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

Política de Armazenamento de Dados em Nuvens Federadas para Dados Biológicos

Ricardo Fernandes Gallon

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 F. Araújo

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof.ª Dr.ª Alba Cristina Magalhaes Alves de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof.ª Dr.ª Maria Emilia Machado Telles Walter — CIC/UnB
Prof. Dr. Roberto Coiti Togawa — EMBRAPA/Recursos Genéticos e Biotecnologia

CIP — Catalogação Internacional na Publicação

Gallon, Ricardo Fernandes.

Política de Armazenamento de Dados em Nuvens Federadas para Dados Biológicos / Ricardo Fernandes Gallon. Brasília : UnB, 2014.
133 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. Política de armazenamento, 2. Computação em nuvem,
3. Armazenamento de dados, 4. Nuvem Federada.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

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

Política de Armazenamento de Dados em Nuvens Federadas para Dados Biológicos

Ricardo Fernandes Gallon

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

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

Prof.ª Dr.ª Maria Emilia Machado Telles Walter
CIC/UnB

Prof. Dr. Roberto Coiti Togawa
EMBRAPA/Recursos Genéticos e Biotecnologia

Prof.ª Dr.ª Alba Cristina Magalhaes Alves de Melo
Coordenador do Mestrado em Informática

Brasília, 07 de Julho de 2014

Dedicatória

A minha esposa, Laura, companheira, parceira, que sempre esteve ao meu lado, me dando apoio para realização de meus projetos.

A meus pais, Hilário e Iraci, que desde quando eu era criança acreditaram em mim, me incentivando a "alçar voos mais altos", além de serem exemplos de força e dedicação.

A Deus por guiar os meus caminhos.

Agradecimentos

Agradeço a todos os professores que tive o prazer de interagir durante este curso, em especial a minha orientadora Prof.ª Dr.ª Maristela Terto de Holanda, por sua dedicação e paciência em me ajudar neste projeto, minha coorientadora Prof.ª Dr.ª Aletéia Patrícia F. Araújo, pela disponibilidade e auxílio nesta pesquisa e a Prof.ª Dr.ª Maria Emilia Machado Telles Walter, uma das primeiras professoras que tive neste curso, e que me mostrou a paixão pela docência e a disposição para fazer com seus alunos adquirissem o conhecimento repassado.

Não poderia deixar de agradecer também aos colegas Edward Ribeiro e Hugo Saldanha pela auxílio, principalmente no início deste projeto. Suas ajudas foram de fundamental importância para a execução deste trabalho.

A meus superiores e colegas de trabalho na CAPES, que foram compreensivos com minhas ausências, principalmente para a realização das disciplinas do curso, sem este auxílio seria impossível a realização deste projeto.

Resumo

A computação em nuvem tem possibilitado a integração de diversos provedores para a execução de tarefas de forma mais rápida em comparação a utilização dos modelos anteriores, e uma visão ao usuário de que os recursos de armazenamento e processamento são ilimitados. A Bioinformática, que lida com grande volume de informações, pode utilizar-se da infraestrutura de computação em nuvem para disponibilizar suas ferramentas, para serem utilizadas nos fluxos de trabalhos, chamados *workflows*. Além disso, as instituições podem associar-se a outras instituições para formar uma federação de nuvens computacionais, proporcionando maior flexibilidade na escolha de provedores de serviço. Neste contexto, o desempenho da execução de *workflows* de Bioinformática é fortemente afetado pelo armazenamento e recuperação de dados, devido ao grande volume de informações das sequências genômicas. Desta forma, a escolha da melhor nuvem para estas operações é fundamental para a eficiência da execução do *workflow*. O presente trabalho tem por objetivo propor uma política de armazenamento de dados genômicos para nuvens computacionais federadas buscando, através da definição de alguns critérios de seleção e estratégias, diminuir o tempo de transferência dos dados e assim contribuir para a diminuição do tempo total de execução do *workflow*. Foi realizado um estudo de caso, com dados reais, utilizando a plataforma BioNimbuZ[1], que é uma arquitetura para execução de *workflows* de bioinformática no ambiente de nuvens federadas. Com os resultados obtidos, foi possível determinar o peso de cada critério da política de armazenamento e realizar as análises em relação a política originalmente criada para a arquitetura. Desta forma, a política de armazenamento proposta apresentou ganhos quanto a eficiência, principalmente em nuvens com grande poder computacional.

Palavras-chave: Política de armazenamento, Computação em nuvem, Armazenamento de dados, Nuvem Federada.

