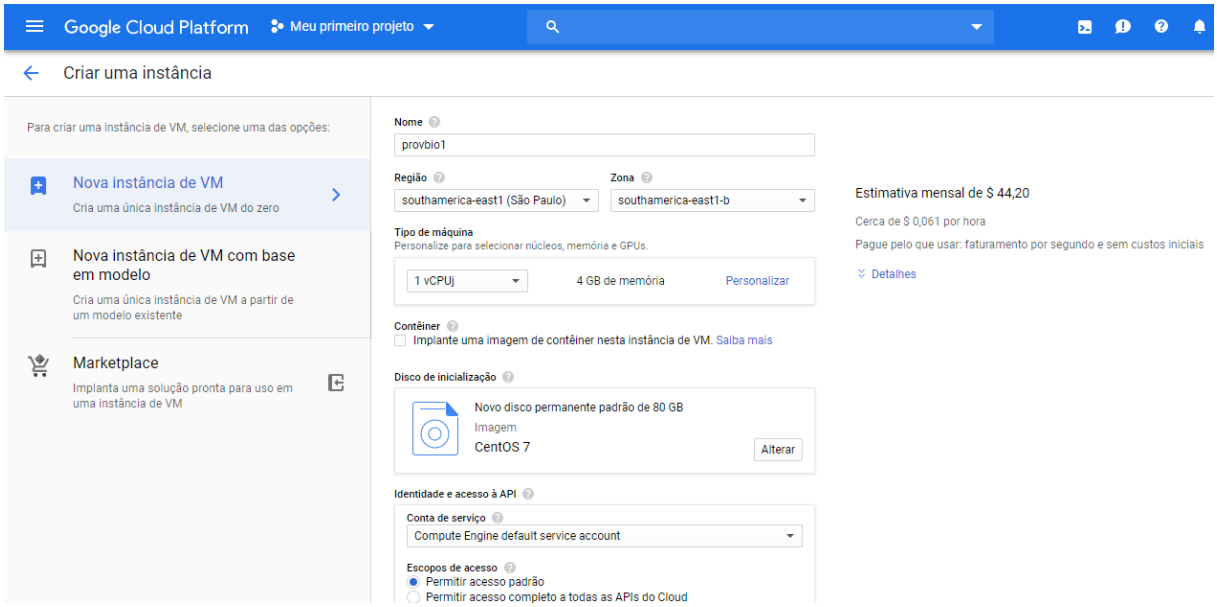


Figura 3.2: Google Cloud Região São Paulo (Google, 2019).

.

é possível definir quais portas de rede da máquina virtual podem ser acessadas pela internet pública. Ao criar uma máquina virtual, um adaptador de rede é criado contendo as informações do IP e sub-rede. Essas informações são importantes para permitir a comunicação dos nós do *cluster* hospedado em nuvem federada.

As configurações utilizadas pelas máquinas virtuais onde os experimentos são executados são relevantes para reprodutibilidade dos experimentos em nuvem federada, pois implicam diretamente na comunicação do *cluster*, no tempo de execução e custo dos experimentos. Com essas informações é possível avaliar o custo financeiro, o tempo de execução de cada atividade e de cada fase do *workflow*, o tempo total de execução, etc. Essas informações podem ser analisadas considerando os provedores e a localização dos nós do *cluster* onde os dados são armazenados de forma distribuída, e também devem ser consideradas pela proveniência de dados em um ambiente de nuvem federada.

3.1.1 Nuvens Computacionais Utilizadas Neste Trabalho

Existem diversos provedores de nuvem públicos que oferecem serviços aos usuários. No escopo desta dissertação, considerou-se três provedores: Microsoft Azure, DigitalOcean e Google Cloud. O requisito que levou a escolha desses provedores foi a disponibilização de *vouchers* que possibilitaram a hospedagem das máquinas virtuais gratuitamente. Além dos provedores de nuvem públicos, também foi disponibilizado acesso aos serviços da nuvem privada CloudJus do Supremo Tribunal Federal para a execução dos experimentos.

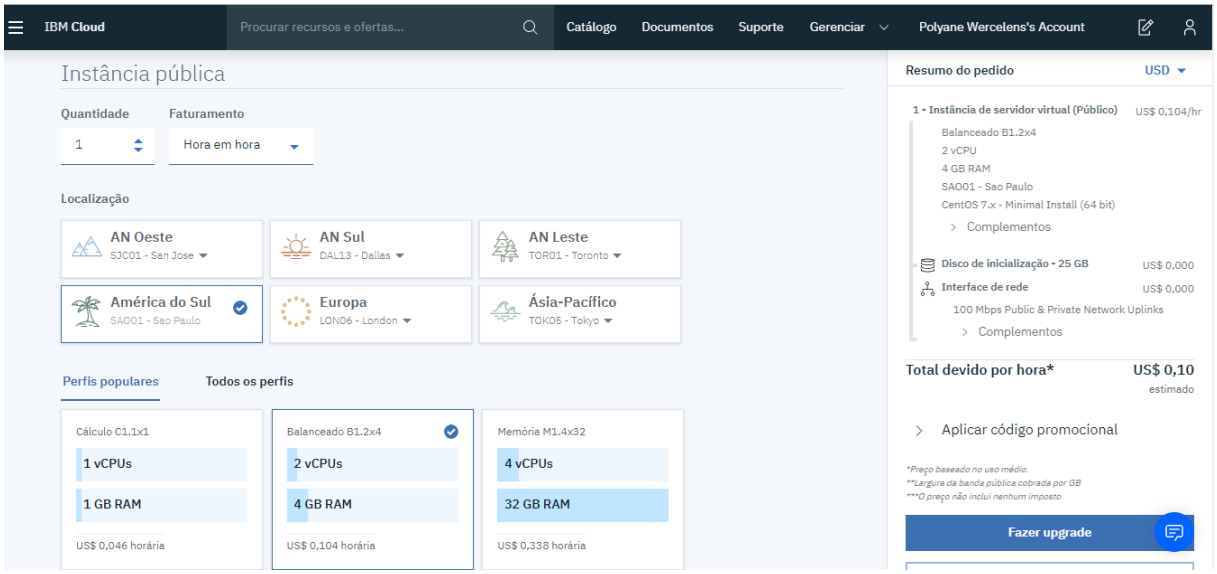


Figura 3.3: IBM Cloud São Paulo (IBM, 2019).

.

Microsoft Azure

O Microsoft Azure é um conjunto de serviços de nuvem que permite a criação, gerenciamento e implantação de aplicativos em uma rede global utilizando ferramentas e estruturas conforme a preferência do usuário. Para a Microsoft Azure, uma região é um conjunto de *datacenters* implantados dentro de um perímetro de latência definida e conectados por meio de uma rede regional dedicada de baixa latência. Sua infraestrutura já está disponível em 44 regiões do mundo e possui planos anunciados em outras 10 regiões, totalizando 54 regiões. Uma localidade geográfica, normalmente possui duas ou mais regiões. As localizações são tolerantes a falhas de região completa. As zonas de disponibilidade são locais separados fisicamente dentro de uma região do Azure. Cada zona de disponibilidade é composta por um ou mais *datacenters* equipados com energia, resfriamento e rede independentes. Elas permitem que os clientes executem aplicativos críticos com alta disponibilidade e replicação de baixa latência (Microsoft, 2019). A definição de localização da instância implica no preço.

Para novos usuários a Microsoft disponibiliza uma assinatura "Avaliação Gratuita" que possui um crédito R$ 750,00. Essa assinatura é limitada a uma por cliente e os créditos podem ser utilizados em um período de até 12 meses.

DigitalOcean

DigitalOcean é uma empresa que tem como objetivo simplificar a computação em

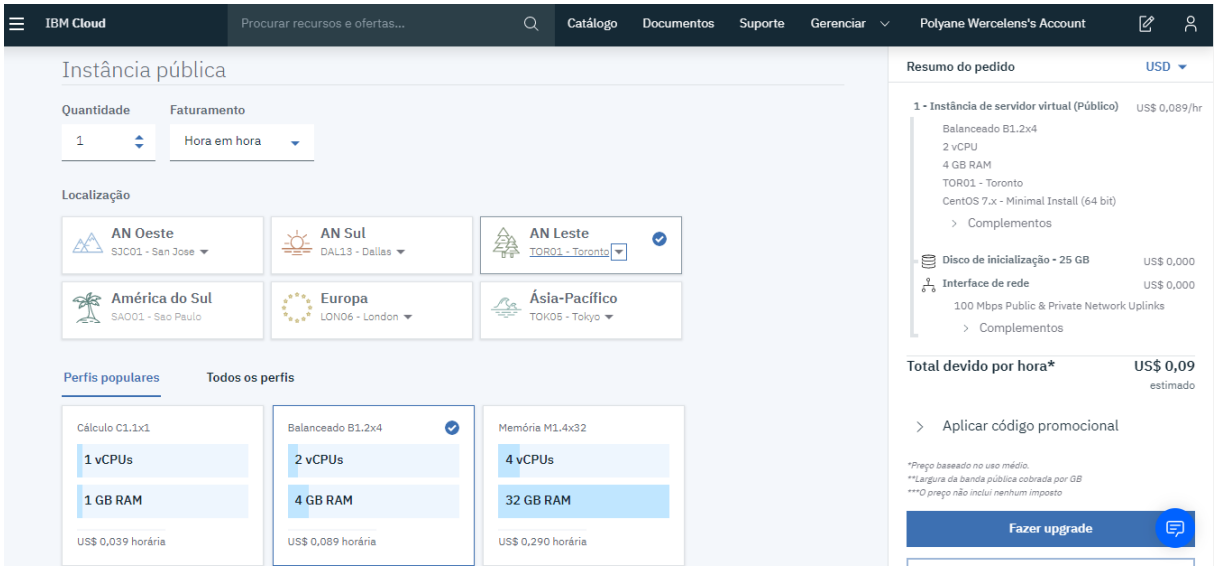


Figura 3.4: IBM Cloud Região Toronto (IBM, 2019).

.

nuvem, para que os desenvolvedores e suas equipes possam dedicar mais tempo à criação de software. Assim como a Microsoft Azure, a DigitalOcean trabalha com os conceitos de localização, região e zona. Sua infraestrutura está disponível em 12 *datacenters* em 8 regiões (DigitalOcean, 2019). A definição de localização da instância não implica no preço.

A empresa disponibiliza *vouchers* de US$10 que podem ser adicionados à conta do usuário por meio de códigos e utilizados para o pagamento da fatura. Outra forma de obter os *vouchers* é através das referências, ou seja, indicando por email ou por meio de compartilhamento de link. Todas as indicações recebem um crédito de US$50 para uso durante 30 dias. Quando as pessoas indicadas utilizarem US$25, a empresa concede um *voucher* de US$25 para quem indicou. Não há limite para a quantidade de crédito que se pode ganhar por meio de indicações.

Google Cloud

O Google Cloud Platform está disponível em 20 regiões e 61 zonas de disponibilidade. Trata-se de um conjunto de ativos físicos e recursos virtuais hospedados em *datacenters* do Google em mais de 200 países ou territórios. Os recursos e produtos são disponibilizados para o usuário como serviço, onde é possível combiná-los para obter estrutura necessária para seu projeto (Google, 2019). A definição da localização da máquina virtual implica em valores de custo diferente para cada região.

Assim como a Microsoft Azure, o Google Cloud Platform disponibiliza para novos usuários um crédito de US$ 300 que pode ser utilizado em um período de 12 meses

limitando um crédito por usuário.

3.2 Modelagem de Dados de Proveniência

Com intuito de demonstrar a proveniência de dados de *workflows* de Bioinformática de forma objetiva, PAULA *et al.*, (2013) utilizou elementos do PROV-DM que obtiveram resultados satisfatórios: Atividade, Agente e Entidade. Ele definiu um esquema de dados conceitual baseado no PROV-DM contendo as entidades Arquivo, Experimento, Projeto, Atividade e Agente, possibilitando a representação gráfica da proveniência por meio de um grafo. O uso do modelo PROV-DM mostrou-se adequado para representar a proveniência em projetos de Bioinformática de forma simples e direta (GONÇALVES *et al.*, 2013), (PAULA *et al.*, 2013), (OLIVEIRA *et al.*, 2014), (ALMEIDA *et al.*, 2019).

A fim de descrever as questões sobre o uso de proveniência de dados da Bioinformática em um ambiente de nuvem computacional, HONDO *et al.* (2017) definiu quatro entidades: Provedor, *Cluster*, Local e Máquina. Seu trabalho mostrou que a modelagem proposta é viável e genérica suficiente inclusive para ser adaptada a outros tipos de *workflows*.

Considerando as entidades definidas por PAULA *et al.* (2013) e HONDO *et al.* (2017), e observando as particularidades de um ambiente de nuvem federada, é proposto o esquema de dados conceitual para gerenciamento de dados de proveniência de *workflows* de Bioinformática apresentado na Figura 3.5 e seu dicionário de dados contendo a descrição de cada atributo das entidades na Tabela 3.1. No modelo de dados a cor amarela representa as entidades definidas por (PAULA *et al.*, 2013), a cor azul representa as entidades definidas por (HONDO *et al.*, 2017) e a cor verde representa as entidades definidas por (HONDO *et al.*, 2017) que foram alteradas para armazenar os elementos necessários para um ambiente de nuvem federada.

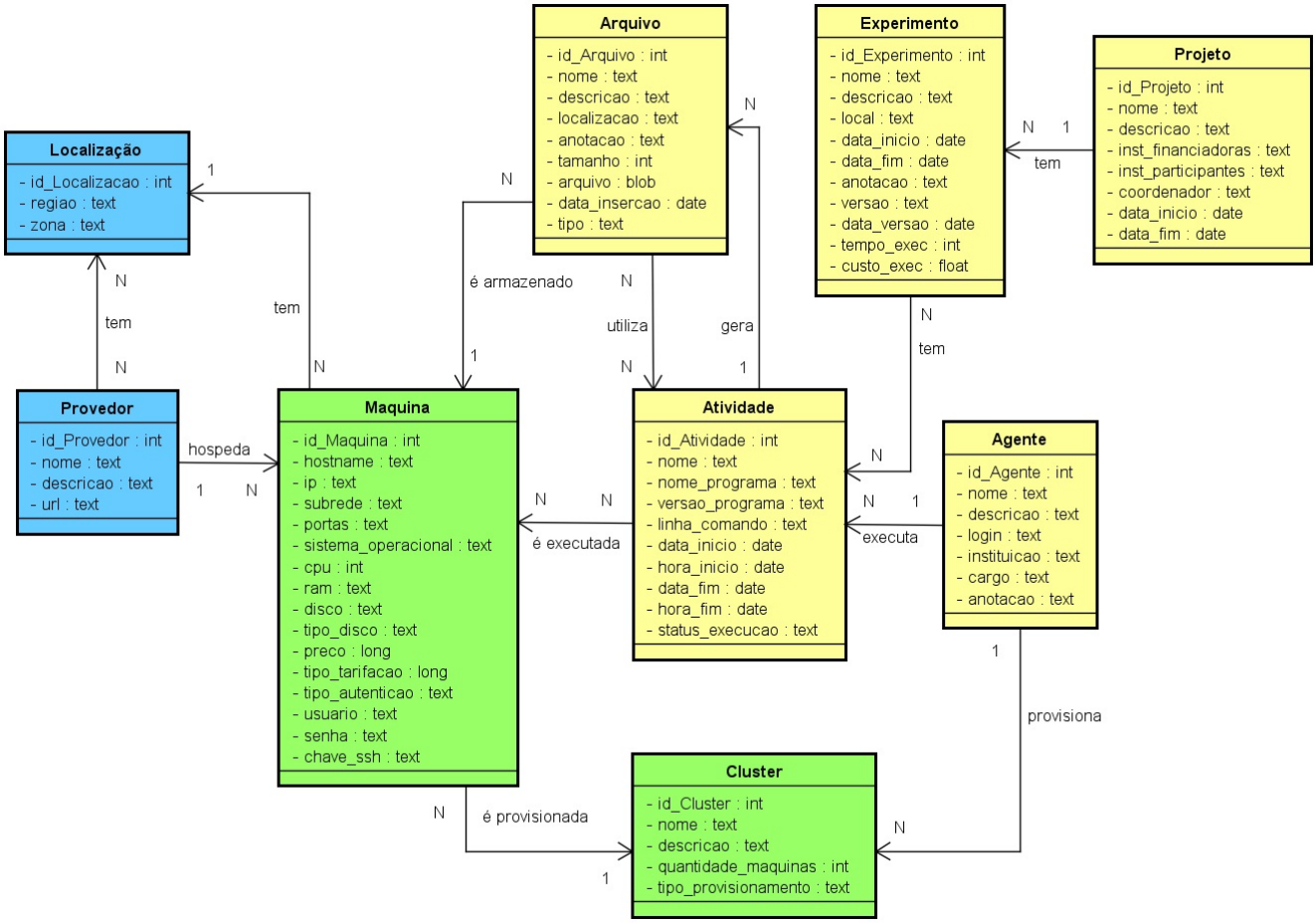


Figura 3.5: Esquema de Dados Conceitual de Proveniência.

Tabela 3.1: Descrição dos Atributos do Esquema de Dados Conceitual de Proveniência.