Abstract

Cloud computing has enabled the integration of providers to perform tasks faster, and has allowed users to view the processing and storage resources as unlimited. Bioinformatics, which handles large volume of information produced by high-throughput sequencing machines, may use the infrastructure of cloud computing to provide tools to be used in workflows. In addition, institutions may join other institutions to form a federation of computing clouds, providing greater flexibility in the choice of service providers. In this context, the performance of the implementation of a workflow is strongly affected by the storage and retrieval of data, due to the large amount of information from genomic sequences. Thus, choosing the best cloud for these operations is critical to the efficiency of the workflow. This work aims to propose a policy for the storage of genomic data for federated cloud computing seeking, by defining some selection criteria and strategies, to reduce the time of data transfer and thus contribute to the reduction of total execution time of the workflow. A case study was carried out with real data, using BioNimbuZ platform, which is an architecture for the implementation of bioinformatics workflows in federated cloud environments. With the results obtained, it was possible to determine the weight of each storage policy criteria and perform the analysis regarding the policy originally created for the architecture. Thus, the storage policy proposal presented efficiency gains, especially in clouds with great computing power.

Keywords: bioinformatic, data storage, replication, cloud computing, federated cloud

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Máquinas de sequenciamento de alto desempenho [6, 7, 8] produzem grandes quantidades de dados biológicos em vários projetos genoma em todo o mundo. Um único projeto genoma gera gigabytes de dados, que são analisadas por ferramentas computacionais que requerem muitos ciclos de CPU e grande espaço de armazenamento. Projetos genoma que buscam desvendar o código genético de um organismo, e análises de bioinformática que utilizam o genoma para encontrar características, em doenças, dentre outros, são apoiados por fluxos de trabalho, chamados *workflows*. Os *workflows* são compostos por ferramentas e bancos de dados, que muitas vezes podem estar localizados em instituições separadas fisicamente.

Nos últimos anos emergiu o paradigma da computação em nuvem [2, 9, 10, 11], que é uma forma de utilizar serviços *online* independentemente de localização física e plataforma. Assim os usuários podem acessar de forma transparente uma grande variedade de infraestruturas de rede e sistemas. Neste ambiente, o processamento e o armazenamento de dados são realizados para dar ao usuário a ilusão de que a quantidade de recursos é ilimitada. No entanto, considerando o crescimento constante das necessidades de poder computacional e de armazenamento decorrente de diversos aplicativos de bioinformática que são continuamente desenvolvidos em diferentes ambientes, a utilização de apenas uma nuvem é restritiva. Por estas necessidades, a utilização de uma federação de nuvens computacionais [12, 13, 14], onde diversas nuvens se unem para compartilhar seus recursos, é muito adequada para a execução de aplicações de bioinformática.

Neste contexto, a arquitetura BioiNimbuZ [4, 1] foi definida como uma plataforma de computação em nuvens federadas que visa integrar e controlar diferentes ferramentas de bioinformática em um ambiente distribuído, de forma transparente, com flexibilidade e tolerância a falhas, fornecendo também o processamento distribuído e grande capacidade

de armazenamento, as quais possibilitam o uso de ferramentas e serviços fornecidos por várias instituições, públicas ou privadas, que podem ser facilmente agregadas à nuvem.

Atualmente, o BioNimbuZ realiza a execução da sequência de *workflows* de bioinformática, inclusive com a troca de arquivos entre as tarefas. Contudo, a política de armazenamento definida é simples e não garante uma execução eficiente do *workflow*. Assim, este trabalho propõe uma política de armazenamento para o BioNimbuZ que contribua para um melhor desempenho do *workflow*, ou seja, uma redução do tempo de transferência dos arquivos e portanto do tempo total de execução. Isto pode ser realizado por meio da escolha do melhor recurso para armazenar ou recuperar os arquivos necessários para a execução do trabalho desejado.

1.2 Motivação

A computação em nuvem tem possibilitado a integração de diversos provedores para a execução de tarefas de forma mais rápida, e uma visão ao usuário de que os recursos de armazenamento e processamento são ilimitados. A união de nuvens, objetivando o compartilhamento de recursos, chamado de federação de nuvens computacionais, por sua vez, além das características citadas, tem se mostrado uma plataforma capaz de integrar diferentes infraestruturas, proporcionando uma maior flexibilidade na escolha de provedores.

A plataforma de federação de nuvens computacionais, denominada BioNimbuZ, foi proposta para a execução de aplicações e *workflows* de bioinformática. Entretanto, na proposta original, a política para o armazenamento de dados foi definida de forma simples com o objetivo de permitir o funcionamento da arquitetura. Dessa forma, a política de armazenamento original implementada no BioNimbuZ selecionava a alocação de dados baseada em um algoritmo *round robin*, um algoritmo simples de escalonamento de processos, atribuindo fatias de tempo para cada processo em partes iguais e de forma circular, e execução de todos os processos sem prioridade. Isto faz com que as aplicações que são executadas no BioNimbuZ e que requerem muita transferência de arquivos grandes tenham um desempenho baixo, sem a devida eficiência das nuvens computacionais.

Neste cenário, a motivação deste trabalho é otimizar o processo de armazenamento e de recuperação dos dados, para as tarefas executadas no ambiente de nuvens federadas BioNimbuZ, por meio de uma escolha correta dos melhores recursos para armazenar arquivos de entrada e saída utilizados nessas tarefas.

1.3 Problema

Atualmente, a política de armazenamento do BioNimbuZ é bastante simples. Assim, os arquivos são armazenados utilizando um algoritmo *round robin* para a seleção do provedor que tenha espaço em disco suficiente para receber o arquivo da operação. Como consequência, o BioNimbuZ:

- Possui uma política de armazenamento de dados implementada de forma básica;
- Não possui mecanismos para garantir a tolerância a falhas, como a realização de redundância, replicação, *backups*, etc;
- O serviço de armazenamento não contribui com o objetivo de proporcionar a execução eficiente de *workflows* de bioinformática.

1.4 Objetivos

O objetivo geral deste trabalho é definir uma política de armazenamento eficiente para nuvens federadas, a fim de diminuir o tempo de transferência dos dados e contribuir para a diminuição do tempo total de execução na plataforma BioNimbuZ.

1.4.1 Objetivos Específicos

No contexto apresentado, e no intuito de atingir o objetivo geral, os objetivos específicos são:

- Definir proposta para armazenamento eficiente no BioNimbuZ;
- Implementar a estratégia proposta;
- Avaliar a solução proposta por meio de *workflows* de bioinformática;
- Integrar a estratégia proposta ao ambiente BioNimbuZ;
- Analisar os tempos de transferência utilizando a política de dados proposta, comparando-os com os da política original do ambiente BioNimbuZ.

1.5 Descrição dos Capítulos

Este trabalho está divido em mais cinco capítulos. No Capítulo 2, são apresentados os conceitos de computação em nuvem e federação de nuvens computacionais. Além disso,

são descritas as tecnologias utilizadas nestes ambientes e a comunicação entre eles. Em seguida, algumas características de armazenamento em nuvem são discutidas.

No Capítulo 3, é apresentado o BioNimbuZ, sua arquitetura e a descrição de seus componentes, além da interação entre os mesmos, identificando os impactos da política de armazenamento escolhida. Cada componente da arquitetura é detalhado mostrando sua função, principalmente, o serviço de armazenamento e como este impacta na execução do *workflow*.

No Capítulo 4, é apresentada a política de armazenamento proposta para otimizar a execução do *workflow*, com a descrição dos fluxos para armazenamento e recuperação, e as razões da escolha de cada critério.

No Capítulo 5, é descrito o estudo de caso, descrevendo o *workflow* e ambiente utilizados, bem como detalhes da implementação. Por fim, os resultados do estudo são discutidos e são realizadas comparações com a política original.

Finalmente, o Capítulo 6 conclui o presente trabalho e sugere alguns trabalhos futuros.

Capítulo 2

Armazenamento de Dados em Nuvens Computacionais

Este capítulo apresenta os conceitos básicos relacionados ao armazenamento de dados em um ambiente de computação em nuvem, os quais são necessários para a definição da política de armazenamento proposta neste trabalho. Na Seção 2.1 são descritos conceitos de computação em nuvem, apresentando as suas definições. Posteriormente, a Seção 2.2 apresenta os conceitos de federação de nuvens computacionais, suas características e seus desafios. Ao final, na Seção 2.3, são tratados o armazenamento de dados em nuvem, as estratégias utilizadas por outras pesquisas e os problemas encontrados.

2.1 Computação em Nuvem

2.1.1 Conceitos Básicos

Computação em nuvem é um modelo para provimento de infraestrutura computacional, na qual as tarefas são distribuídas entre um número de computadores para utilizar os serviços disponíveis. Deste modo, as necessidades do usuário podem ser atendidas de forma transparente, pois a aquisição é por serviços e não por recursos computacionais.