Projeto	
Id_Projeto	Identificador do projeto
Nome	Nome do projeto
Descricao	Descrição do projeto
Coordenador	Coodenador do projeto
Inst_Financiadoras	Instituições que financiam o projeto
Inst_Participantes	Instituições que participam do projeto
Data_Inicio	Data de início do projeto
Data_Fim	Data de encerramento do projeto
Atividade	
Id_Atividade	Identificador da atividade
Nome	Nome da atividade
Linha_Comando	Linha de comando contendo os parâmetros utilizados

Programa	Nome do programa executado
Versao_Programa	Versão do programa executado
Data_Inicio	Data de início da execução da atividade
Hora_Inicio	Hora de início da execução da atividade
Data_Fim	Data final da execução da atividade
Hora_Fim	Hora final da execução da atividade
Status_Execucao	Status da execução da atividade
Máquina	
Id_Maquina	Identificador da máquina
Hostname	Nome utilizado para identificação da máquina
CPU	Quantidade de CPUs alocadas
Sistema_Operacional	Nome do sistema operacional utilizado
RAM	Quantidade de memória RAM alocada
Disco	Tamanho do disco alocado
Tipo_Disco	Tipo de disco. Exemplo: SSD
Preco	Preço da máquina
Tipo_Tarifacao	Tipo de tarifação: mensal ou hora de uso
IP	Endereço IP da máquina
Subrede	Intervalo de endereços IP que podem ser utilizados na rede da máquina virtual
Tipo_Autenticacao	Indica se a conta de administrador da máquina usará o nome de usuário/senha ou chaves SSH para a autenticação
Usuario	Nome do usuário administrador da máquina virtual
Senha	Senha do usuário administrador da máquina virtual
Chave_SSH	Chave SSH utilizada para o acesso à máquina virtual
Portas	Portas liberadas para acesso pela internet
Provedor	
Id_Provedor	Identificador do provedor de nuvem
Nome	Nome do provedor de nuvem
Descricao	Descrição do provedor de nuvem
URL	URL do provedor de nuvem
Localizacao	
Id_Localizacao	Identificador da localização geográfica
Regiao	Localização geográfica específica
Zona	Zona de disponibilidade dos recursos

Experimento	
Id_Experimento	Identificador do experimento
Nome	Nome do experimento
Descricao	Descrição do experimento
Versao	Número da versão do experimento
Data_Versao	Data da versão do experimento
Anotacao	Anotações adicionais sobre o projeto
Custo_Exec	Curso total da execução do experimento
Tempo_Exec	Tempo total da execução do experimento
Local	Local de execução
Data_Inicio	Data de início da execução
Data_Fim	Data final da execução
Arquivo	
Id_Arquivo	Identificador do arquivo
Nome	Nome do arquivo
Descricao	Descrição do arquivo
Data_Insercao	Data de inserção do arquivo
Arquivo	Arquivo físico
Tamanho	Tamanho do arquivo
Anotacao	Anotações adicionais sobre o arquivo
Localizacao	Localização do arquivo
Tipo	Especifica o tipo do arquivo: Arquivo de Entrada ou Arquivo de Saída
Agente	
Id_Agente	Identificador do agente
Nome	Nome do usuário
Descricao	Descrição do usuário
Instituicao	Instituição a qual o usuário pertence
Cargo	Cargo ou função do usuário
Login	Login de acesso do usuário
Anotacao	Anotações adicionais sobre o usuário
Cluster	
Id_Cluster	Identificador do cluster
Nome	Nome do cluster
Descricao	Descrição do cluster

Quantidade_Maquinas	Quantidade de máquinas provisionadas
Tipo_Provisionamento	Forma como as máquinas virtuais foram provisionadas no Cluster: Usuário ou Provedor

Projetos de Bioinformática consistem na execução de diversos experimentos, que podem ser compostos por diversas atividades. As atividades utilizam arquivos e dados de entrada e geram arquivos e dados de saída. Elas são executadas por um agente em máquinas que podem ser instanciadas em um único provedor de nuvem ou em um ambiente de nuvem federada. Cada máquina possui uma localização (região e zona) e configurações específicas para possibilitar a federação de nuvens, como portas liberadas para acesso via internet, tipo de autenticação SSH ou por meio de usuário e senha, e sub-rede contendo um intervalo de endereços IP que podem ser utilizados na rede da máquina virtual. Dessa forma, os atributos subrede, portas, tipo_autenticacao, usuario, senha e chave_ssh foram adicionados a entidade Maquina.

O provisionamento das máquinas nos provedores de nuvem e sua alocação no *cluster* pode ser feito pelo agente ou pelo próprio provedor de serviços. Assim, a entidade *Cluster* recebeu o atributo tipo_provisionamento para indicar a forma como as máquinas foram provisionadas: pelo usuário (agente) ou pelo provedor. Quando a máquina é provisionada pelo agente, as informações a respeito do agente podem ser consultadas por meio do relacionamento que foi incluído entre as entidades *Cluster* e Agente.

Para que os dados de proveniência sejam persistidos corretamente em cada sistema de banco de dados NoSQL, é necessário definir uma modelagem que especifique os objetos de banco de dados necessários considerando as particularidades de cada NoSQL. Dessa forma, foram levantadas as características dos sistemas OrientDB (baseado em grafo), MongoDB (baseado em documentos) e Cassandra (baseado em colunas).

3.2.1 Modelagem de Dados OrientDB

O OrientDB é um sistema de banco de dados NoSQL híbrido que implementa as famílias de grafo, documento e chave-valor. Ele suporta uma arquitetura de banco de dados distribuída com replicação. O armazenamento de dados pode ser feito na memória e no disco (FERNANDES; BERNARDINO, 2018). As linguagens de manipulação que possuem suporte nativo no OrientDB são: Java, SQL e Gremlin. Os objetos do do banco de dados e os dados podem ser manipulados também por meio de interface gráfica nativa (Figura 3.7).

Formalmente, um grafo é uma coleção de vértices e arestas, ou seja, um conjunto de nós e os relacionamentos que os conectam (DEO, 2017). Os modelos de grafos facilitam a

modelagem de dados, pois aproximam os domínios técnicos aos domínios de negócio. Os esquemas de dados são flexíveis possibilitando alterações e inclusões de novas entidades e relacionamentos sem necessidade de reestruturação do banco de dados. Eles representam entidades como nós e a forma como essas entidades se relacionam como relacionamentos (ROBINSON; WEBBER; EIFREM, 2013).

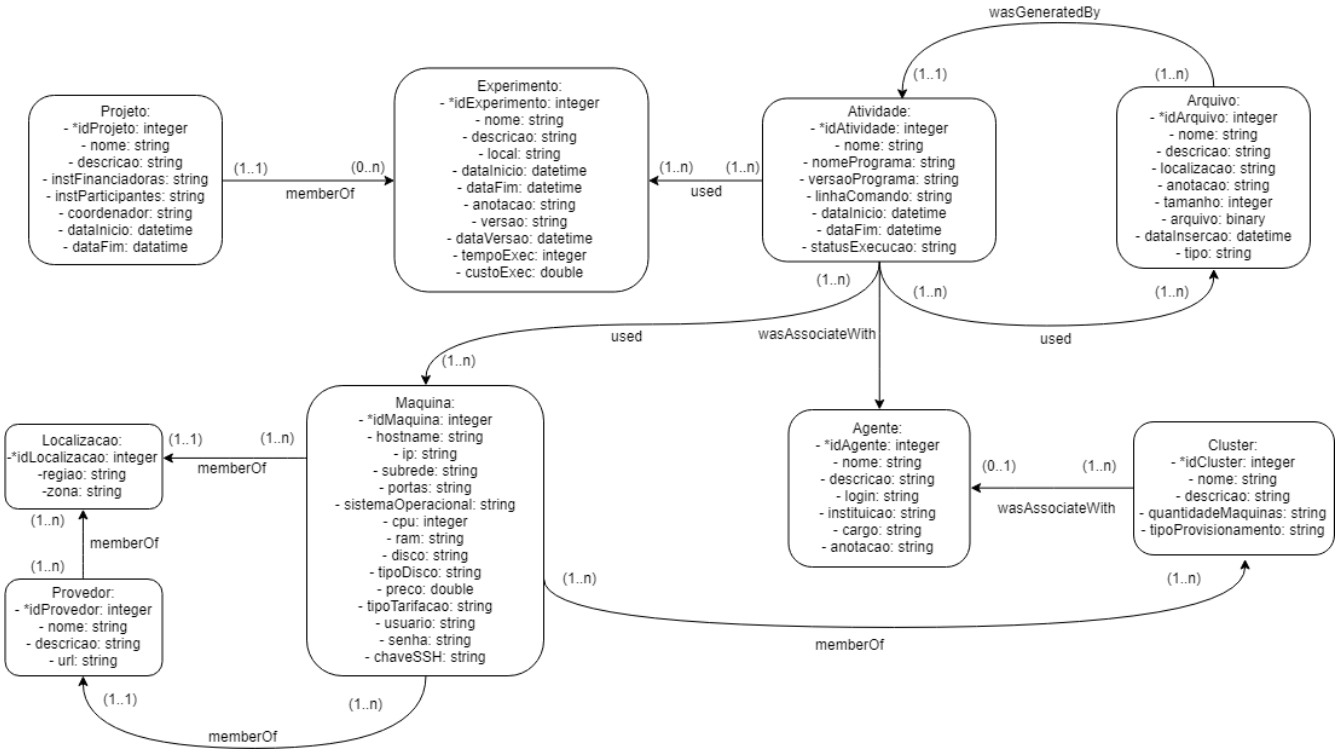


Figura 3.6: Modelagem Proposta para o Sistema de Banco de Dados Baseado em Grafo.

.

A Figura 3.6 apresenta o modelo de dados baseado em grafo para o armazenamento da proveniência no OrientDB. Este modelo foi desenhado utilizando o diagrama de descrição de grafos para bancos de dados de grafo definido por (ERVEN *et al.*, 2019). As entidades presentes no esquema de dados conceitual (Figura 3.5) foram mapeadas para o modelo de grafo, onde se transformaram em vértices do grafo. Os vértices Projeto, Experimento, Localizacao, Provedor, Máquina, *Cluster* e Arquivo representam as entidades do PROV-DM, o vértice Atividade representa as atividades e o vértice Agente os agentes. Os relacionamentos entre as entidades foram mapeados para arestas direcionadas rotuladas seguindo o modelo PROV-DM.

3.2.2 Modelagem de Dados MongoDB

O MongoDB é um sistema de banco de dados NoSQL orientado a documentos. É proposto como uma solução fácil de escalar e possui esquema flexível. A abordagem

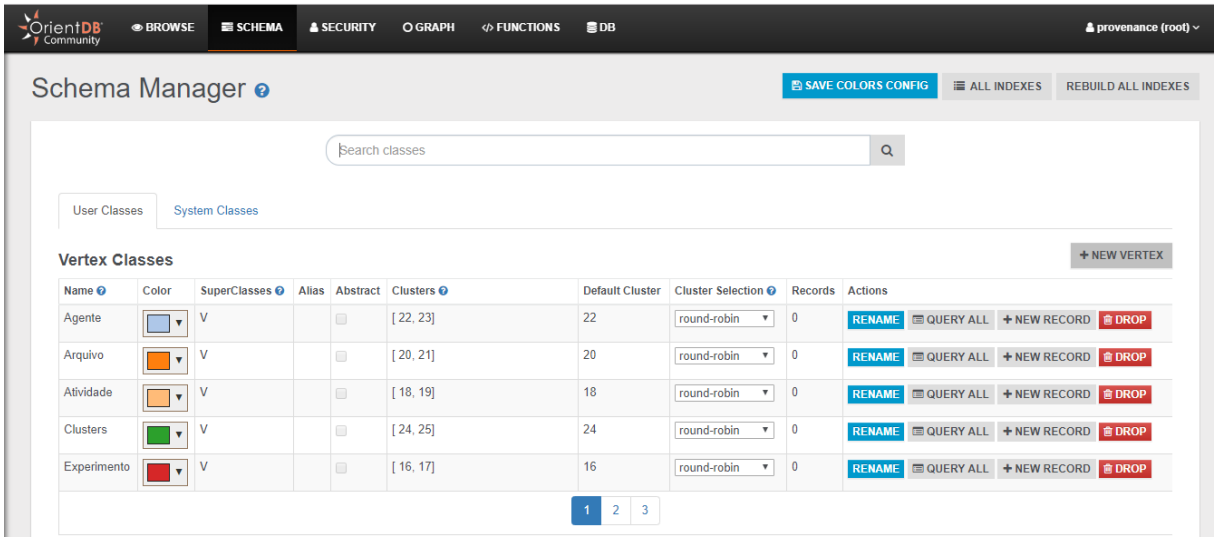


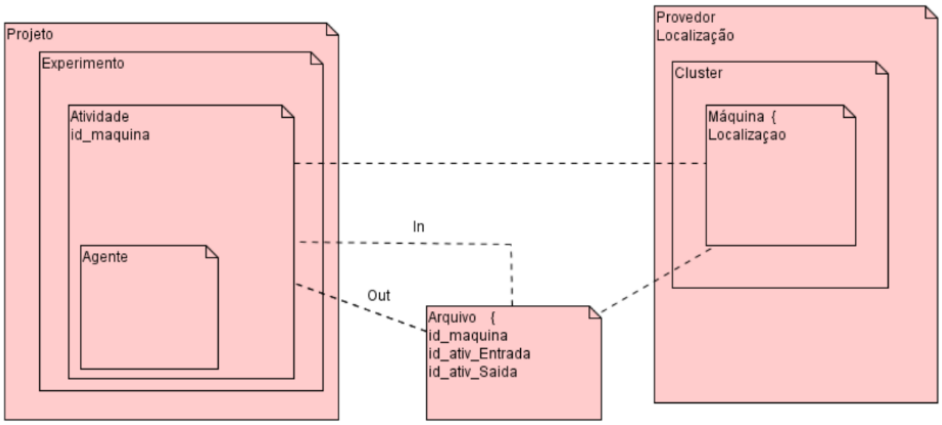
Figura 3.7: Interface Gráfica do OrientDB.

orientada a documentos possibilita a representação de relações hierárquicas complexas por meio de um único registro, permitindo documentos e *arrays* embarcados (KAUR; RANI, 2013). Além dos documentos embarcados, também é possível relacionar os documentos por meio de referência ou link (CHODOROW, 2013).

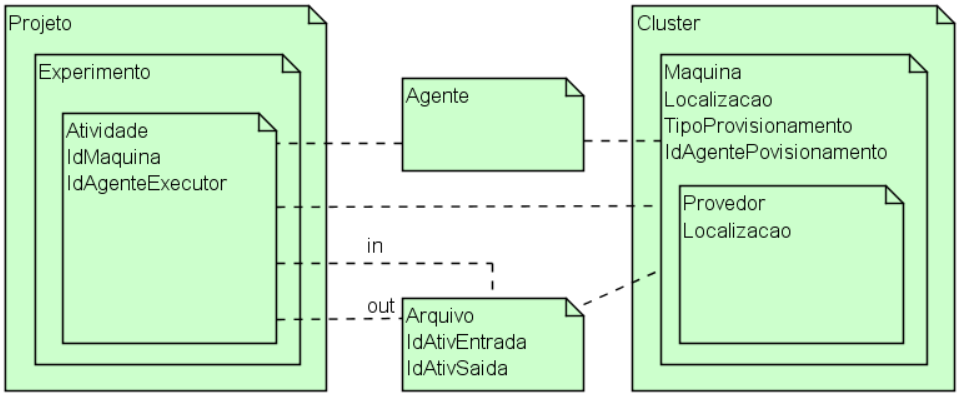
No modelo de dados de documentos proposto por (HONDO *et al.*, 2017) (Figura 3.8a), uma máquina era um documento embarcado no documento *Cluster*, que por sua vez era um documento embarcado no documento Provedor. Tal modelagem é aderente quando os experimentos são executados em ambiente nuvem computacional, pois quando se está trabalhando com apenas um provedor de nuvem, as máquinas provisionadas no *cluster* são necessariamente de um único provedor. No contexto de nuvem federada, um *cluster* pode ter máquinas de diversos provedores de nuvem. Dessa forma, o modelo de documento proposto por este trabalho (Figura 3.8b), define que um *cluster* possui diversas máquinas que estão alocadas em seus respectivos provedores de nuvem. Além disso, foi incluído o atributo tipo_provisionamento para informar se a máquina foi provisionada pelo agente ou pelo provedor de serviços.

Outra alteração no modelo foi no documento Agente, que deixou de ser embarcado no documento Atividade e passou a se relacionar por referência. Também foi acrescentado um relacionamento entre Agente e Máquina que será utilizado quando o usuário escolher os provedores de nuvem onde se deseja alocar as máquinas virtuais. Assim, o modelo será capaz de responder aos experimentos executando em nuvem federada.

Os dados são armazenados no MongoDB em documentos no formato JSON. Assim, foi definido um esquema de dados físico apresentado na Figura 3.10a, Figura 3.9a, Figura 3.9b e Figura 3.10b, composto pelos documentos Projeto, Agente, Arquivo e *Cluster* respectivamente, seus atributos e documentos embarcados.

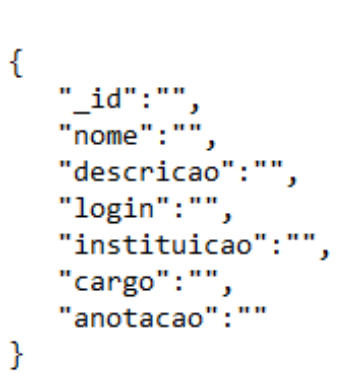


(a) Modelagem para o Sistema de Banco de Dados Baseado em Documento - Nuvem Computacional (HONDO, 2018).



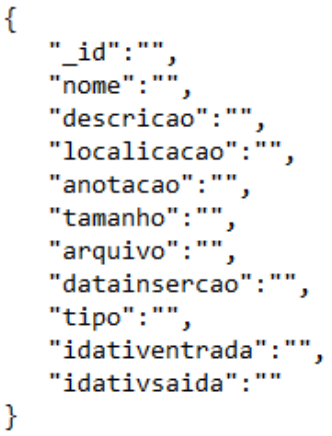
(b) Modelagem Proposta para o Sistema de Banco de Dados Baseado em Documento - Nuvem Federada.

Figura 3.8: Modelagem de Dados Baseada em Documento.



```
{
   "_id":"",
   "nome":"",
   "descricao":"",
   "login":"",
   "instituicao":"",
   "cargo":"",
   "anotacao":""
}
```

(a) Documento Agente.



```
{
   "_id":"",
   "nome":"",
   "descricao":"",
   "localicacao":"",
   "anotacao":"",
   "tamanho":"",
   "arquivo":"",
   "datainsercao":"",
   "tipo":"",
   "idativentrada":"",
   "idativsaida":""
}
```

(b) Documento Arquivo.

Figura 3.9: Esquema de Dados Físico do MongoDB - Agente e Arquivo.

(a) Documento Projeto.

(b) Documento *Cluster*.

Figura 3.10: Esquema de Dados Físico do MongoDB - Projeto e Cluster.

3.2.3 Modelagem de Dados Cassandra

O Cassandra é um sistema de banco de dados NoSQL da família de colunas, seu modelo de dados é desnormalizado, projetado para capturar e consultar dados rapidamente. Embora seus objetos sejam similares aos objetos de um banco relacional, como por exemplo tabelas, chaves primárias e índices, a modelagem de dados não deve seguir o paradigma entidade-relacionamento. O modelo de dados do Cassandra pode ser descrito como um armazenamento particionado de linha, onde os dados são armazenados em tabelas *hash sparse* multidimensionais. *Sparse* significa que para qualquer linha é possível ter uma ou mais colunas, mas cada linha não precisa ter todas as mesmas colunas que outras linhas similares (CARPENTER; HEWITT, 2016).

O modelo de dados do Cassandra busca atender as consultas que serão submetidas ao banco de dados. Dessa forma, é importante analisar como o dado será consultado pelas aplicações e preparar o modelo para atendê-las, com intuito de gerar tabelas mais adequadas e eficientes. Operações como *joins* e agregações presentes na linguagem SQL, não estão presentes na linguagem de consulta do Cassandra (CHEBOTKO; KASHLEV; LU, 2015). Assim, para propor uma modelagem de dados baseada em colunas, foi necessário analisar questões elaboradas por biólogos, considerando a forma como os dados serão acessados e a geração do grafo de proveniência. A Figura 3.11 mostra o modelo de dados proposto composto por duas entidades: ExpBioAmbienteComputacional e ExpBioAtividade.

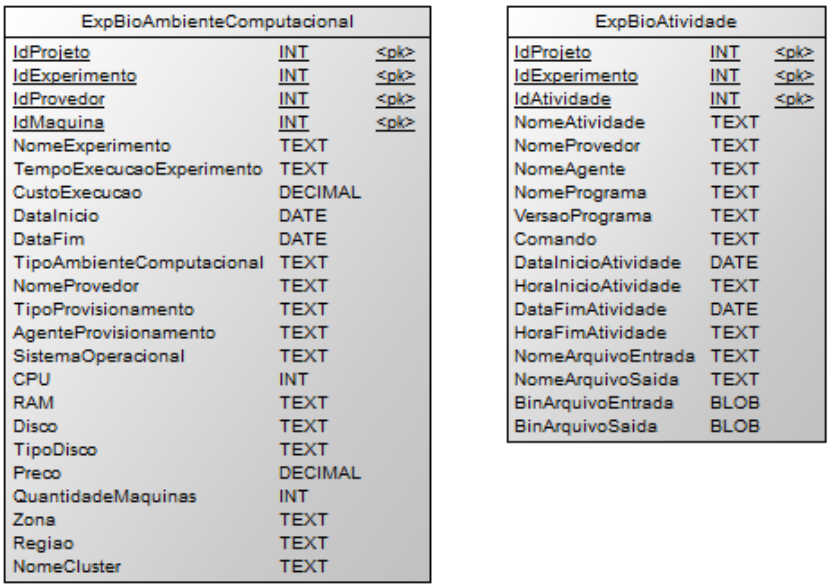


Figura 3.11: Modelagem Proposta para o Sistema de Banco de Dados Baseado em Colunas.

Para responder consultas relativas ao ambiente computacional onde os experimentos foram executados, a tabela ExpBioAmbienteComputacional foi projetada contendo

informações do projeto, experimento, provedor, máquina, localização e tipo de provisionamento do ambiente computacional. O atributo TipoAmbienteComputacional informa se o ambiente computacional é nuvem federada ou nuvem computacional. O TipoProvisionamento indica se as máquinas foram provisionadas por um agente ou pelo provedor de serviços. Se o provisionamento foi feito por um provedor de serviço, trata-se de um ambiente de nuvem federada. Se o provisionamento foi feito por um agente, o nome do agente estará preenchido no atributo NomeAgente.

A tabela ExpBioAtividade foi projetada contendo informações relativas às atividades executadas, projeto, experimento, provedor e arquivos de entrada e saída utilizados pelas atividades. Essa tabela contém as informações necessárias para a geração do grafo de proveniência.

3.3 Serviço de Gerenciamento de Proveniência em Ambiente de Nuvem Federada

A Figura 3.12 apresenta a arquitetura de sistema do serviço construído para gerenciar os dados de proveniência durante a execução de diferentes *workflows* da Bioinformática em ambiente de nuvem federada, utilizando a modelagem de dados de proveniência proposta baseada no PROV-DM e implementada em três sistemas de bancos de dados NoSQL das famílias de Grafo (OrientDB), Documento (MongoDB) e Colunas (Cassandra). A linguagem de programação utilizada para codificar foi Python. Os critérios utilizados para selecionar a linguagem foi a familiaridade e experiência prévia com a tecnologia, o que facilitou desenvolvimento.

O serviço pode ser acessado por aplicações clientes e usuários e é composto por quatro módulos:

- Módulo de Execução: responsável por cadastrar os dados referentes aos experimentos e executar as atividades dos *workflows*;
- Módulo de Gerenciamento de Proveniência: coleta os dados definidos pelo usuário no cadastro das informações referentes aos experimentos, dados de proveniência prospectiva referentes a especificação das tarefas computacionais cadastradas e dados de proveniência retrospectiva referentes às tarefas em execução no módulo de execução. A proveniência é enviada para o componente de armazenamento;
- Módulo de Persistência: tem como função persistir a proveniência de dados na forma de grafo, documentos e família de colunas, e recuperar os dados das consultas submetidas pelo módulo de consulta;

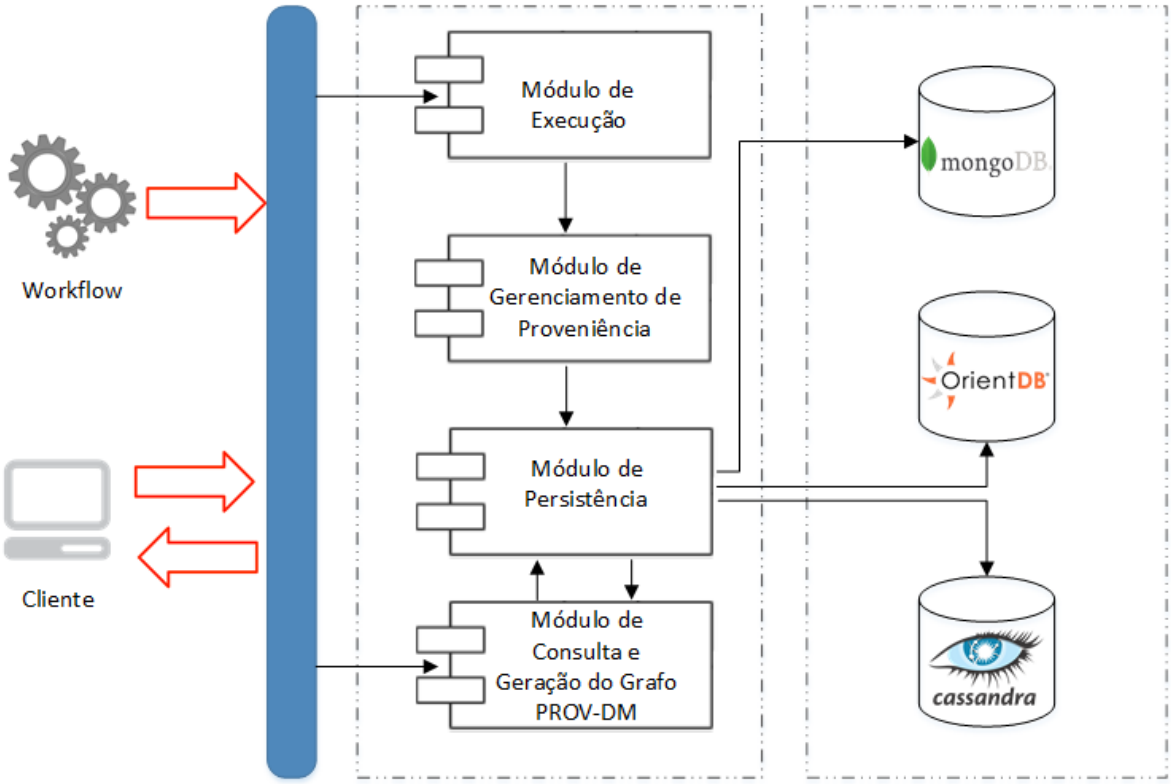


Figura 3.12: Serviço de Gerenciamento de Proveniência.

- Módulo de Consulta e Geração de Grafo de Proveniência: gerencia as requisições de consultas, a geração e recuperação dos grafos de proveniência.

3.3.1 Módulo de Execução

Para que as aplicações clientes e usuários acessem a arquitetura do sistema proposto foi implementado o módulo de execução que pode ser acessado por meio de linha de comando ou pode ser integrado a sistemas gerenciadores de *workflows*.

Neste módulo é possível solicitar o cadastro dos dados definidos pelo usuário, como por exemplo o projeto, e os dados de proveniência prospectiva, como por exemplo os experimentos e as atividades que os compõem. Além disso, os experimentos podem ser executados por meio de comandos que são enviados deste módulo para o sistema operacional da máquina onde ele será executado utilizando a biblioteca os.py[1].

3.3.2 Módulo de Gerenciamento de Proveniência

Neste módulo os dados de proveniência são coletados de acordo com o seu tipo: dados definidos pelo usuário, proveniência prospectiva e proveniência retrospectiva. O módulo

[1]https://docs.python.org/3/library/os.html

de execução possui uma funcionalidade que solicita o cadastro dos dados definidos pelo usuário e os dados de proveniência prospectiva. Esses dados são capturados e enviados ao módulo de persistência de dados a fim de serem armazenados nos bancos de dados. Caso ocorra algum erro durante a transação de cadastro, este módulo retorna para o módulo de execução a informação de que ocorreu uma falha.

Durante a execução dos experimentos no módulo de execução, este módulo captura os dados referentes às etapas que estão sendo executadas, a situação de cada tarefa em execução, bem como todas as informações relativas ao ambiente computacional que está sendo utilizado, inclusive quando executado em nuvem federada. Assim, a proveniência retrospectiva é enviada ao módulo de persistência para ser gravada nos bancos de dados.

3.3.3 Módulo de Persistência

As famílias de banco de dados NoSQL suportadas por este serviço são: colunas, documentos e grafos. O módulo de persistência recebe os dados do módulo de gerenciamento de proveniência e armazena nos três sistemas de banco de dados NoSQL: Cassandra, MongoDB e OrientDB. Além do armazenamento dos dados de proveniência, este módulo é capaz de criar os objetos de cada banco de dados NoSQL. O uso dos três bancos de dados é opcional sendo possível utilizar apenas um banco de dados.

Além das famílias utilizadas nesta dissertação, (CORBELLINI *et al.*, 2017) descreve também a família chave-valor. Todavia, relata que os bancos de dados chave-valor não fazem suposições sobre os valores das chaves. Assim, devido à dificuldade de representar relacionamentos nos bancos de dados dessa categoria, ela não foi considerada no escopo desta dissertação.

Para a comunicação deste módulo como os sistemas de banco de dados NoSQL foram utilizadas as bibliotecas PyOrient[2], PyMongo[3] e o Python Cassandra Driver[4]. Essas bibliotecas do Python possibilitam o acesso do serviço de proveniência aos bancos de dados para a criação dos objetos de banco de dados e manipulação dos dados. A Figura 3.13 mostra um exemplo do código utilizado para criar os objetos do banco de dados de grafo no OrientDB.

3.3.4 Módulo de Consulta e Geração de Grafo de Proveniência

Este módulo recebe requisições do módulo de persistência e das aplicações clientes para gerar o grafo de proveniência no formato do PROV-DM. Para a implantação dessa

[2]https://orientdb.com/docs/last/PyOrient.html
[3]https://api.mongodb.com/python/current
[4]https://datastax.github.io/python-driver

funcionalidade foi utilizada a API Java Prefuse [5]. Outra funcionalidade disponível neste módulo são as consultas que podem ser realizadas nos bancos de dados de proveniência. Para isso, o módulo de consulta envia a consulta para o módulo de persistência e após receber o retorno envia o resultado da consulta para a aplicação cliente ou usuários.

```
#!/usr/bin/env python
# coding=utf-8

# Module Imports
from  pyorient.ogm import Graph, Config
import pyorient

#Create connection
client = pyorient.OrientDB("23.97.96.38", 2424)
session_id = client.connect( user, senha )

#Create a database
client.db_create( db_name, pyorient.DB_TYPE_GRAPH, pyorient.STORAGE_TYPE_MEMORY )

#open database
client.db_open( db_name, user, senha )

#Create Class
cluster_id = client.command( "create class Projeto extends V" )
cluster_id = client.command( "create class Experimento extends V" )
cluster_id = client.command( "create class Atividade extends V" )
cluster_id = client.command( "create class Arquivo extends V" )
cluster_id = client.command( "create class Agente extends V" )
cluster_id = client.command( "create class Clusters extends V" )
cluster_id = client.command( "create class Maquina extends V" )
cluster_id = client.command( "create class Provedor extends V" )
cluster_id = client.command( "create class Localizacao extends V" )
cluster_id = client.command( "create class Provisionamento extends V" )

#Create Property
cluster_id = client.command( "create property Projeto.IdProjeto Integer" )
cluster_id = client.command( "create property Projeto.Nome String" )
cluster_id = client.command( "create property Projeto.Descricao String" )
cluster_id = client.command( "create property Projeto.Coordenador String" )
```

Figura 3.13: Criação do Esquema de Dados Baseado em Grafos Utilizando a Linguagem Python.

3.4 Plataforma de Bioinformática

Para executar os experimentos e gerenciar seus dados de proveniência, foi implementada uma plataforma que disponibiliza os *clusters* dos sistemas de banco de dados NoSQL utilizados, incluindo seus esquemas de dados necessários para armazenar a proveniência. A plataforma também possui uma máquina contendo as ferramentas utilizadas pelos *workflows* de Bioinformática que serão utilizados por este trabalho e o serviço de gerenciamento de dados de proveniência em nuvem federada. De maneira geral, o ambiente utiliza quatro máquinas, conforme demonstrado na Figura 3.14.

[5] http://prefuse.org

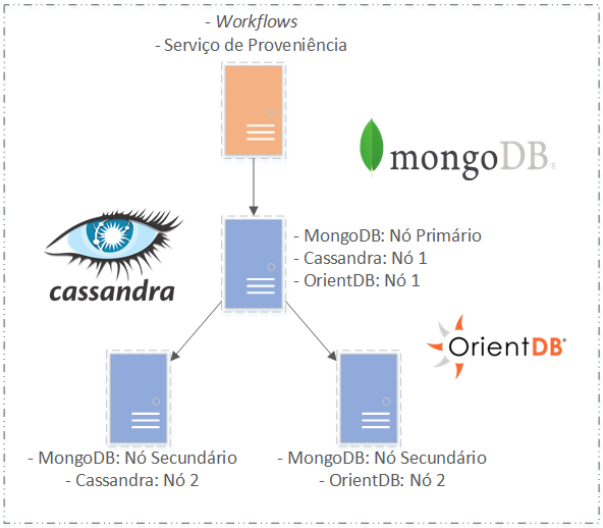


Figura 3.14: Plataforma de Bioinformática - Visão Geral.

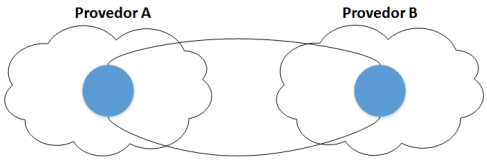
A quantidade de máquinas sugeridas pode ser alterada conforme a necessidade do usuário. A sugestão de se utilizar quatro máquinas, leva em consideração a configuração dos *clusters* de banco de dados com foco em alta disponibilidade e utilização de federação de nuvens. Nesta configuração, os dados estão replicados em dois nós dos sistemas de banco de dados NoSQL.

O MongoDB, na configuração do seu *cluster*, utiliza o conceito de nós primário e secundário a Figura 3.15c apresenta o nó onde ocorre a comunicação das aplicações com o banco de dados (nó primário) e os nós para os quais os dados são replicados (nós secundários). Assim, para que o dado seja replicado em dois nós são necessários três nós provisionados no *cluster* do MongoDB. Nesta configuração, o nó primário pode ser alocado no provedor de nuvem "A", e os nós secundários nos provedores de nuvem "B" e "C", executando em nuvem federada.

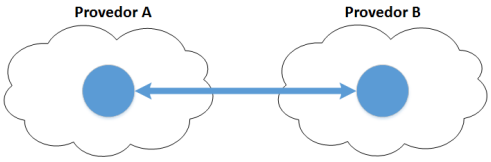
O Cassandra possui uma topologia de rede composta de um *cluster* de nós. Sua arquitetura de banco de dados é chamada de anel, onde os nós estão conectados e em execução em diferentes servidores. A Figura 3.15a apresenta o *cluster* do banco de dados Cassandra, em um ambiente de federação de nuvens, onde um nó do *cluster* pode ser hospedado em uma nuvem do provedor "A" e o outro nó pode ser hospedado em uma nuvem do provedor "B".

O *cluster* do OrientDB, nesta plataforma, é configurado com dois nós. De maneira similar ao Cassandra um nó do *cluster* pode ser hospedado em uma nuvem do provedor "A" e o outro nó pode ser hospedado em uma nuvem do provedor "B" (Figura 3.15b).

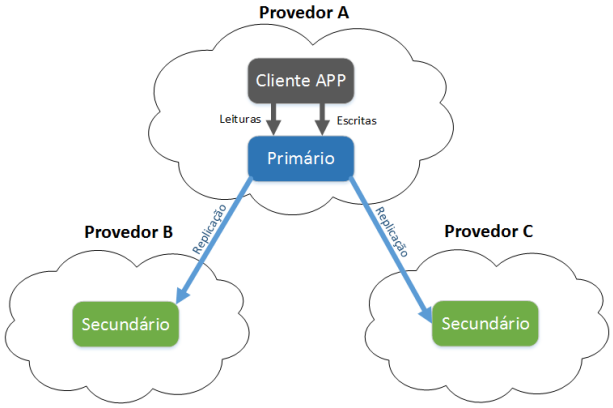
A instalação e configuração das ferramentas e dos sistemas de banco de dados NoSQL é feita por meio de imagens Docker que podem ser construídas em qualquer sistema operacional no qual o Docker esteja instalado. Os *hosts* são provisionados em nuvem



(a) *Cluster* do Cassandra em Nuvem Federada.



(b) *Cluster* do OrientDB em Nuvem Federada.



(c) *Cluster* do MongoDB em Nuvem Federada.

Figura 3.15: *Clusters* dos Sistemas de Banco de Dados NoSQL Disponibilizados pela Plataforma Proposta.

federada a partir das imagens utilizando diferentes sistemas operacionais.

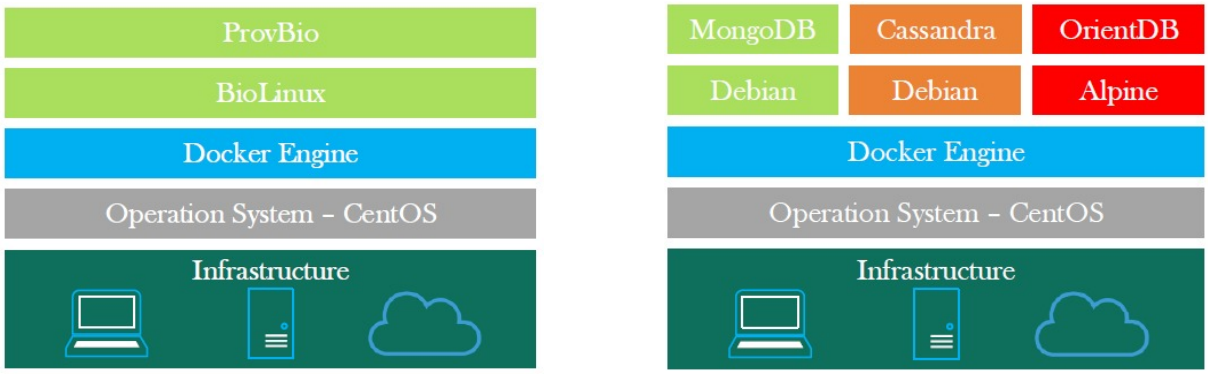
Um contêiner Docker encapsula completamente um aplicativo de software, fornecendo todas as dependências necessárias e garantindo as mesmas bibliotecas e pacotes de software durante a criação e execução de processos. Para provisionar as ferramentas necessárias para a execução dos *workflows*, foi construída uma imagem chamada ProvBio (Figura 3.17a) que deriva da imagem BioLinux (NETWORK, 2018) e acrescenta as ferramentas e configurações utilizadas pelos *workflows*. As imagens podem ser construídas a partir de um arquivo chamado *Dockerfile*, a Figura 3.16 mostra o arquivo *Dockerfile* contendo as configurações necessárias para a imagem ProvBio.