Há muitas definições distintas de computação em nuvem. De acordo com Foster *et al.* [3], computação em nuvem pode ser definida como um paradigma de computação distribuída de larga escala, em que o poder de processamento e de armazenamento são entregues sob demanda para clientes por meio da Internet, por meio de plataforma e serviços abstratos, virtualizados, gerenciados e dinamicamente escalável.

Por outro lado, Buyya *et al.* [12] define computação em nuvem como um modelo para a oferta e a utilização de recursos como serviços sob demanda, em um ambiente com múltiplos provedores, seguindo uma economia de escala.

Vaquero *et al.* [2] propuseram uma definição de nuvem como um grande *pool* de recursos virtualizados, facilmente utilizáveis onde os recursos podem ser dinamicamente reconfigurados de acordo com uma carga variável.

Todavia, diante de diferentes definições, é importante ressaltar que o objetivo da computação em nuvem é oferecer a ideia aos usuários de que eles dispõem de recursos ilimitados, mas pagando apenas por aqueles que realmente forem utilizados (modelo *pay-per-use*). Além disso, a computação em nuvem evita a necessidade do usuário gerenciar a infraestrutura computacional, que é de fato controlada pelos prestadores de serviços.

2.1.2 Características de uma Nuvem

A computação em nuvem representa uma nova maneira de usar os recursos computacionais. Dentre as suas características destacam-se: serviço sob demanda, flexibilidade de acesso, *pool* de recursos compartilhados, elasticidade e medição de utilização de serviço com um mecanismo automatizado para faturar o custo dos clientes. Segundo o *National Institute of Standards and Technology* (NIST) [15], estas características são essenciais no modelo de computação em nuvem. Para esclarecer melhor:

- Serviço sob demanda: o usuário pode usar os serviços da nuvem conforme as suas necessidades, aumentando ou diminuindo capacidades computacionais alocadas, como capacidade de processamento, armazenamento, memória, entre outros. Isto é possível pois o hardware e o software dentro de uma nuvem podem ser automaticamente reconfigurados, sendo as modificações transparantes ao usuário;
- Flexibilidade de acesso: os serviços da nuvem podem ser acessados de qualquer plataforma, assim o cliente pode acessar tanto a partir de um dispositivo móvel quanto de um computador ou qualquer outra plataforma;
- *Pool* de recursos compartilhados: os recursos virtuais são dinamicamente atribuídos conforme demanda, desta forma, o cliente não possui controle sobre a real localizacão dos recursos que está utilizando;
- Elasticidade: capacidade de alocar mais ou menos recursos no momento que for necessário, dando a ilusão de recursos ilimitados. A virtualização contribui para que a elasticidade aconteça de forma rápida, através da criação de várias instâncias de recursos, utilizando apenas um único recurso real. Podem existir dois tipos de virtualização: a paravirtualização, que permite que um único servidor físico possa criar diversos servidores virtuais e, *clustering*, onde múltiplos servidores físicos possam ser tratados como um único servidor virtual;

- Modelo *pay-per-use*: o valor a ser pago é aquele decorrente do tempo e da quantidade de recursos utilizados, ou seja, quanto mais se usa, maior é o valor a ser pago. Assim não é necessário que o usuário reserve uma quantidade exata de recursos. Com isso, uma aplicação pode existir por um curto período de tempo utilizando muitos recursos e/ou por um longo período de tempo utilizando poucos recusos. A cobrança é baseada no consumo dos recursos, como a quantidade de horas utilizadas de CPU, o volume de dados armazenados e a transferência de dados. A grande vantagem deste modelo é a redução dos riscos de subutilização e de saturação da infraestrutura.

A arquitetura de uma nuvem é composta por uma série de elementos, que interagem para o seu funcionamento. Entre eles estão os atores envolvidos no processo e as camadas básicas de abstração da nuvem, onde ocorre a divisão das diferentes funções desempenhadas pelos atores e pela infraestrutura utilizada por eles [12], conforme mostrada na Figura 2.1.