Os bancos de dados foram provisionados utilizando as imagens oficiais dos seus fabricantes (Docker Hub, 2019). A Figura 3.17b apresenta os diferentes sistemas operacionais (Debian e Alpine) utilizados pelos bancos de dados nos contêineres Docker, executando em uma máquina virtual que contém o Docker *engine* instalado no sistema operacional CentOS. O provisionamento da plataforma pode ser reproduzido seguindo as etapas disponíveis no github (WERCELENS, 2019).

A comunicação entre as máquinas virtuais hospedadas nas diferentes nuvens foi feita por meio de uma rede de sobreposição chamada *Overlay Network* utilizando o Docker *Swarm*. A rede de sobreposição (10.0.0.0/24) fornece comunicação entre os contêineres executados em máquinas hospedadas em diferentes provedores, conforme apresentado na

```
FROM gawbul/docker-bio-linux8
RUN sudo apt-get update && sudo apt-get install python-numpy && sudo apt-get install
COPY sratoolkit.2.9.0-ubuntu64.tar.gz /home/biolinux/sratoolkit.2.9.0-ubuntu64.tar.g
RUN sudo tar -xzf sratoolkit.2.9.0-ubuntu64.tar.gz && sudo ln -s /home/biolinux/srat
COPY sickle.zip /home/biolinux/sickle.zip
RUN sudo unzip sickle.zip && sudo rm -rf sickle.zip && cd sickle-master && sudo make
COPY hisat2-2.1.0-Linux_x86_64.zip /home/biolinux/hisat2-2.1.0-Linux_x86_64.zip
RUN sudo unzip hisat2-2.1.0-Linux_x86_64.zip && sudo rm -rf hisat2-2.1.0-Linux_x86_64
COPY samtools-1.7.tar.bz2 /home/biolinux/samtools-1.7.tar.bz2
RUN sudo tar -xvf samtools-1.7.tar.bz2 && sudo rm -rf samtools-1.7.tar.bz2 && cd samt
RUN sudo mkdir /opt/provbio/ && sudo mkdir /opt/provbio/workflowFiles
RUN sudo apt-get update
RUN sudo apt-get install -y python-pip && sudo pip install pyorient && sudo pip insta

```

Figura 3.16: Configuração da Imagem Docker ProvBio.



(a) Imagem ProvBio. (b) Imagens dos Bancos de Dados.

Figura 3.17: Imagens Docker Utilizadas na Plataforma.

Figura 3.18. Neste exemplo, a máquina ProvBio1 está hospedada no provedor de nuvem Google Cloud e a máquina ProvBio2 está hospedada no provedor de nuvem DigitalOcean. Os pacotes são tratados de forma transparente e roteados para a máquina correta e o contêiner de destino. Essa rede é usada pela plataforma, permitindo o provisionamento do *cluster* em um ambiente de nuvem federada. Para isso, também é necessário liberar as portas utilizadas pelos bancos de dados para acesso por meio da internet ou rotear o tráfego para outra porta que a aplicação consiga fazer requisições aos bancos de dados.

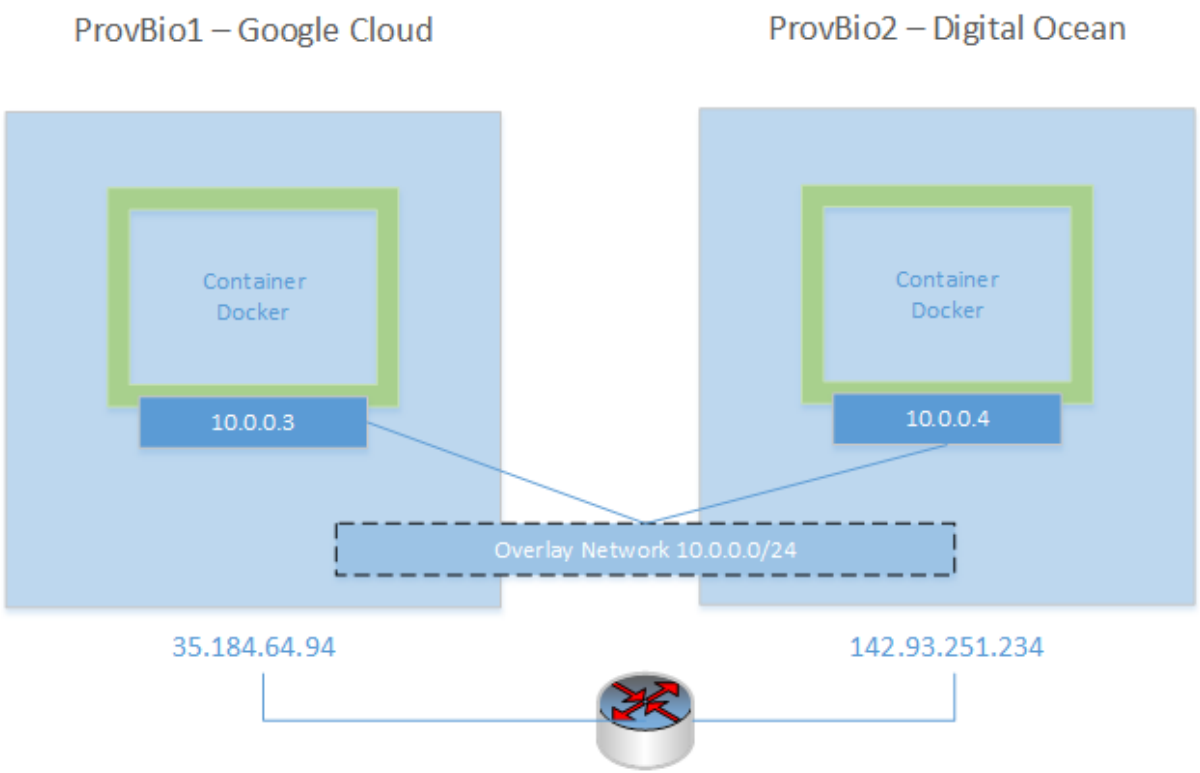


Figura 3.18: Rede de sobreposição - Docker Swarm.

Capítulo 4

Validação e Análise dos Resultados

Este capítulo apresenta os estudos de casos aplicados para a validação da arquitetura e esquemas de dados propostos no capítulo anterior. Para isto, este capítulo está divido da seguinte forma: a Seção 4.1 apresenta a caracterização do cenário, contendo uma explanação sobre *workflows* executados, os provedores de nuvem utilizados e o ambiente de nuvem federada. A Seção 4.2 mostra o resultado das execuções dos *workflows*. A Seção 4.3 faz uma análise da utilização da arquitetura proposta e os esquemas de dados dos sistemas de banco de dados NoSQL, bem como uma análise comparativa entre eles. Por fim, a Seção 4.4 apresenta os resultados em publicações.

4.1 Caracterização do Cenário

Para possibilitar a execução dos *workflows* e gerenciamento dos dados de proveniência nos provedores de nuvem distintos foi realizado um teste de provisionamento da plataforma por meio de imagens docker. Seguindo o tutorial disponível no github (WERCELENS, 2019), foram criados os contêineres nos provedores de nuvem pública Google Cloud, DigitalOcean, Microsoft Azure e no provedor de nuvem privada CloudJus. Dessa forma, foi possível confirmar que a plataforma pode ser provisionada em um ou vários provedores distintos simultaneamente.

Com intuito de validar os esquemas de dados de proveniência de *workflows* de Bioinformática para nuvem federada e a arquitetura proposta, foram utilizados três *workflows* de Bioinformática e três provedores de nuvem públicos. Neste cenário, os dados de proveniência foram armazenados nos bancos de dados e os dados brutos no sistema de arquivos da máquina virtual.

A fim de demonstrar a possibilidade de armazenar os dados brutos nos bancos de dados, foi realizado um teste de inserção e extração em cada sistema de banco de dados NoSQL coletando o tempo despendido para cada tipo de transação nos bancos de dados.

Para demonstrar a aderência ao modelo de dados PROV-DM foram gerados grafos de proveniência das fases dos *workflows* de Bioinformática. Além do grafo de proveniência, foram realizadas consultas aos bancos de dados elaboradas com o objetivo de responder perguntas feitas por biólogos do Departamento de Biologia Celular da Universidade de Brasília.

4.1.1 *Workflows* Executados

Os *workflows* de Bioinformática, utilizam como dados de entradas as chamadas *reads*, que são cadeias de caracteres em um alfabeto limitado (ACGT) representando fragmentos de DNA. Alinhamentos específicos dessas leituras dão origem a um conjunto produzindo sequências contíguas que representam amostras do DNA original (ZERBINO; BIRNEY, 2008). Um *workflow* típico de montagem do genoma é composto pelas fases sequenciais de filtragem, montagem e análise, que inclui a anotação (HAAS *et al.*, 2013).

A fase de filtragem limpa as leituras usando parâmetros de qualidade específicos (GUO *et al.*, 2013). A fase de montagem pode ser uma montagem baseada em referência ou *de novo*. O conjunto de fragmentos baseado em referência utiliza um genoma de referência, alinhando as leituras do organismo alvo com seu genoma ou um genoma de um organismo filogeneticamente próximo. A chamada montagem *de novo* é executada sem um genoma de referência (BLEIDORN, 2017). A fase de análise é muito diversificada e depende da resposta biológica buscada no experimento, e os dados são processados para validar a hipótese inicial do experimento (GUO *et al.*, 2013).

Neste trabalho foram utilizados três *workflows* de Bioinformática:

- *Workflow* 1 - Pipeline didático para análise quantitativa de RNA-seq de células endoteliais humanas: este *workflow* consiste em mapear leituras de DNA expresso (cDNA) de células endoteliais microvasculares primárias cardíacas humanas e células endoteliais endocárdicas derivadas de hPSC para o cromossomo 22 humano como referência (Apêndice A). Ele é composto pelas fases de filtragem, mapeamento e análise. O dado de entrada é um arquivo no formato FASTQ de aproximadamente 7 GB. O pipeline foi montado pelo Departamento de Biologia Celular da Universidade de Brasília. As ferramentas utilizadas por este *workflow* são: Sickle (JOSHI; FASS *et al.*, 2011), Samtools (LI *et al.*, 2009), Hisat (KIM; LANGMEAD; SALZBERG, 2015) e o HTSeq (ANDERS; PYL; HUBER, 2015);

- *Workflow* 2 - Pipeline didático para montagem *de novo* do genoma da bactéria *Enterobacter kobei*, uma espécie multirresistente a drogas: consiste em uma montagem de genoma *de novo* de um isolado *Enterobacter kobei* (Apêndice B). O DNA do *E. kobei* foi sequenciado usando a plataforma HiSeq (Illumina) gerando leituras de 100bp

paired-end reads (JUDGE *et al.*, 2016). O pipeline foi montado pelo Departamento de Biologia Celular da Universidade de Brasília. As ferramentas utilizadas são o Trimmomatic (BOLGER; LOHSE; USADEL, 2014) para filtragem, Abyss (SIMPSON *et al.*, 2009) para a montagem e Quast (GUREVICH *et al.*, 2013) para a análise estatística dos resultados. Os dados de entrada são dois arquivos no formato FASTQ.GZ de aproximadamente 190MB cada;

- *Workflow* 3 - RNA-Seq do fungo *Aspergillus fumigatus*: este experimento é descrito por (LATGÉ, 1999) e trata sobre uma análise de RNA-Seq do fungo *Aspergillus fumigatus*, com foco na sua biologia e nas doenças que ele causa. Este *workflow* é composto pelas fases de filtragem, montagem e análise. Os dados de entrada da fase de filtragem são seis arquivos no formato FASTQ com tamanho médio de 1.33 GB cada, totalizando aproximadamente 8 GB.

4.1.2 Ambiente Computacional

A execução dos experimentos ocorreu em dois ambientes: nuvem computacional e nuvem federada. O escopo deste trabalho trata de ambiente de nuvem federada, mas optou-se por executar também em nuvem computacional para demonstrar que a solução pode ser utilizada em ambos.

O ambiente de nuvem federada utilizou três provedores de nuvem pública Microsoft Azure, DigitalOcean e Google Cloud. As configurações das instâncias foram similares, conforme mostra a Tabela 4.1. O ambiente de nuvem computacional utilizou dois provedores de nuvem pública DigitalOcean e Google Cloud, e um provedor de nuvem privada CloudJus. As instâncias de máquina virtual também são similares (Tabela 4.2).

Neste trabalho, o uso de sistemas operacionais heterogêneos foi intencional, para verificar a compatibilidade da arquitetura proposta com um ambiente de nuvem federada. Uma característica relevante da arquitetura proposta é fornecer um ambiente no qual a complexidade da instalação e configuração do ambiente seja reduzida.

Conforme descrito nas seções anteriores, são utilizados três sistemas de banco de dados NoSQL. O objetivo dessa diversificação de sistemas de banco de dados NoSQL é fornecer mais opções de banco de dados para que os sistemas de gerenciamento de *workflows*, os aplicativos de gerenciamento de dados de proveniência e os pesquisadores possam escolher livremente de acordo com sua preferência. A adaptabilidade de cada um desses tipos de sistemas de banco de dados em relação ao PROV-DM, já foi apresentada no trabalho de HONDO *et al.* (2017).

Seguindo o tutorial disponível no github (WERCELENS, 2019), foram provisionadas quatro instâncias de máquina virtual nos provedores de nuvem utilizados por cada ambi-

Tabela 4.1: Configuração do Ambiente de Nuvem Federada.

Provedor	**Hostname**	**Sistema Operacional**	**CPU**	**RAM**	**Disco**
Azure	ProvBio1	Ubuntu 14	2	4GB	80GB
Azure	ProvBio2	CentOS 7	2	4GB	80GB
DigitalOcean	ProvBio3	CentOS 7	2	4GB	80GB
GoogleCloud	ProvBio4	CentOS 7	2	4GB	80GB

ente. As máquinas virtuais usam o Docker para executar contêineres a partir das imagens utilizadas pela plataforma, onde se manteve os *workflows* e os sistemas de banco de dados NoSQL em contêineres separados. Cada contêiner executa seu próprio sistema operacional e configurações específicas para cada ferramenta. Os sistemas de banco de dados NoSQL Cassandra e MongoDB executam no sistema operacional Linux Debian, o sistema de banco de dados NoSQL OrientDB executa no sistema operacional Linux Alpine e os *workflows* executam no sistema operacional BioLinux. As instâncias de máquina virtual que hospedam os contêineres de banco de dados (ProvBio2, ProvBio3 e ProvBio4) utilizam o sistema operacional Linux CentOS 7 e a instância que executa os *workflows* (ProvBio1) utiliza o sistema operacional Linux Ubuntu 14.

As versões dos sistemas de banco de dados NoSQL foram: Cassandra 3.11.4, MongoDB 2.6, OrientDB 3.0.19. Os dados de proveniência foram armazenados de forma distribuída com duas réplicas dos dados nos nós do *Cluster* para os três bancos de dados.

4.2 Resultados das Execuções dos Workflows

As execuções dos *workflows* foram realizadas por meio do serviço de gerenciamento de dados de proveniência construído na linguagem Python. As chamadas do serviço foram feitas na máquina ProvBio1 (Tabelas 4.1 e 4.2) que possui o papel de executar os *workflows* conforme mostra a Tabela 4.3. O serviço capturou e armazenou os dados de proveniência para cada banco de dados provisionado nos NoSQL hospedados nas máquinas ProvBio2, ProvBio3 e ProvBio4 (Tabelas 4.1 e 4.2). A Tabela 4.3 apresenta os nós dos bancos de dados configurados em cada máquina e os contêineres utilizados para cada NoSQL.

Os tempos de execução de cada atividade foram contabilizados e armazenados. Para gerar o grafo de proveniência aderente ao modelo PROV-DM, foi utilizada a API Java Prefuse [1]. Além disso, o NoSQL OrientDB possui uma ferramenta gráfica capaz de gerar gráficos.

[1] http://prefuse.org

Tabela 4.2: Configuração do Ambiente de Nuvem Computacional.

Provedor	**Hostname**	**Sistema Operacional**	**CPU**	**RAM**	**Disco**
GoogleCloud DigitalOcean CloudJus Microsoft Azure	ProvBio1	Ubuntu 14	2	4GB	80GB
GoogleCloud DigitalOcean CloudJus Microsoft Azure	ProvBio2	CentOS 7	2	4GB	80GB
GoogleCloud DigitalOcean CloudJus Microsoft Azure	ProvBio3	CentOS 7	2	4GB	80GB
GoogleCloud DigitalOcean CloudJus Microsoft Azure	ProvBio4	CentOS 7	2	4GB	80GB

Tabela 4.3: Configurações dos Contêineres Docker.

Hostname	**Papel**	**Contêiner Docker**
ProvBio1	Execução dos *Workflows*	ProvBio (derivado do Biolinux)
ProvBio2	Cassandra Nó 1 OrientDB Nó 1 MongoDB Nó Secundário 1	Cassandra e MongoDB (derivado do Debian Linux) OrientDB (derivado do Alpine Linux)
ProvBio3	Cassandra Nó 2 OrientDB Nó 2 MongoDB Nó Secundário 2	Cassandra e MongoDB (derivado do Debian Linux) OrientDB (derivado do Alpine Linux)
ProvBio4	MongoDB Nó Primário	MongoDB (derivado do Debian Linux)

4.2.1 Execução do *Workflow* 1

O *workflow* 1 é um pipeline didático para análise quantitativa de RNA-seq de células endoteliais humanas e possui as fases de filtragem, mapeamento e análise. O arquivo de entrada utilizado é um FASTQ de 7 GB. Conforme detalhado na seção anterior, as configurações dos ambientes de nuvem computacional e nuvem federada foram as mesmas com intuito de garantir a isonomia dos testes. Foram realizadas quatro execuções do *workflow* 1 para cada provedor de nuvem individualmente e quatro execuções no ambiente de nuvem federada.

No ambiente de nuvem computacional configurado conforme a Tabela 4.2, o *workflow* 1 foi executado nos provedores de nuvem públicos (DigitalOcean, Google Cloud e Microsoft Azure) individualmente. O tempo médio de execução incluindo o gerenciamento de dados de proveniência foi de 27m07s, 19m30s e 37m21s para a DigitalOcean, GoogleCloud e Microsoft Azure respectivamente. No provedor de nuvem privada CloudJus o tempo foi de 20m24s.

No ambiente de nuvem federada configurado conforme a Tabela 4.1, o tempo total de execução já incluindo o tempo de armazenamento dos dados de proveniência foi de 46m55s. O comparativo dos tempos das quatro execuções em minutos para cada provedor de nuvem computacional e das quatro execuções no ambiente de nuvem federada é apresentado na Figura 4.2.

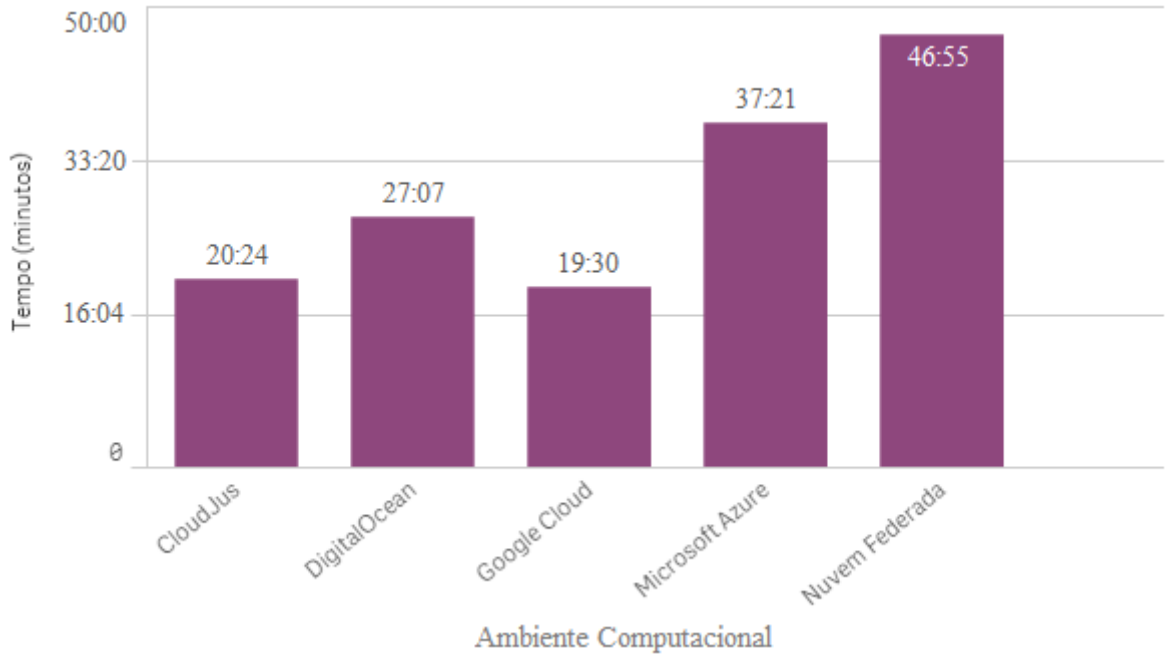


Figura 4.1: Média dos tempos (minutos) de execução e gerenciamento de dados de proveniência do *workflow* 1.

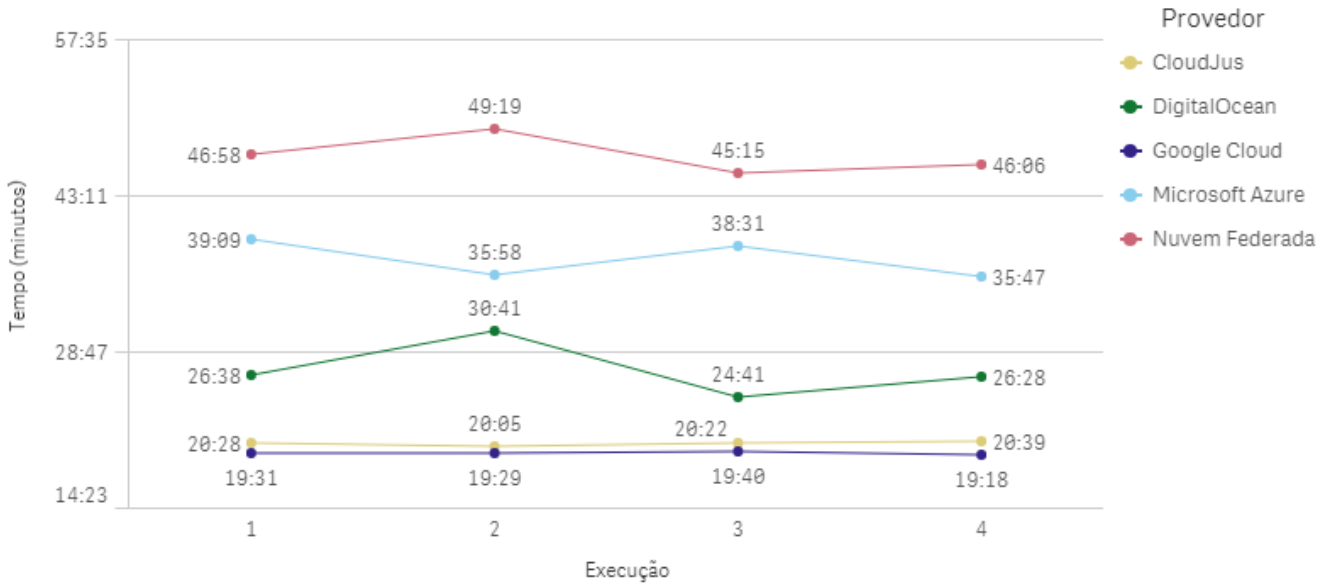


Figura 4.2: Tempos (minutos) de execução e gerenciamento de dados de proveniência do *workflow* 1.

A partir dos dados armazenados nos bancos de dados, foram gerados os grafos de proveniência. A Figura 4.3 apresenta o grafo de proveniência gerado para a fase de mapeamento deste *workflow*, onde o agente executor (Ag_Polyane) aparece em cinza como responsável pela execução da atividade At_Mat_Hisat2 em amarelo, que por sua vez utilizou os arquivos em azul SRR5181508_FILTERED.fastq e chromosome.22.hisat2.idx e gerou o arquivo SRR5181508.sam.

4.2.2 Execução do *Workflow* 2

O *workflow* 2 é um pipeline didático para montagem *de novo* do genoma da bactéria *Enterobacter kobei*, que é uma espécie multirresistente a drogas. O pipeline é composto pelas fases de filtragem, montagem e análise. Para a execução do *workflow* 2 também foram utilizadas as mesmas configurações dos ambientes de nuvem computacional e nuvem federada (Tabela 4.1 e Tabela 4.2). As execuções foram repetidas quatro vezes para cada provedor de nuvem computacional e para o ambiente de nuvem federada. Os tempos de execução contemplam também o tempo despendido para gerenciar os dados de proveniência.

A Figura 4.4 apresenta o tempo médio de execução em minutos no ambiente de nuvem computacional. Para os provedores de nuvem pública DigitalOcean, Google Cloud e Microsoft Azure, o tempo médio foi de 33m34s, 25m24s e 42m48s respectivamente, e no provedor de nuvem privada CloudJus o tempo médio foi de 35m14s.

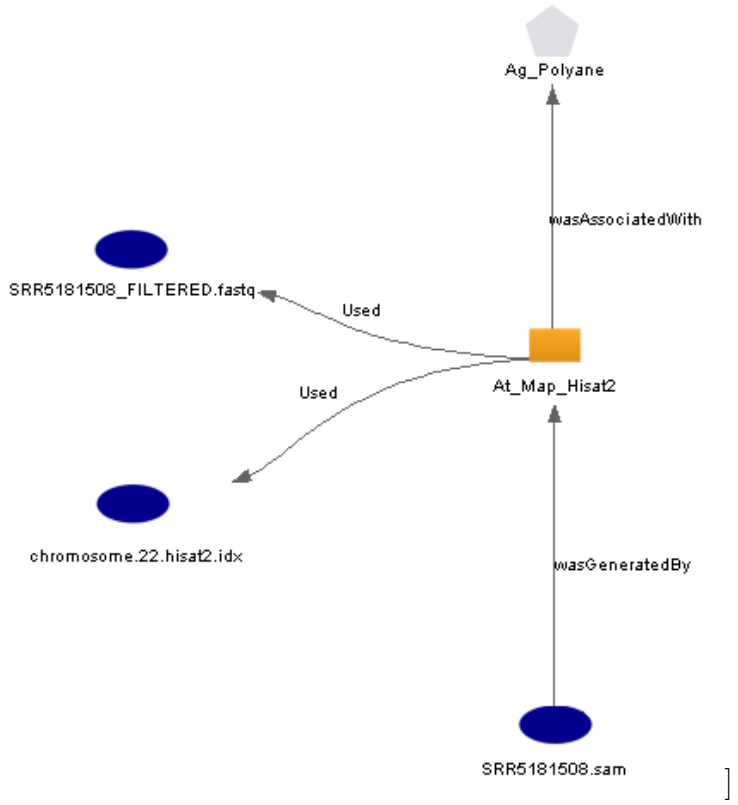


Figura 4.3: Grafo de Proveniência da Fase de Mapeamento do *Workflow* 1.

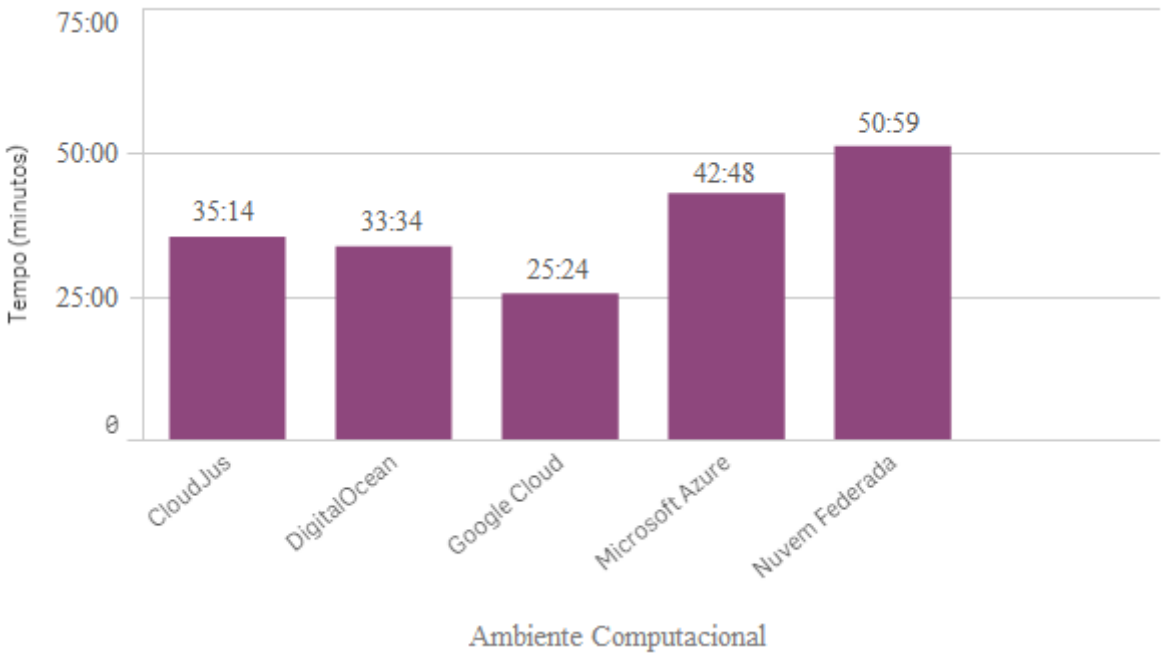


Figura 4.4: Média dos tempos (minutos) de execução e gerenciamento de dados de proveniência do *workflow* 2.

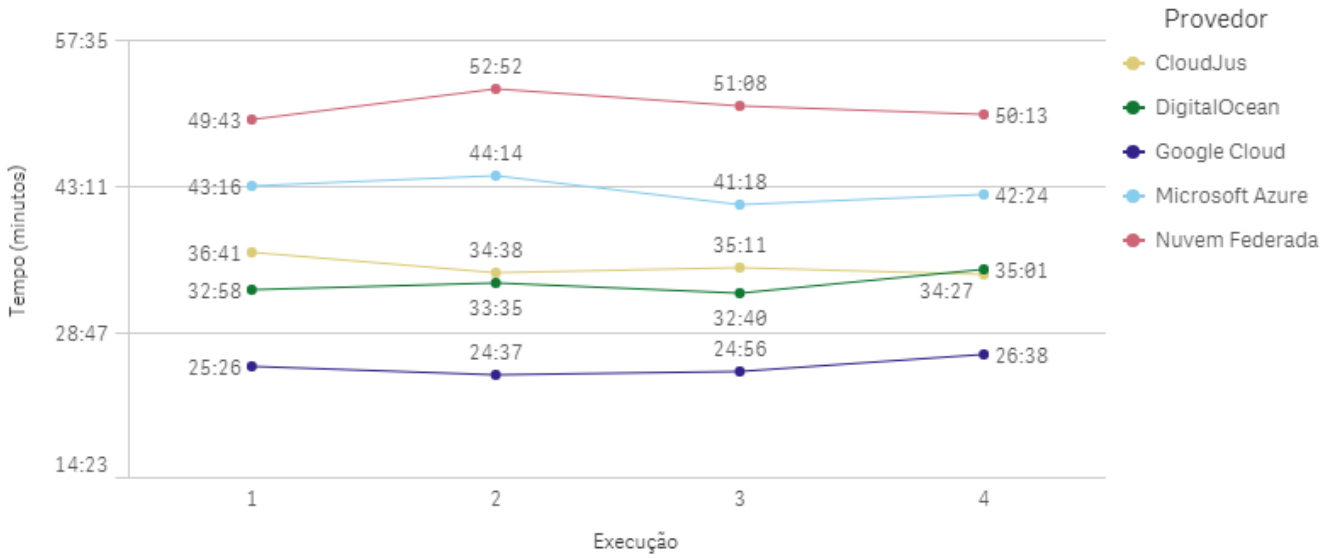


Figura 4.5: Tempos (minutos) de execução e gerenciamento de dados de proveniência do *workflow* 2.

No ambiente de nuvem federada o tempo médio das execuções já incluindo o tempo de gerenciamento dos dados de proveniência foi de 50m59. A Figura 4.5 mostra o tempo de cada execução para os ambientes de nuvem computacional e nuvem federada em minutos.

Após a execução do *workflow* 2 foi gerado o grafo de proveniência da fase de filtragem. A Figura 4.6 mostra a atividade de filtragem At_Filt_Trimmomatic em amarelo que foi executada pelo agente Ag_Polyane em cinza. Os dados de entrada da atividade foram os arquivos ERR885455_1.fastq e ERR885455_2.fastq, e os dados de saída foram os arquivos ERR885455_1_FILTERED.fastq, ERR885455_2_FILTERED.fastq, ERR885455_1_UNPAIRED.fastq e ERR88545_2_UNPAIRED.fastq.

4.2.3 Execução do Workflow 3

O *workflow* 3 é descrito por (LATGÉ, 1999) e trata sobre uma análise de RNA-Seq do fungo *Aspergillus fumigatus*. A execução do *workflow* 3 foi realizada no provedor de nuvem privada CloudJus. O tempo total de execução é de 23h46m11s. Considerando o longo tempo de execução e o fato de restarem poucos créditos oriundos dos *vouchers* disponíveis nos provedores de nuvem pública, optou-se por utilizar os dados brutos para verificar os tempos de inserção e extração em cada NoSQL.

Os dados brutos foram inseridos e extraídos quatro vezes para cada NoSQL e os tempos de cada arquivo por transação foram contabilizados. Conforme detalhado nas seções anteriores, o tamanho total dos seis arquivos de entrada da fase de filtragem deste *workflow* é de aproximadamente 8 GB. As configurações do ambiente de nuvem computacional utilizadas foram as mesmas (Tabela 4.2) com intuito de garantir a isonomia dos testes.

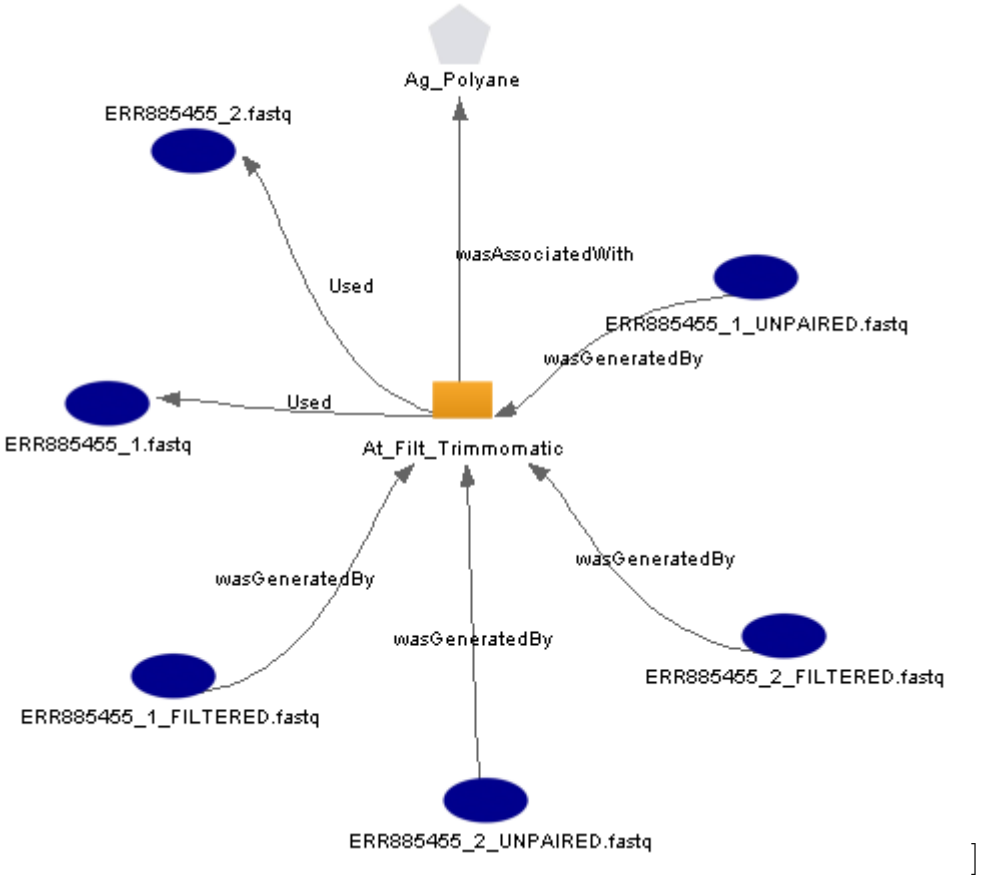


Figura 4.6: Grafo de Proveniência da Fase de Filtragem do *Workflow* 2.

A Figura 4.7 apresenta os tempos de inserção (verde) e extração (vermelho) dos dados brutos. O tempo de inserção dos três sistemas de bancos de dados NoSQL foi mais lento quando comparados aos seus respectivos tempos de extração dos arquivos, ou seja, de maneira geral os três bancos de dados possuem um melhor desempenho na extração comparado com a inserção.

O MongoDB apresentou um desempenho melhor com relação aos tempos de inserção e extração, esses tempos variaram em torno de 20% entre eles, desconsiderando a margem de erro. Ele destacou-se no desempenho de inserção apresentando sempre o melhor tempo para todos os arquivos, o tempo médio foi de 120s considerando a margem de erro de 10s para mais ou para menos. O OrientDB e o MongoDB apresentaram um desempenho similar com relação à extração. O desempenho menos satisfatório foi o do Cassandra, apresentando um tempo médio de 339s de inserção com margem de erro de 61s para mais ou para menos e um tempo médio de 178s de extração com margem de erro de 11s para mais ou para menos.