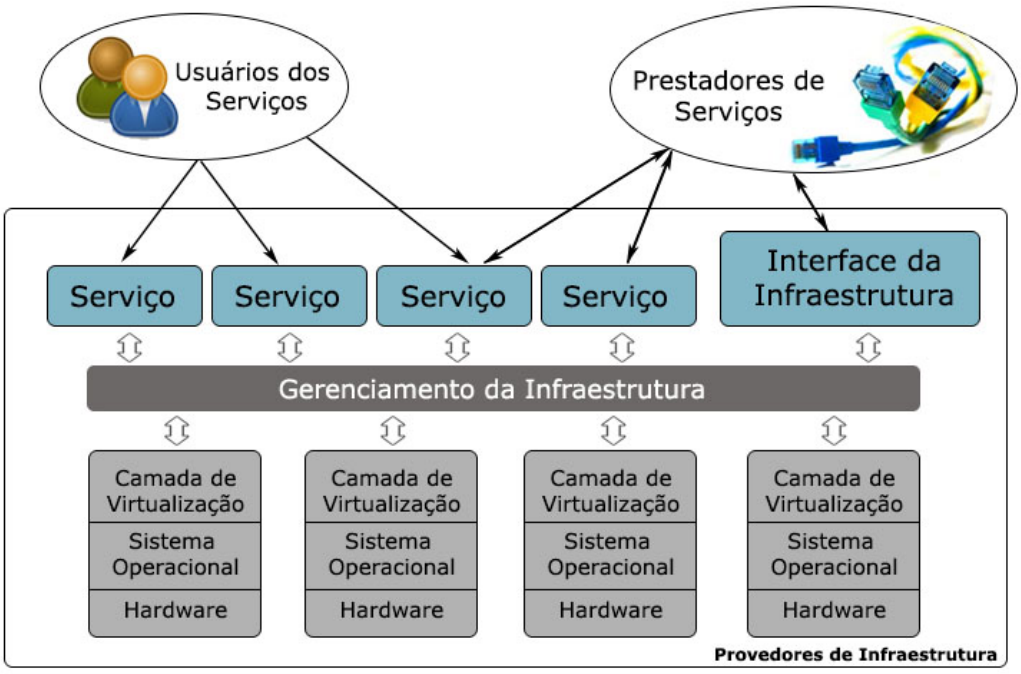

Figura 2.1: Atores Envolvidos na Computação em Nuvem e sua Interação, adaptado de [2].

Na Figura 2.1 são apresentados os atores envolvidos e como acontece as interações entre si. Os três principais atores são: Provedores de Serviço (*ServiceProviders*), Usuários de Serviços (*Service Users*) e os Provedores de Infraestrutura (*Infrastructure Providers*). Os Provedores de Serviços oferecem os serviços que são acessados pelos Usuários de Serviços, por meio de interfaces disponíveis na Internet, e os Provedores de Infraestrutura, que

gerenciam os recursos computacionais e os disponibilizam aos provedores de serviço para que ganhem disponibilidade e reduzam os custos de manutenção [2].

2.1.3 Modelos de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviços. Esses modelos definem um padrão arquitetural para soluções de computação em nuvem. Os modelos de serviços existentes para a computação em nuvem são [2]:

- *Infrastructure-as-a-Service* (IaaS): é um modelo em que uma organização terceiriza os equipamentos utilizados para apoiar as operações, incluindo o armazenamento, o hardware, os servidores e os componentes de rede. O prestador de serviço possui os equipamentos e é responsável pela hospedagem, pelo funcionamento e pela manutenção dos mesmos. O cliente normalmente paga pelos recursos utilizados, entre eles, o tempo de processamento, a quantidade de dados armazenados e/ou de largura de banda utilizada. A *Amazon Elastic Compute Cloud* (EC2) [9] e o GoogleFS [16] são alguns exemplos de ferramentas que fornecem esta camada de serviço.
- *Platform-as-a-Service* (PaaS): o objetivo do PaaS é facilitar o desenvolvimento de aplicações destinadas aos usuários de uma nuvem, criando uma plataforma que agiliza esse processo, utilizando os serviços disponibilizados pelos provedores de infraestrutura. Os provedores dos serviços oferecem APIs para o desenvolvimento de aplicações específicas para a nuvem, permitindo assim, a escalabilidade, bem como a produtividade no desenvolvimento das aplicações. Exemplos de ambientes da camada PaaS são Google App Engine [17] e Windows Azure [18].
- *Software-as-a-Service* (SaaS): nesta camada são fornecidas como serviços as aplicações desenvolvidas, especificamente, para o ambiente de computação em nuvem. Nesta camada os serviços podem utilizar os ambientes de programação fornecidos pelos provedores da camada PaaS, ou podem utilizar diretamente a infraestrutura fornecida pela camada IaaS. No SaaS, o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistema operacional, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas. O Google Docs [19] e o Zoho [20] são alguns exemplos de aplicações que disponibilizam serviços na camada SaaS.