Após a realização do teste de desempenho de armazenamento de dados brutos, foi coletada a proveniência do *workflow* completo. A Figura 4.8 mostra o grafo de proveniência

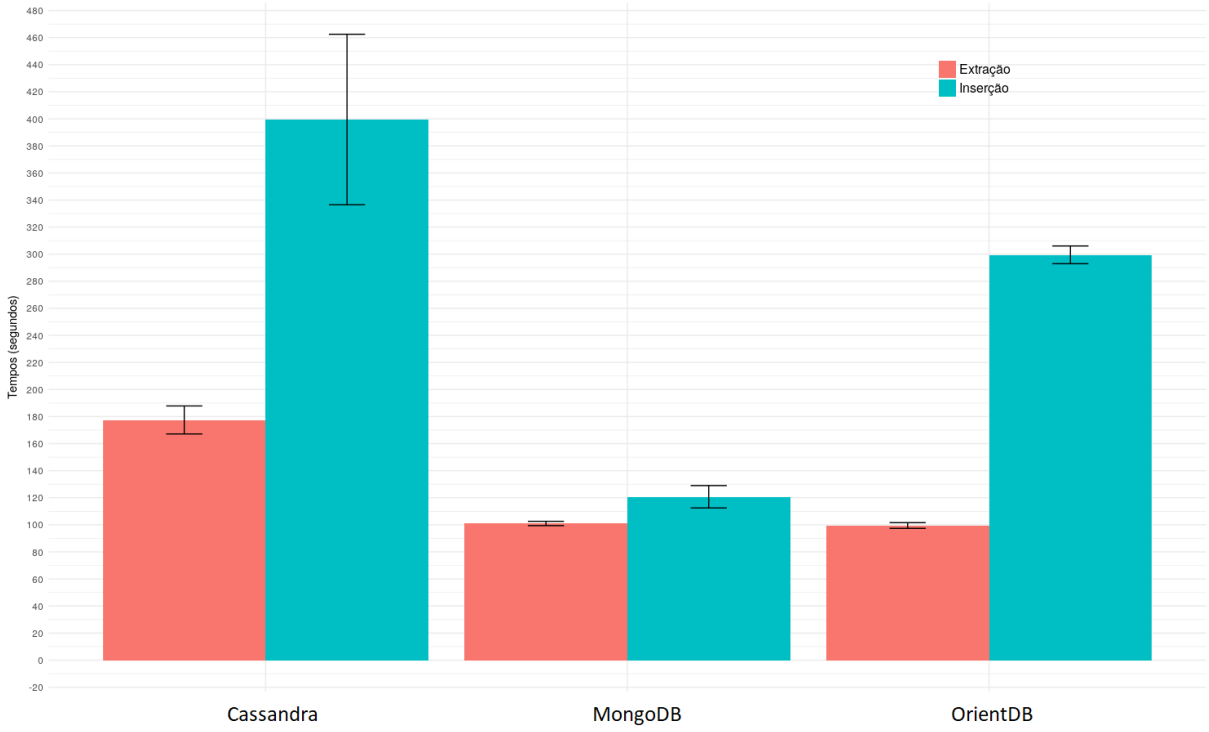


Figura 4.7: Média dos tempos (segundos) de inserção e extração dos dados brutos do *workflow*.

gerado para a fase de filtragem, onde é possível verificar o agente executor (Ag_Polyane) representado na cor cinza, a atividade de filtragem em amarelo, os arquivos de entrada e saída de cada atividade em azul e seus relacionamentos.

4.3 Considerações sobre os Resultados

Para os testes nos ambientes de nuvem computacional e de nuvem federada que foram utilizados pela plataforma e pelo serviço de gerenciamento dos dados de proveniência contendo as mesmas configurações. O uso do Docker facilitou o provisionamento e portabilidade das máquinas, além de garantir a isonomia dos testes com relação aos recursos alocados, ferramentas necessárias e suas configurações.

Para validar os esquemas de dados de proveniência, além de gerar o grafo no formato PROV-DM, foram realizadas consultas aos bancos de dados a partir de perguntas elaboradas por um pesquisador do Departamento de Biologia Celular da Universidade de Brasília. Cada banco de dados tem sua própria linguagem de consulta e, em alguns casos, foi necessário criar funções especiais de agregação.

O modelo de dados do Cassandra não suporta agregações, uma vez que seu objetivo é ser projetado para responder perguntas de negócio previamente conhecidas. Dessa

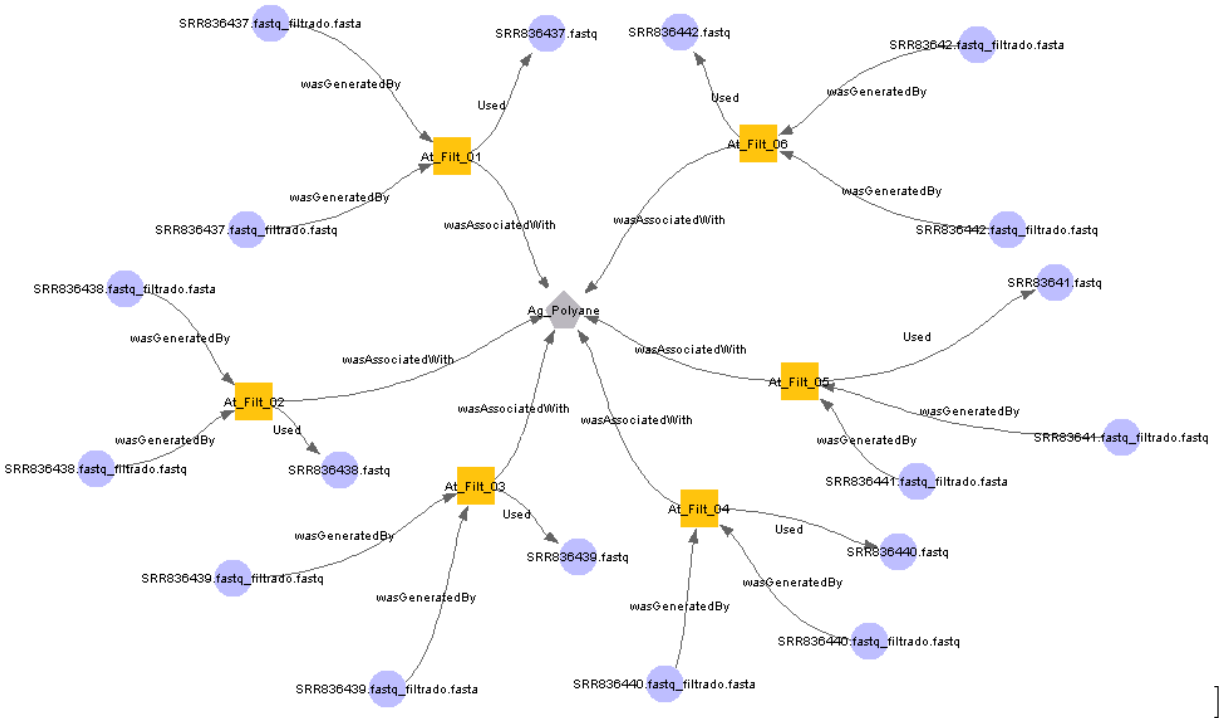


Figura 4.8: Grafo de Proveniência da Fase de Filtragem do *Workflow* 3.

forma, a própria aplicação armazena o dado agregado no banco de dados ou pode-se criar funções no sistema gerenciador de banco de dados Cassandra escritas na linguagem Java. O OrientDB possui algumas funções de agregação, mas também é possível criar funções em Java no próprio SGBD. O MongoDB possui suporte nativos para funções de agregação. A seguir, são apresentadas as consultas detalhadas.

4.3.1 Consulta 1: Ambiente Computacional

A primeira consulta foi sobre o ambiente de execução dos experimentos: "Quais as configurações das máquinas (provedor de nuvem, processadores, memória , etc.) utilizadas no ambiente de nuvem federada para a execução dos experimentos?". A Tabela 4.4 mostra as consultas realizadas para cada um dos bancos de dados.

O filtro utilizado para cada uma das consultas é a forma em que as máquinas foram provisionadas, ou seja, neste caso deve-se informar TipoProvisionamento = 'Provedor'. Esse atributo indica se as máquinas foram provisionadas pelo agente ou provedor de serviços. Com isso, é possível listar as configurações das máquinas onde os experimentos foram executados em ambiente de nuvem federada. Os resultados da Consulta 1 podem ser observados no Apêndice C. O OrientDB e o Cassandra apresentam os resultados no formato de linhas e colunas, o MongoDB retorna um arquivo no formato JSON [2].

[2]https://www.json.org/

Tabela 4.4: Consulta 1 - Ambiente Computacional.

NoSQL	**Consulta**
Cassandra	SELECT nomecluster, nomeprovedor, hostname, sistemaoperacional, tipoprovisionamento, cpu, ram,disco,tipodisco FROM provenance.expbioambientecomputacional WHERE tipoprovisionamento = 'Provedor';
OrientDB	SELECT cluster.nomecluster, provedor.nomeprovedor, maquina.hostname, maquina.sistemaoperacional, cluster.tipoprovisionamento, maquina.cpu, maquina.ram, maquina.disco, maquina.tipodisco FROM provenance.maquina WHERE cluster.tipoprovisionamento = 'Provedor';
MongoDB	db.cluster.find({"maquinas.tipoProvisionamento": "Provedor"}, { nomecluster: 1, "maquinas.hostname": 1, "maquinas.provedor.nome":1, "maquinas.tipoProvisionamento":1, "maquinas.sistemaOperacional":1, "maquinas.cpu": 1, "maquinas.ram": 1, "maquinas.disco": 1, "maquinas.tipoDisco": 1);}

Tabela 4.5: Consulta 2 - Atividades dos *Workflows*.

NoSQL	Consulta
Cassandra	SELECT nomeprograma, versaoprograma FROM provenance.expbioatividade WHERE idprojeto=1 and idexperimento=2;
OrientDB	SELECT atividade.nomeprograma, atividade.versaoprograma FROM provenance.atividade GROUP BY nomeprograma WHERE projeto.idprojeto=1 and experimento.idexperimento=2;
MongoDB	db.projeto.find({"experimento.idexperimento": 2, "idprojeto": 1}, {"experimento.atividade.nomeprograma": 1, "experimento.atividade.versaoprograma":1});

4.3.2 Consulta 2: Atividades dos *Workflows*

A segunda pergunta é relativa às atividades dos *workflows* executadas durante o experimento científico e as ferramentas utilizadas: "Quais os nomes e as versões dos programas utilizados para executar as atividades do *workflow* 2?".

A Tabela 4.5 apresenta as consultas escritas para cada um dos bancos de dados utilizando suas próprias linguagens de consulta. Os filtros utilizados são IdProjeto = 1 e o experimento solicitado na pergunta que é o IdExperimento=2. O retorno das consultas contém os nomes dos programas e versões que foram utilizadas na execução de cada fase do *workflow* 2 (Apêndice D).

4.3.3 Comparação entre os Ambientes de Nuvem Federada e Nuvem Computacional

O tempo de execução dos experimentos foram coletados para cada ambiente (nuvem federada e nuvem computacional). Não é o foco deste trabalho realizar análise comparativa de desempenho entre os diferentes provedores de nuvem. Todavia, é um dado armazenado na proveniência. A Figura 4.9 mostra o tempo médio de execução do *workflow* 1 e do *workflow* 2 no escopo de nuvem computacional e também quando existe uma federação. Executar experimentos em nuvem federada, em geral levou mais tempo que as demais nuvens, em média 2937s ou 49m35s.

É possível observar que em nuvem federada para o *workflow* 1 o tempo decorrido foi de 46m55s (Figura 4.1), menos satisfatório que a DigitalOcean, Google Cloud, Microsoft Azure e CloudJus, que executaram em aproximadamente 27m07s, 19m30s, 37m21 e 20m24s respectivamente. A Google Cloud apresentou o menor tempo quando comparada ao demais provedores de nuvem.

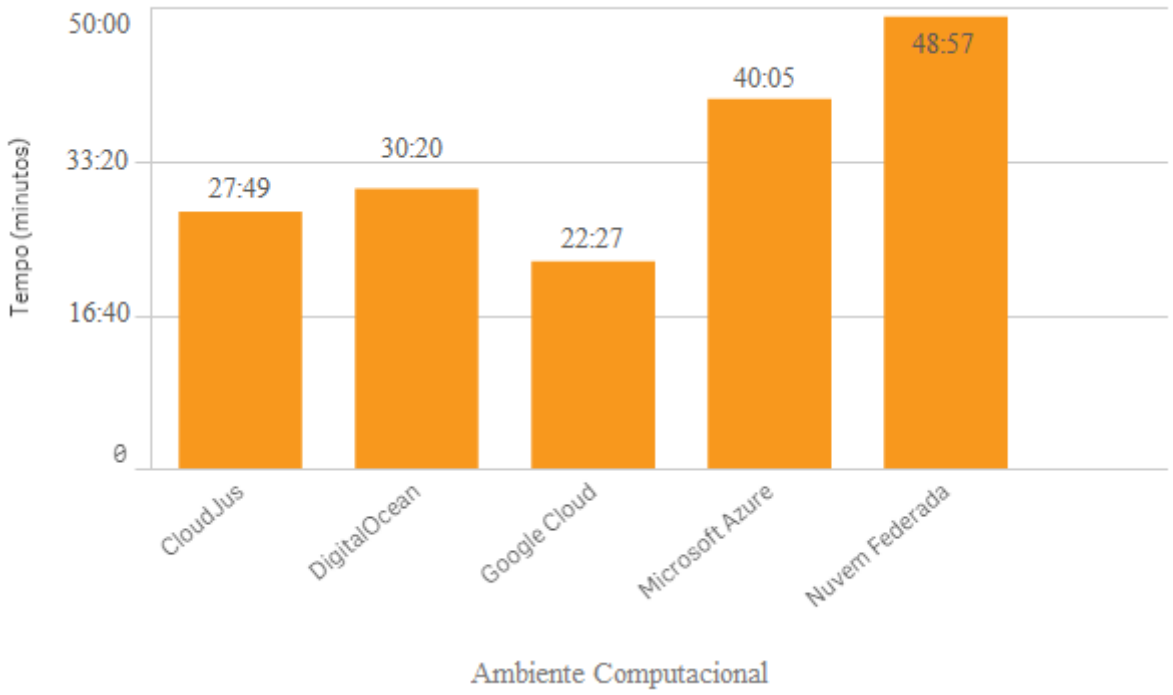


Figura 4.9: Média dos tempos (minutos) de execução e gerenciamento de dados de proveniência dos *workflows* 1 e 2.

Para o *workflow* 2 o mesmo ocorre, observa-se a execução em nuvem federada levou cerca de 50m59s. Todavia, mais uma vez a Google Cloud obtém o melhor tempo (aproximadamente 25m24s) e em segundo lugar a DigitalOcean (aproximadamente 33m34s).

Apesar do desempenho dos experimentos no ambiente de nuvem federada ser menos satisfatório, seu uso é viável e tem como benefício a disponibilidade da solução. Considerando que o uso de provedores distintos minimiza a dependência de um único provedor de nuvem.

Os dados apresentados nas Figura 4.1, Figura 4.4 e Figura 4.9 não consideram a inserção de dados brutos no banco de dados, haja vista que não foram armazenados em banco de dados durante a execução do *workflow* 1 e do *workflow* 2. O armazenamento de dados brutos foi testado com os arquivos de filtragem do *workflow* 3, conforme mostra Figura 4.7, onde o MongoDB destacou-se no desempenho de inserção apresentando sempre o melhor tempo para todos os arquivos e também demonstrou equilíbrio entre seus tempos médios de inserção e extração, variando cerca de 20% sem considerar a margem de erro.

4.4 Resultados em Publicações

Durante a pesquisa foram publicados três artigos completos e mais dois submetidos para publicação, todos baseados em gerenciamento de dados de proveniência em ambiente

de nuvem computacional, nuvem federada e/ou bancos de dados NoSQL.

Trabalhos publicados:

- Uso de bancos de dados NoSQL para gerenciamento de dados em *workflow* de Bioinformática - 32º Simpósio Brasileiro de Bancos de Dados (SBBD 2017);
- *Data Provenance Management for Bioinformatics Workflows using NoSQL Database Systems in a Cloud Computing Environment - IEEE International Conference on Bioinformatics and Biomedicine* (BIBM 2017);
- *Graph databases in Molecular Biology - 11th Brazilian Symposium on Bioinformatics* (BSB 2018).

Trabalhos aceitos:

- *Bioinformatics Workflows with NoSQL Database in Cloud Computing - Journal: Evolutionary Bioinformatics* - Aguardando publicação;
- *Platform for Data Provenance Management of Bioinformatics Workflows in Federated Clouds - IEEE International Conference on Bioinformatics and Biomedicine* (BIBM 2018) - Aceito, mas não foi publicado por retrições de recursos que impossibilitaram a apresentação do artigo.

Capítulo 5

Conclusões

Este trabalho teve como objetivo desenvolver uma solução que permita o gerenciamento de dados de proveniência de *workflows* de Bioinformática, armazenando de forma distribuída em esquemas de dados baseados no PROV-DM, utilizando sistemas de banco de dados NoSQL em um ambiente de nuvem federada.

O uso de nuvem federada oferece flexibilidade para o usuário, mas pode aumentar o trabalho de configuração do ambiente quando comparado a um ambiente de nuvem computacional. Com o objetivo de minimizar esse trabalho foi construída uma plataforma capaz de provisionar os bancos de dados e as ferramentas necessárias para a execução dos experimentos.

Configurar um ambiente de computação é uma tarefa que pode ser complexa, demorada e especializada em conhecimento técnico. Essa complexidade pode aumentar com a diversidade de nuvens computacionais existentes. Tudo isso torna mais difícil gerenciar os dados de origem. Neste estudo, foi explorado o uso de imagens e contêineres Docker para automatizar a configuração de ambientes e provisioná-los de maneira mais direta e eficiente. Seu uso foi fundamental, considerando a agilidade no provisionamento do ambiente nos diferentes provedores de nuvem, sendo necessário apenas executar o contêiner de cada imagem previamente definida. Sem o uso do Docker haveria um esforço maior na configuração do ambiente, visto que as ferramentas utilizadas pelos *workflows* e os sistemas bancos de dados NoSQL precisariam ser instalados e configurados em cada máquina virtual para cada provedor de nuvem.

A federação de nuvens suporta a ideia de cooperação entre provedores de nuvem e com isso menos dependência de um único provedor para os serviços hospedados, contribuindo para a disponibilidade, eficiência e eficácia dos serviços. A comunicação entre os nós do *cluster* hospedado por provedores de nuvem distintos foi feita usando uma rede de sobreposição, que permitia a comunicação entre os contêineres mesmo em redes diferentes. Os resultados mostraram que os ambientes podem ser provisionados com esforços técnicos

limitados, seguindo as etapas descritas tutorial disponível no Github.

Em relação às bases de dados, os resultados mostraram que foi possível armazenar os dados de forma distribuída em diferentes nuvens. Este trabalho disponibiliza três opções de tecnologias de banco de dados para armazenar a proveniência de dados usando o modelo de dados PROV-DM, incluindo o esquema de dados específico de cada banco de dados, que pode ser usado de acordo com a preferência do pesquisador ou integrado aos sistemas de gerenciamento de *workflows*.

A modelagem de dados definida neste trabalho, demonstrou ser adequado para o gerenciamento dos dados de proveniência em nuvem federada. Foi possível responder as questões elaboradas por pesquisadores, bem como gerar o grafo baseado no PROV-DM a partir dos dados armazenados em cada NoSQL. Este trabalho contribuiu para a reprodutibilidade de experimentos científicos em bioinformática.

As execuções dos experimentos em nuvem federada, nesta dissertação, em geral levaram mais tempo quando comparadas com as execuções em um ambiente provisionado em um único provedor de nuvem. Apesar do desempenho ser menos satisfatório, o uso de nuvem federada aprimora eficácia, considerando que em caso de falha em um dos nós do *cluster* em devido ao provedor de nuvem, por exemplo, os dados de proveniência estão replicados em outros provedores. Assim, o pesquisador tem confiabilidade de que seu experimento será executado por completo.

Como trabalho futuro tem-se: integrar com sistemas gerenciadores de *workflows* científicos, tal como BioNimbuz; implementar mecanismos de segurança para o *cluster*; explorar a disponibilidade do serviço de gerenciamento de proveniência de dados em um ambiente de nuvem federada.

Referências Bibliográficas

AFGAN, Enis *et al.* The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic acids research*, Oxford University Press, v. 44, n. W1, p. W3–W10, 2016. 25

ALMEIDA, Rodrigo Pinheiro de. *Gerenciamento de dados de proveniência de workflow de bioinformática com banco de dados baseados em grafo*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade de Brasília, Brasília, DF, Brasil, 2016. 10

ALMEIDA, Rodrigo Pinheiro de *et al.* Managing data provenance for bioinformatics workflows using AProvBio. *Int. J. Computational Biology and Drug Design*, v. 12, n. 2, 2019. 1, 4, 13, 24, 25, 31

ANDERS, Simon; PYL, Paul Theodor; HUBER, Wolfgang. Htseq—a python framework to work with high-throughput sequencing data. *Bioinformatics*, Oxford University Press, v. 31, n. 2, p. 166–169, 2015. 50

ASSIS, Marcio RM; BITTENCOURT, Luiz Fernando. A survey on cloud federation architectures: identifying functional and non-functional properties. *Journal of Network and Computer Applications*, Elsevier, v. 72, p. 51–71, 2016. 3, 18

BARGA, Roger; GANNON, Dennis. Scientific versus business workflows. In: *Workflows for e-Science*. [S.l.]: Springer, 2007. p. 9–16. 7

BARRIL, Jose Farnesio Huesca; RUYTER, Jeff; TAN, Qing. A view on internet of things driving cloud federation. In: IEEE. *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. [S.l.], 2016. p. 221–226. 16

BIRAN, Yahav *et al.* Federated cloud computing as system of systems. In: IEEE. *2017 International Conference on Computing, Networking and Communications (ICNC)*. [S.l.], 2017. p. 711–718. x, 19, 20, 26

BITTMAN, Thomas. *The evolution of the cloud computing market. Gartner Blog Network*. 2008. 18

BLEIDORN, Christoph. Assembly and data quality. In: *Phylogenomics*. [S.l.]: Springer, 2017. p. 81–103. 50

BOETTIGER, Carl. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, ACM, v. 49, n. 1, p. 71–79, 2015. 22

BOLGER, Anthony M; LOHSE, Marc; USADEL, Bjoern. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, Oxford University Press, v. 30, n. 15, p. 2114–2120, 2014. 51

DE BREVERN, Alexandre G *et al.* 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*, Hindawi, v. 2015, 2015. 21

BUNEMAN, Peter; KHANNA, Sanjeev; WANG-CHIEW, Tan. Why and where: A characterization of data provenance. In: SPRINGER. *International conference on database theory*. [S.l.], 2001. p. 316–330. 9

BUNGE, Mario. Treatise on basic philosophy: Volume 3: Ontology 1: The furniture of the world. *Reidel, Boston*, p. 77, 1977. 10

BUYYA, Rajkumar; RANJAN, Rajiv; CALHEIROS, Rodrigo N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: SPRINGER. *International Conference on Algorithms and Architectures for Parallel Processing*. [S.l.], 2010. p. 13–31. 18

CAMACHO, C *et al.* Blast command line applications user manual. 2013. *Reference Source*, v. 10, p. 50–51, 2013. 9

CARPENTER, Jeff; HEWITT, Eben. *Cassandra: The Definitive Guide: Distributed Data at Web Scale*. [S.l.]: "O'Reilly Media, Inc.", 2016. 40

Cassandra. *Cassandra*. 2019. Disponível em: <http://cassandra.apache.org>. 22

CELESTI, Antonio *et al.* How to enhance cloud architectures to enable cross-federation. In: IEEE. *2010 IEEE 3rd international conference on cloud computing*. [S.l.], 2010. p. 337–345. 16, 18

CHANDRABABU, Suganya; BASTOLA, Dhundy. Graph model for the identification of multi-target drug information for culinary herbs. In: SPRINGER. *International Work-Conference on Bioinformatics and Biomedical Engineering*. [S.l.], 2019. p. 498–512. 5

CHEBOTKO, Artem; KASHLEV, Andrey; LU, Shiyong. A big data modeling methodology for apache cassandra. In: IEEE. *2015 IEEE International Congress on Big Data*. [S.l.], 2015. p. 238–245. 40

CHIGURUPATI, Swaroop *et al.* No sql approach for handling bioinformatics data using mongodb. In: *Emerging Technologies in Data Mining and Information Security*. [S.l.]: Springer, 2019. p. 281–287. 5, 21

CHODOROW, Kristina. *MongoDB: the definitive guide: powerful and scalable data storage*. [S.l.]: "O'Reilly Media, Inc.", 2013. 37

CORBELLINI, Alejandro *et al.* Persisting big-data: The NoSQL landscape. *Information Systems*, Elsevier, v. 63, p. 1–23, 2017. 2, 4, 21, 43

COSTA, Raquel L *et al.* Gennet: an integrated platform for unifying scientific workflows and graph databases for transcriptome data analysis. *PeerJ*, PeerJ Inc., v. 5, p. e3509, 2017. 24, 25

CouchDB. *CouchDB*. 2019. Disponível em: <http://couchdb.apache.org>. 22

DEO, Narsingh. *Graph theory with applications to engineering and computer science*. [S.l.]: Courier Dover Publications, 2017. 35

DigitalOcean. *DigitalOcean*. 2019. Disponível em: <https://www.digitalocean.com>. 30

Docker Hub. *Docker Hub*. 2019. Disponível em: <https://hub.docker.com>. 46

Docker Inc. *Docker*. 2019. Disponível em: <https://www.docker.com/>. 5, 22

EKBLOM, Robert; WOLF, Jochen BW. A field guide to whole-genome sequencing, assembly and annotation. *Evolutionary applications*, Wiley Online Library, v. 7, n. 9, p. 1026–1042, 2014. 9

VAN ERVEN, Gustavo Cordeiro Galvão *et al.* Designing graph databases with graphed. *Journal of Database Management (JDM)*, IGI Global, v. 30, n. 1, p. 41–60, 2019. 36

FERNANDES, Diogo; BERNARDINO, Jorge. Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications,{DATA}*. [S.l.: s.n.], 2018. p. 373–380. 35

FERREIRA, Guilherme R.; FILIPE JR., Carlos; OLIVEIRA, Daniel de. Uso de SGBDs NoSQL na gerência da proveniência distribuída em workflows científicos. *XXIX Simpósio Brasileiro de Bancos de Dados*, 2014. 23, 25

FOSTER, Ian *et al.* Cloud computing and grid computing 360-degree compared. In: IEEE. *Grid Computing Environments Workshop, 2008. GCE'08*. [S.l.], 2008. p. 1–10. 14

FREIRE, Juliana *et al.* Provenance for computational tasks: A survey. *Computing in Science & Engineering*, IEEE, v. 10, n. 3, 2008. 1, 5, 10

GEORGAKOPOULOS, Diimitrios; HORNICK, Mark; SHETH, Amit. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, Springer, v. 3, n. 2, p. 119–153, 1995. 1, 8

GESSERT, Felix *et al.* Nosql database systems: a survey and decision guidance. *Computer Science-Research and Development*, Springer, v. 32, n. 3-4, p. 353–365, 2017. 21

GONÇALVES, João *et al.* Performance analysis of data filtering in scientific workflows. *Journal of Information and Data Management*, v. 4, n. 1, p. 17, 2013. 31

Google. *Google Cloud Platform*. 2019. Disponível em: <https://cloud.google.com>. x, 27, 28, 30

GROZEV, Nikolay; BUYYA, Rajkumar. Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, Wiley Online Library, v. 44, n. 3, p. 369–390, 2014. 2

GUEDES, Thaylon *et al.* A practical roadmap for provenance capture and data analysis in spark-based scientific workflows. In: IEEE. *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. [S.l.], 2018. p. 31–41. 10

GUO, Yan *et al.* Three-stage quality control strategies for dna re-sequencing data. *Briefings in bioinformatics*, Oxford Univ Press, p. bbt069, 2013. 50

GUREVICH, Alexey *et al.* Quast: quality assessment tool for genome assemblies. *Bioinformatics*, Oxford University Press, v. 29, n. 8, p. 1072–1075, 2013. 51

HAAS, Brian J *et al.* De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. *Nature protocols*, Nature Publishing Group, v. 8, n. 8, p. 1494–1512, 2013. 8, 50

HAN, Jing *et al.* Survey on nosql database. In: IEEE. *2011 6th international conference on pervasive computing and applications*. [S.l.], 2011. p. 363–366. 3, 21

HARTIG, Olaf; ZHAO, Jun. Publishing and consuming provenance metadata on the web of linked data. In: *Provenance and annotation of data and processes*. [S.l.]: Springer, 2010. p. 78–90. 11

HASHAM, Khawar; MUNIR, Kamran. Reproducibility of scientific workflows execution using cloud-aware provenance (recap). *Computing*, Springer, v. 100, n. 12, p. 1299–1333, 2018. 24, 25

HASHAM, Khawar; MUNIR, Kamran; MCCLATCHEY, Richard. Cloud infrastructure provenance collection and management to reproduce scientific workflows execution. *Future Generation Computer Systems*, Elsevier, v. 86, p. 799–820, 2018. 24, 25

HBase. *HBase*. 2019. Disponível em: <https://hbase.apache.org>. 22

HECHT, Robin; JABLONSKI, Stefan. Nosql evaluation: A use case oriented survey. In: IEEE. *Cloud and Service Computing (CSC), 2011 International Conference on*. [S.l.], 2011. p. 336–341. 21

HONDO, Fernanda. *Gerenciamento de Proveniência de Dados de Workflows de Bioinformática em Ambiente de Nuvem Computacional*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade de Brasília, Brasília, DF, Brasil, 2018. 26, 38

HONDO, Fernanda *et al.* Data provenance management for bioinformatics workflows using nosql database systems in a cloud computing environment. In: IEEE. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. [S.l.], 2017. p. 1929–1934. 1, 2, 4, 13, 24, 25, 31, 37, 51

HUACARPUMA, Ruben Cruz. *Modelo de dados para um Pipeline de seqüenciamento de alto desempenho transcritômico*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade de Brasília, Brasília, DF, Brasil, 2012. 9

IBM. *IBM Cloud*. 2019. Disponível em: <https://cloud.ibm.com>. x, 29, 30

JOSHI, NA; FASS, JN *et al. Sickle: A sliding-window, adaptive, quality-based trimming tool for FastQ files (Version 1.33)[Software]*. 2011. 50

JUDGE, Kim *et al.* Comparison of bacterial genome assembly software for minion data and their applicability to medical microbiology. *Microbial genomics*, Microbiology Society, v. 2, n. 9, 2016. 51

KAUR, Karamjit; RANI, Rinkle. Modeling and querying data in nosql databases. In: IEEE. *2013 IEEE International Conference on Big Data*. [S.l.], 2013. p. 1–7. 37

KIM, Baekdoo *et al.* Bio-docklets: virtualization containers for single-step execution of ngs pipelines. *GigaScience*, 2017. 23

KIM, Daehwan; LANGMEAD, Ben; SALZBERG, Steven L. Hisat: a fast spliced aligner with low memory requirements. *Nature methods*, Nature Publishing Group, v. 12, n. 4, p. 357, 2015. 50

KIM, G *et al.* How to Create World-Class Agility. *Reliability and Security in Technology Organizations*, v. 2, 2016. 14

KOGIAS, Dimitrios G; XEVGENIS, Michael G; PATRIKAKIS, Charalampos Z. Cloud federation and the evolution of cloud computing. *Computer*, IEEE, v. 49, n. 11, p. 96–99, 2016. 16

KURZE, Tobias *et al.* Cloud federation. *Cloud Computing*, Citeseer, v. 2011, p. 32–38, 2011. 17

LATGÉ, Jean-Paul. Aspergillus fumigatus and aspergillosis. *Clinical microbiology reviews*, Am Soc Microbiol, v. 12, n. 2, p. 310–350, 1999. 51, 57

LEIPZIG, Jeremy. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, Oxford University Press, v. 18, n. 3, p. 530–536, 2017. 1

LI, Heng *et al.* The sequence alignment/map format and samtools. *Bioinformatics*, Oxford University Press, v. 25, n. 16, p. 2078–2079, 2009. 50

LI, Tao *et al.* Provenancelens: Service provenance management in cloud. In: *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. [S.l.: s.n.], 2014. 9, 23, 25

MARINHO, A. *et al.* A strategy for provenance gathering in distributed scientific workflows. In: *2009 Congress on Services - I*. [S.l.: s.n.], 2009. p. 344–347. ISSN 2378-3818. 2, 10

MATTOSO, Marta *et al.* Experiences in using provenance to optimize the parallel execution of scientific workflows steered by users. In: *Workshop of Provenance Analytics*. [S.l.: s.n.], 2014. v. 1. 1, 7

MATTOSO, Marta *et al.* Gerenciando experimentos científicos em larga escala. *SBC-SEMISH*, v. 8, p. 121–135, 2008. 8, 13

MELL, Peter; GRANCE, Tim; OTHERS. The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011. 14

MENDES, Felipe Lopes de Souza. *BioNimbuZ 2: uma plataforma de federação de nuvens em uma arquitetura orientada a microsserviços*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade de Brasília, Brasília, DF, Brasil, 2018. 25

Microsoft. *Microsoft Azure*. 2019. Disponível em: <https://azure.microsoft.com>. 29

MOHAMED, Mohamed A; ALTRAFI, Obay G; ISMAIL, Mohammed O. Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, v. 3, n. 03, p. 598–601, 2014. 20

MongoBD. *MongoDB*. 2019. Disponível em: <https://www.mongodb.com>. 22

MOREAU, Luc *et al. Governance of the Open Provenance Model*. [S.l.], 2009. Disponível em: <URLhttp://twiki.ipaw.info/pub/OPM/WebHome/governance.pdf>. 11

MOREAU, Luc; MISSIER, Paolo (Ed.). *PROV-DM: The PROV Data Model*. [S.l.], 2013. Disponível em: <http://eprints.soton.ac.uk/356851/>. 11

MOURA, Breno Rodrigues de. *Arquitetura de um Controlador de SLA para Ambiente de Nuvens Federadas*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, DF, Brasil, 2017. 18

Neo4J. *Neo4J*. 2019. Disponível em: <https://neo4j.com>. 22

NETWORK, EON (Environmental Omics). *Bio-Linux 8*. 2018. 46

OKONECHNIKOV, Konstantin *et al.* Unipro ugene: a unified bioinformatics toolkit. *Bioinformatics*, Oxford University Press, v. 28, n. 8, p. 1166–1167, 2012. xi, 77

OLIVEIRA, Ary Henrique de; OLIVEIRA, Daniel de; MATTOSO, Marta. Clouds and reproducibility: A way to go to scientific experiments? In: *Cloud Computing*. [S.l.]: Springer, 2017. p. 127–151. 1, 7, 9, 10, 13, 14

OLIVEIRA, Wellington Moreira de *et al.* Captura e consulta a dados de proveniência retrospectiva implícita intra-atividade. *Anais Do XXIX Simpósio Brasileiro de Banco de Dados*, p. 37–46, 2014. 31

OPM. *The OPM Provenance Model (OPM)*. 2019. Disponível em: <https://openprovenance.org/opm/>. 1

OrientDB. *OrientDB*. 2019. Disponível em: <https://orientdb.com>. 22

PAULA, Renato de. *Proveniência de dados em workflows de bioinformática*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade de Brasília, Brasília, DF, Brasil, 2013. 10

PAULA, Renato de *et al.* Provenance in bioinformatics workflows. *BMC bioinformatics*, BioMed Central, v. 14, n. 11, p. S6, 2013. 1, 4, 8, 10, 11, 13, 23, 25, 31

PAULINO, Carlos *et al.* Captura de metadados de proveniencia para workflows científicos em nuvens computacionais. *Anais Do XXV Simpósio Brasileiro de Banco de Dados*, 2010. 1, 2, 14

Project Voldemort. *Voldemort*. 2019. Disponível em: <http://www.project-voldemort.com/voldemort>. 21

RAM, Sudha; LIU, Jun. Understanding the semantics of data provenance to support active conceptual modeling. In: *Active conceptual modeling of learning*. [S.l.]: Springer, 2007. p. 17–29. 10

Redis. *Redis*. 2019. Disponível em: <https://redis.io>. 21

ROBINSON, Ian; WEBBER, Jim; EIFREM, Emil. *Graph databases*. [S.l.]: "O'Reilly Media, Inc.", 2013. 36

ROSA, Michel *et al.* Bionimbuz: A federated cloud platform for bioinformatics applications. In: IEEE. *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*. [S.l.], 2016. p. 548–555. 8

SAHOO, Satya S; SHETH, Amit P. Provenir ontology: Towards a framework for escience provenance management. The Ohio Center of Excellence in Knowledge Enabled Computing (Kno.e.sis), 2009. 11

SALDANHA, Hugo Vasconcelos. *Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, DF, Brasil, 2012. 8, 16

SEMPÉRÉ, Guilhem *et al.* Gigwa—genotype investigator for genome-wide analyses. *GigaScience*, BioMed Central, v. 5, n. 1, p. 25, 2016. 23, 25

SHAO, Borong; CONRAD, T. Are nosql data stores useful for bioinformatics researchers?,". *International Journal on Recent and Innovation Trends in Computing and Communication*, v. 3, n. 3, p. 1704–1708, 2015. 21

SIMPSON, Jared T *et al.* Abyss: a parallel assembler for short read sequence data. *Genome research*, Cold Spring Harbor Lab, v. 19, n. 6, p. 1117–1123, 2009. 51