Assim sendo, como pode ser observado na Figura 2.2 a camada SaaS é a de mais alto nível, e através dela são oferecidas diversas aplicações, na forma de serviço, aos usuários. A camada logo abaixo, PaaS, provê os serviços para que as aplicações possam ser desenvolvidas e mantidas no ambiente de computação em nuvem pelos prestadores de

serviço. A camada de infraestrtura (IaaS) é a camada na qual os serviços infraestrutura, tais como os serviços de rede e de armazenamento são disponibilizados.

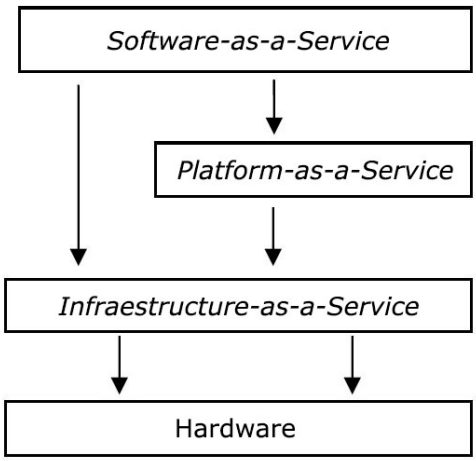

Figura 2.2: Modelo de Serviços da Computação em Nuvem, adaptado de [3].

2.1.4 Modelos de Implantação

Os serviços prestados por uma nuvem podem variar quanto à infraestrutura, à plataforma e aos softwares disponíveis. Os modelos de implantação existentes para a computação em nuvem são [15]:

- Nuvens públicas: a infraestrutura e os serviços são disponibilizados para o público em geral, sendo acessado por qualquer usuário que conheça a localização do serviço;
- Nuvens comunitárias: a infraestrutura da nuvem é compartilhada por várias organizações e serve como ferramenta para um grupo específico de usuários com interesses em comum, podendo ser implantada interna ou externamente às organizações envolvidas;
- Nuvens privadas: a infraestrutura e os serviços são utilizados exclusivamente por uma organização, sendo esta nuvem local ou remota, e administrada pela própria empresa ou por terceiros;
- Nuvens híbridas: a infraestrutura é composta por duas ou mais nuvens (pública, comunitária ou privada), unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações.

Mesmo com todo o potencial ofertado, as nuvens não estão conseguindo suprir o aumento exponencial da quantidade de dados produzidos, pois os seus recursos são limitados

e os dados continuam crescendo. Surgiu daí a ideia das federações de nuvens, que buscam agregar recursos de outras nuvens, formando uma arquitetura, que dá ao usuário a impressão de que são ilimitados.

Buscando tomar proveito da diversidade dos serviços e agregar recursos de outras nuvens, surgem as federações de nuvens computacionais, cujos conceitos básicos serão apresentados na seção Seção 2.2.

2.2 Federação de Nuvens Computacionais

Conforme apresentado nas seções anteriores, há variação quanto aos serviços prestados por uma nuvem. Isto pode estar relacionado à infraestrutura, à plataforma e aos softwares disponíveis. Independentemente das configurações utilizadas, as nuvens podem ser mantidas por organizações que prestam este serviço e que são grandes empresas reconhecidas no mercado, como também podem ser nuvens menores utilizadas internamente em algumas organizações. Com essas alternativas, surge a possibilidade de utilização de nuvens federadas, que procuram otimizar os recursos disponíveis entre as diversas organizações parceiras.

Nuvens federadas podem ser definidas como um conjunto de provedores de computação em nuvem pública e privada, conectados por meio da Internet [12, 14], as quais também são chamadas de *inter-cloud* ou *cross-cloud* [14]. Dentre seus objetivos, pode-se destacar a ilusão de recursos ilimitados, a eliminação da exclusividade no uso de um provedor de infraestrutura específico e a otimização quanto ao uso dos recursos dos provedores federados.

Assim, a federação permite que cada fornecedor aumente sua capacidade de processamento e de armazenamento, solicitando mais recursos para outras nuvens na federação. Isso significa que um provedor de nuvem local é capaz de satisfazer as solicitações do usuário além de sua capacidade, já que os recursos ociosos de outros fornecedores podem ser utilizados. Além disso, na ocorrência de falha em um dos fornecedores, os recursos podem ser requisitados a outro, proporcionando maior tolerância a falhas [9, 21].

Devido às diferentes características dos provedores participantes da federação, a implementação deste modelo não é trivial, sendo necessário o atendimento de alguns requisitos, conforme apresentado por Celesti *et al.* [14]:

- Automação e escabilidade: uma nuvem, utilizando-se de mecanismos de descoberta, deve ser capaz de selecionar as nuvens que satisfaçam às suas necessidades e reagindo às mudanças ocorridas de forma automática e transparente;

- Segurança interoperável: é necessário integrar diversas tecnologias de segurança, permitindo assim que uma nuvem possa se federar sem a necessidade de alterar suas políticas de segurança.