SOTOMAYOR, B *et al.* Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, v. 13, p. 5, 2009. 14

TALREJA, Disha *et al.* Performance scaling of cassandra on high-thread count servers. In: ACM. *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. [S.l.], 2019. p. 179–187. 5

TOOSI, Adel Nadjaran; CALHEIROS, Rodrigo N; BUYYA, Rajkumar. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, ACM, v. 47, n. 1, p. 7, 2014. x, 3, 16, 17

TRUYEN, Eddy *et al.* Evaluation of container orchestration systems for deploying and managing nosql database clusters. In: *Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), Zurich, Switzerland*. [S.l.: s.n.], 2018. p. 17–20. 22

VAQUERO, Luis M *et al.* A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, ACM, v. 39, n. 1, p. 50–55, 2008. 14

VOORSLUYS, William; BROBERG, James; BUYYA, Rajkumar. Cloud computing: Principles and paradigms. *Ch. Introduction to Cloud Computing*, p. 1–44, 2011. 2, 14, 15

W3C. *PROV-DM: The PROV Data Model*. 2019. Disponível em: <https://www.w3.org/TR/2013/REC-prov-dm-20130430>. x, 1, 2, 11, 12

WERCELENS, P. *Tutorial - Gerenciamento de Proveniência de Dados de Workflows de Bioinformática em Ambiente de Nuvem Federada*. 2019. Disponível em: <https://github.com/Polyane/dataprovenance>. 5, 46, 49, 51

ZERBINO, Daniel R; BIRNEY, Ewan. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, Cold Spring Harbor Lab, v. 18, n. 5, p. 821–829, 2008. 50

Apêndice A

Workflow 1: análise quantitativa de RNA-seq de células endoteliais humanas

Workflow para mapeamento de *reads* para uma referência.
Illumina HiSeq2500 RNA-Seq diponível em NCBI[1] sob o número de acesso SRR5181508.

```
# downloading and converting raw files (reads)
fastq-dump SRR5181508

# downloading the GFF file of human genome
$ wget ftp://ftp.ensembl.org/pub/release-88/gtf/homo_sapiens/Homo_sapiens
    .GRCh38.88.gtf.gz
$ gzip -d Homo_sapiens.GRCh38.88.gtf.gz

# downloading the human chromosome 22
$ wget ftp://ftp.ensembl.org/pub/release-88/fasta/homo_sapiens/dna/
    Homo_sapiens.GRCh38.dna.chromosome.22.fa.gz
$ gzip -d Homo_sapiens.GRCh38.dna.chromosome.22.fa.gz

# Filtering with sickle
# se => sigle sequences; --qual-type => type of quality; --output-file +>
     output file;
```

[1] https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR5181508

```
# -q => flag designates the minimum quality; -l => the minimum read
    length;
$ sickle se --fastq-file SRR5181508.fastq --qual-type sanger --output-
    file SRR5181508_FILTERED.fastq -q 30 -l 25

# Building the hisat2 index;
# -p number of processors
$ hisat2-build -p 4 Homo_sapiens.GRCh38.dna.chromosome.22.fa chromosome
    .22.hisat2.idx

# Mapping the filtered reads to the refrence;
# -p => number of processors; -q => input in fastq format; -S => output
    file
$ hisat2 -p 2 -x chromosome.22.hisat2.idx -q SRR5181508_FILTERED.fastq -S
     SRR5181508.sam

# Analisys
# Converting formats
$ samtools view -bS SRR5181508.sam > SRR5181508.bam

# Sorting data and converting
$ samtools sort -n SRR5181508.bam -o SRR5181508_SORTED.sn
$ samtools view -h -o SRR5181508_SORTED.sn.sam SRR5181508.bam

# Counting reads in features
# -m => reads overlapping more than one feature; -s => strand-specific
    assay;
# -o output file; -a => skip reads with alignment quality lower than x (x
     = default: 10);
$ htseq-count -m intersection-nonempty -s no -a 10 SRR5181508_SORTED.sn.
    bam Homo_sapiens.GRCh38.88.gtf -o SRR5181508.count
```

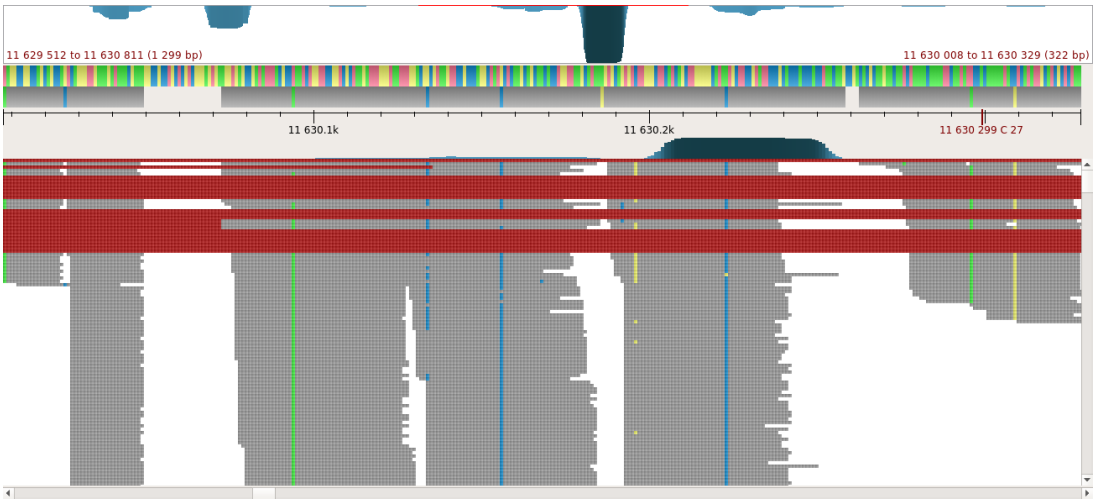


Figura A.1: A partial view of the mapping results of Workflow 1 generated using Ugene (OKONECHNIKOV *et al.*, 2012).

Apêndice B

Workflow 2: conjunto genômico da bactéria *Enterobacter kobei*

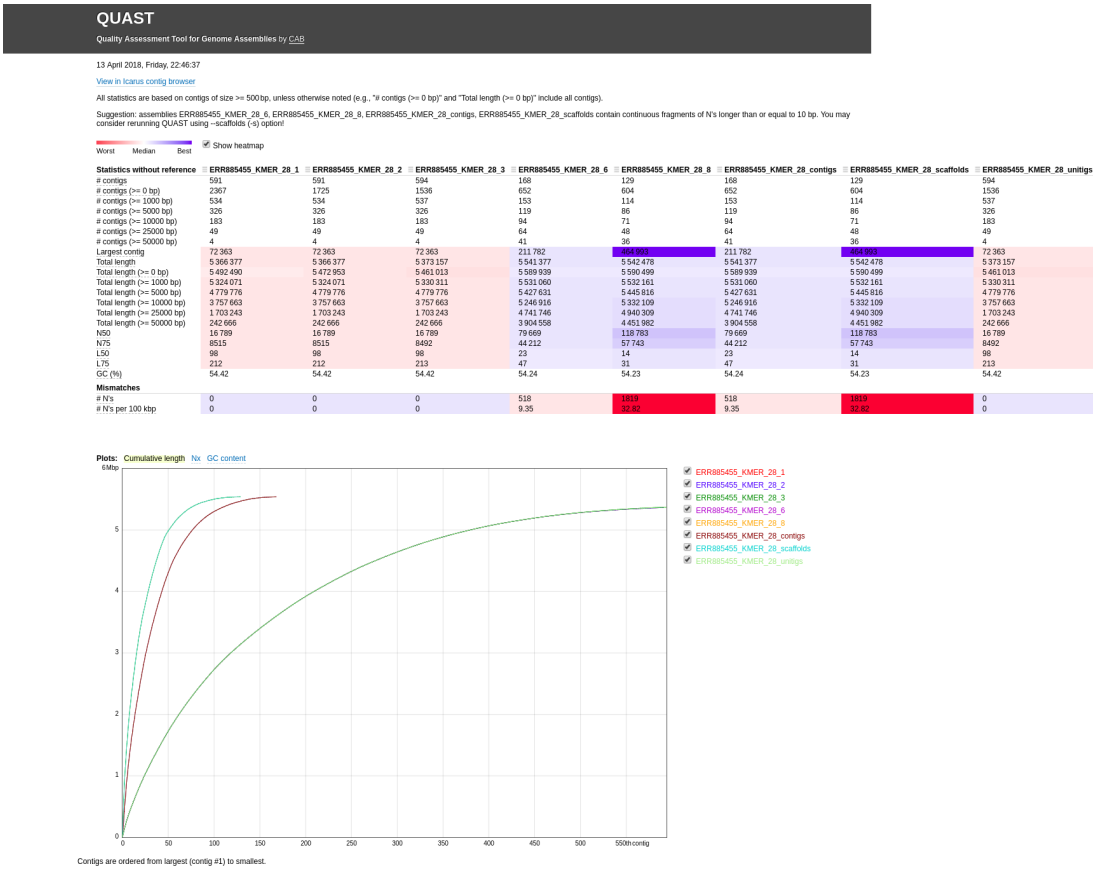
Workflow para montagem *de novo*.

Illumina HiSeq 2000 sequências *paired end* disponíveis em *European Nucleotide Archive* (ENA) [1] sob o número de acesso ERR885455

```
# download raw files (reads)
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR885/ERR885455/ERR885455_1.
    fastq.gz
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR885/ERR885455/ERR885455_2.
    fastq.gz

# Filtering with Trimmomatic
$ java -jar /opt/Trimmomatic-0.36/trimmomatic-0.36.jar PE -phred33
    ERR885455_1.fastq.gzERR885455_2.fastq.gz
ERR885455_1_FILTERED.fastq ERR885455_1_UNPAIRED.fastq
    ERR885455_2_FILTERED.fastq ERR885455_2_UNPAIRED.fastq ILLUMINACLIP:/
    opt/Trimmomatic-0.36/adapters/TruSeq3-PE.fa:2:30:10 LEADING:3
    TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36 -validatePairs

# de novo assembly with ABySS
# abyss-pe => for paired-end reads; k => size of the kmer;
# np =>number of processors (depends on mpi modules installed);
```

[1] https://www.ebi.ac.uk/ena/data/view/ERR885455

```
# in => input file for forward/reverse trimmed reads; name => output file
     prefix
$ abyss-pe in='ERR885455_1_FILTERED.fastq ERR885455_2_FILTERED.fastq' k
    =28 name=ERR885455_KMER_28 np=4

# Statistical analisys with QUAST
$ python /opt/quast/quast.py -o ERR885455_stats ERR885455*.fa
```



QUAST

Quality Assessment Tool for Genome Assemblies by CAB

13 April 2018, Friday, 22:46:37

View in Icarus contig browser

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

Suggestion: assemblies ERR885455_KMER_28_6, ERR885455_KMER_28_8, ERR885455_KMER_28_contigs, ERR885455_KMER_28_scaffolds contain continuous fragments of N's longer than or equal to 10 bp. You may consider rerunning QUAST using --scaffolds (-s) option!

Worst Median Best — Show heatmap

Statistics without reference	ERR885455_KMER_28_1	ERR885455_KMER_28_2	ERR885455_KMER_28_3	ERR885455_KMER_28_6	ERR885455_KMER_28_8	ERR885455_KMER_28_contigs	ERR885455_KMER_28_scaffolds	ERR885455_KMER_28_unitigs
# contigs	591	591	594	168	129	168	129	594
# contigs (>= 0 bp)	2367	1725	1536	652	604	652	604	1536
# contigs (>= 1000 bp)	534	534	537	153	114	153	114	537
# contigs (>= 5000 bp)	326	326	326	119	86	119	86	326
# contigs (>= 10000 bp)	183	183	183	94	71	94	71	183
# contigs (>= 25000 bp)	49	49	49	64	48	64	48	49
# contigs (>= 50000 bp)	4	4	4	41	36	41	36	4
Largest contig	72 363	72 363	72 363	211 782	454 205	211 782	454 205	72 363
Total length	5 366 377	5 366 377	5 373 157	5 541 377	5 542 478	5 541 377	5 542 478	5 373 157
Total length (>= 0 bp)	5 492 490	5 472 953	5 461 013	5 589 939	5 590 499	5 589 939	5 590 499	5 461 013
Total length (>= 1000 bp)	5 324 071	5 324 071	5 330 311	5 531 060	5 532 161	5 531 060	5 532 161	5 330 311
Total length (>= 5000 bp)	4 779 776	4 779 776	4 779 776	5 427 631	5 445 816	5 427 631	5 445 816	4 779 776
Total length (>= 10000 bp)	3 757 663	3 757 663	3 757 663	5 246 916	5 332 109	5 246 916	5 332 109	3 757 663
Total length (>= 25000 bp)	1 703 243	1 703 243	1 703 243	4 741 746	4 940 309	4 741 746	4 940 309	1 703 243
Total length (>= 50000 bp)	242 666	242 666	242 666	3 904 558	4 451 982	3 904 558	4 451 982	242 666
N50	16 789	16 789	16 789	79 669	118 783	79 669	118 783	16 789
N75	8515	8515	8492	44 212	57 743	44 212	57 743	8492
L50	98	98	98	23	14	23	14	98
L75	212	212	213	47	31	47	31	213
GC (%)	54.42	54.42	54.42	54.24	54.23	54.24	54.23	54.42
Mismatches								
# N's	0	0	0	518	1819	518	1819	0
# N's per 100 kbp	0	0	0	9.35	32.82	9.35	32.82	0

Plots: Cumulative length Nx GC content

Contigs are ordered from largest (contig #1) to smallest.

Figura B.1: Quark stats for the assembly of *Enterobacter kobei* using a K-mer = 28.

Apêndice C

Resultados da Consulta 1

Cassandra:

```
nomecluster | nomeprovedor    | hostname | sistemaoperacional | tipoprovisionamento | cpu | ram | disco | tipodisco
------------+-----------------+----------+--------------------+---------------------+-----+-----+-------+-----------
 ProvBio    | Microsoft Azure | ProvBio1 | Ubuntu 14          | Provedor            | 2   | 4GB | 80GB  | SSD
 ProvBio    | Microsoft Azure | ProvBio2 | CentOS 7           | Provedor            | 2   | 4GB | 80GB  | SSD
 ProvBio    | DigitalOcean    | ProvBio3 | CentOS 7           | Provedor            | 2   | 4GB | 80GB  | SSD
 ProvBio    | Google Cloud    | ProvBio4 | CentOS 7           | Provedor            | 2   | 4GB | 80GB  | SSD
```

Figura C.1: Cassandra Resultado da Consulta 1.

OrientDB:

```
+----+------------+----------------+---------+------------------+--------------------+---+---+-----+----------+
|#   |nomecluster | nomeprovedor   |hostname |sistemaoperacional|tipoprovisionamento|cpu|ram|disco|tipodisco |
+----+------------+----------------+---------+------------------+--------------------+---+---+-----+----------+
|0   |ProvBio     |Microsoft Azure |ProvBio1 |Ubuntu 14         |Provedor            |2  |4GB|80GB |SSD       |
|1   |ProvBio     |Microsoft Azure |ProvBio2 |CentOS 7          |Provedor            |2  |4GB|80GB |SSD       |
|2   |ProvBio     |DigitalOcean    |ProvBio3 |CentOS 7          |Provedor            |2  |4GB|80GB |SSD       |
|3   |ProvBio     |Google Cloud    |ProvBio4 |CentOS 7          |Provedor            |2  |4GB|80GB |SSD       |
+----+------------+----------------+---------+------------------+--------------------+---+---+-----+----------+
```

Figura C.2: OrientDB Resultado da Consulta 1.

MongoDB:


```
{
   "_id": "ObjectId("5d0b1d40eaa56df43210ff90"),
   "nomeCluster":"ProvBio",
   "maquinas":[
      {
         "hostname":"ProvBio1",
         "sistemaOperacional":"Ubuntu 14",
```

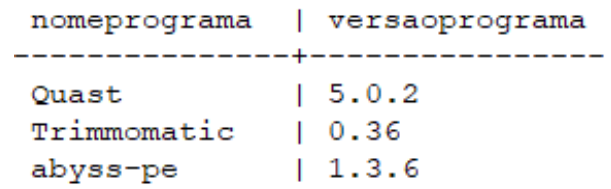
```
    "cpu":2,
    "ram":"4GB",
    "disco":"80GB",
    "tipoDisco":"SSD",
    "tipoProvisionamento":"Provedor",
    "provedor":{
      "nome":"Microsoft Azure"
    }
  },
  {
    "hostname":"ProvBio2",
    "sistemaOperacional":"CentOS 7",
    "cpu":2,
    "ram":"4GB",
    "disco":"80GB",
    "tipoDisco":"SSD",
    "tipoProvisionamento":"Provedor",
    "provedor":{
      "nome":"Microsoft Azure"
    }
  },
  {
    "hostname":"ProvBio3",
    "sistemaOperacional":"CentOS 7",
    "cpu":2,
    "ram":"4GB",
    "disco":"80GB",
    "tipoDisco":"SSD",
    "tipoProvisionamento":"Provedor",
    "provedor":{
      "nome":"DigitalOcean"
    }
  },
  {
    "hostname":"ProvBio4",
    "sistemaOperacional":"CentOS 7",
    "cpu":2,
```

```
        "ram":"4GB",
        "disco":"80GB",
        "tipoDisco":"SSD",
        "tipoProvisionamento":"Provedor",
        "provedor":{
           "nome":"Google Cloud"
        }
     }
  ]
}
```

Apêndice D

Resultados da Consulta 2

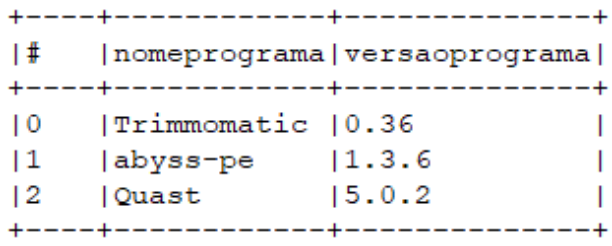
Cassandra:



nomeprograma	versaoprograma
Quast	5.0.2
Trimmomatic	0.36
abyss-pe	1.3.6

Figura D.1: Cassandra Resultado da Consulta 2.

OrientDB:



#	nomeprograma	versaoprograma
0	Trimmomatic	0.36
1	abyss-pe	1.3.6
2	Quast	5.0.2

Figura D.2: OrientDB Resultado da Consulta 2.

MongoDB:

```
{
   "_id": ObjectId("5d0b3bad1262ff1c254a2976"),
   "experimento":[
      {
         "atividade":[
```

```
            {
               "nomeprograma":"Trimmomatic",
               "versaoprograma":"0.36"
            },
            {
               "nomeprograma":"abyss-pe",
               "versaoprograma":"1.3.6"
            },
            {
               "nomeprograma":"Quast",
               "versaoprograma":"5.0.2"
            }
         ]
      }
   ]
}
```