Buyya *et al.* [12] apresentam as necessidades para explorar a infraestrura das federações de nuvems computacionais, as quais são:

- Prever a carga das aplicações: é importante que os sistemas estejam aptos a prever a carga e os comportamentos dos serviços armazenados, assim, é possível tomar decisões para dinamicamente aumentar ou diminuir os recursos da infraestrutura da federação;
- Mapeamento flexível dos serviços para os recursos: com o aumento dos custos de operação é importante maximizar a eficiência, o custo-benefício e a utilização da nuvem;
- Técnicas de otimização: buscar a melhor combinação entre o serviço e o provedor para atender as demandas;
- Integração e interoperabilidade: é necessário que a federação integre diferentes tecnologias, sejam elas de armazenamento de arquivos, autenticação, entre outros, sem que a nuvem membro tenha que alterar sua forma de trabalho e gestão;
- Monitoramento da escalabilidade dos componentes do sistema: é necessário que o sistema gerencie os diversos componentes da federação, sem perder a escalabilidade e o desempenho, para assim, atender as requisições.

Depois dos conceitos apresentados, é necessário entender como o armazenamento dos dados ocorre na plataforma de nuvem.

2.3 Armazenamento em Nuvem Computacional

O armazenamento de dados em nuvem computacional deve buscar garantir a disponibilidade da informação através da tolerância a falhas. Esse processo pode ser realizado de diversas maneiras, como a replicação de dados entre as nuvens, a replicação entre fornecedores diferentes ou sistemas de arquivos distribuídos. Além disso, fornecer *interfaces* genéricas para o armazenamento e a recuperação dos dados favorece a utilização do recurso. Outro aspecto que deve ser considerado é o custo-benefício do provedor selecionado para a operação.

Pesquisas têm trabalhado no projeto e na implementação de infraestruturas genéricas para serviços de armazenamento [22, 23, 24]. Muitos fornecedores de serviços de armazenamento precisam de um forte trabalho de integração devido às interfaces distintas e aos conjuntos específicos de recursos oferecidos por cada fornecedor [25, 26]. Particularmente, em um ambiente de nuvens federadas, a execução de aplicações envolve o armazenamento e a transferência de arquivos entre as nuvens integrantes da federação.

Ruiz-Alvarez e Humphrey [27] desenvolveram um mecanismo automático para selecionar o gerenciamento de armazenamento. As principais contribuições deste trabalho foram um esquema XML para descrever os provedores de nuvem e os respectivos recursos de armazenamento; e um algoritmo para seleção de um serviço de armazenamento com base em critérios fornecidos pelos usuários, por exemplo, o custo de transferência de dados, a durabilidade e o versionamento de dados. Uma característica fundamental de seu mecanismo é a possibilidade de acessar as estatísticas dos provedores de armazenamento em tempo real, de forma que os algoritmos possam usar a informação atual para decidir o melhor local para armazenar os dados.

Outro aspecto importante é apresentado em Bermbach *et al.* [28], que relata que a dependência de um serviço único de armazenamento proporciona limitações quanto à disponibilidade e a escalabilidade em relação ao fornecedor selecionado, além de poder causar atrasos nas execuções realizadas pelos demandantes do serviço. Para minimizar este problema, os autores propõem o MetaStorage, um sistema de armazenamento federado utilizando tabelas *hash*, que pode integrar diversos fornecedores por meio da replicação dos dados. Nessa arquitetura, os agentes, que atuam em uma camada acima dos provedores de armazenamento, realizam o armazenamento e a recuperação de dados através dos nós que estão situados na camada mais baixa, e fornecem a *interface* genérica a fim de abstrair os detalhes técnicos da infrestrutura subjacente.

Devido a grande quantidade de dados, uma das preocupações recorrentes nas pesquisas envolvendo armazenamento em nuvem é a compressão dos dados. Nesse cenário, Bogdan [21] apresenta o BlobSeer, que é um modelo para o gerenciamento de dados distribuídos, especialmente implementado para ler, escrever e agrupar grande quantidade de informações em larga escala, trabalhando de forma transparente para realizar a compressão dos dados. Esta abordagem propicia a diminuição do espaço necessário em disco, bem como, da utilização de banda para a transferência dos dados.

Lin *et al.* [29] tratam do problema da preservação de banco de dados genéticos, o qual gera abundância de informações e dificuldades de gerenciamento, também, devido ao compartilhamento das mesmas por diversas equipes. Neste contexto, os autores utilizam um sistema gerenciador de banco de dados em nuvem, HBASE [30], que integra informação de diversos *hosts* e pesquisadores locais. A implementação da estrutura de

armazenamento, entretanto, fica sob responsabilidade dos provedores do serviço.

Fetai e Schuldt [31] usam a replicação para explorar as vantagens da nuvem. O SO-1SR, Self-Optimizing One-Copy Serializability Protocol (auto-otimização 1SR) utiliza os recursos existentes de forma adaptativa e dinâmica para replicar dados entre os nós de nuvem. A estratégia utilizada é a diminuição da espera por meio de técnicas de otimização nas diferentes fases do ciclo de vida da transação, sendo que uma transação consiste de uma operação de acesso aos objetos por meio de seu identificador. O SO-1SR mantém um catálogo para gerenciar as informações sobre as cópias disponíveis.

Yuan *et al.* [32] propuseram um algoritmo para selecionar uma nuvem como provedora de armazenamento de dados, utilizando uma estratégia de grupo com base em uma estratégia *K-means* [33]. A estratégia contém dois algoritmos que agrupam os provedores existentes em k conjuntos. Neste modelo é gerada uma matriz baseada em k *datacenters* durante a fase de criação do *workflow*, e dinamicamente agrupa os conjuntos de dados recém gerados para os *datacenters* mais apropriados durante a fase de execução.

Stockinger *et al.* [34], em sua proposta de modelo de custo para distribuição e replicação de dados, apresentaram alguns aspectos, destacando-se entre eles a localização física, a largura de banda, o tamanho do arquivo que está sendo transferido e o protocolo de transmissão. Cada um destes aspectos pode influenciar a transferência, assim sendo, preferencialmente, os dados são armazenados em redes LAN que estão mais próximas do destino e, possivelmente, possuem melhor velocidade de transmissão. Apesar do trabalho destes autores estar relacionado ao custo, os aspectos apresentados são também considerados, pois tem por objetivo realizar a operação no menor tempo possível, o que proporcionaria menor custo.

Outros autores, em propostas de arquiteturas e sistemas, apresentaram o armazenamento de dados em nuvem como parte de um conjunto de funcionalidades. Na maioria das vezes, esse serviço é delegado a provedores específicos por tratar estas operações, ou seja, os sistemas utilizam as funcionalidades disponibilizadas pelos provedores.

Zhang *et al.* [35] propuseram uma arquitetura baseada em quatro camadas (acesso, aplicações, gerenciamento de infraestrutura e armazenamento), por meio de comunicação P2P. Na camada de acesso é feito o controle dos usuários que têm permissão para acessar as aplicações para diferentes finalidades. As aplicações por sua vez utilizam dos recursos disponibilizados pela camada de gerenciamento, que é responsável por gerenciar todos os recursos disponíveis e selecionar o melhor entre eles para o armazenamento dos dados, bem como realizar a compressão e a redundância da informação. Assim, esta camada abstrai as particularidades de cada um dos provedores de modo que o usuário e/ou a aplicação possa acessar um nível mais abstrato do sistema de armazenamento. A camada de armazenamento controla o processo a ser executado. A atualização e a recuperação,

entre outros, também devem garantir a disponibilidade da informação.

Zeng *et al.* [36] propuseram uma arquitetura em camadas (infraestrutura de rede e armazenamento, gerenciamento de armazenamento, gerenciamento de metadados, sobreposição de armazenamento e *interface*) para implementar um serviço de armazenamento de dados em nuvem. Os serviços de virtualização e recuperação atuam para conectar os dispositivos de armazenamento e expor de forma simplificada e padronizada as estrutura de dados para a camada de *interface*.

Kavalionak e Montresor [37] analisam a relação entre redes P2P e a computação em nuvem, e sua utilização conjunta por meio do uso de recursos ociosos dos usuários em contrapartida à redução dos custos do provedor. Neste experimento, os autores implementam uma rede híbrida P2P/nuvem, onde os participantes da nuvem são *peers*. O algoritmo apresentado trata de um dos grandes problemas da comunicação P2P que é garantir a confiabilidade e a disponibilidade dos dados. A implementação é capaz de auto-regular a quantidade de nuvens utilizadas, garantindo desta forma o nível mínimo de serviço solicitado pelo usuário. Esta abordagem chamada de SLA (*Service-Level Agreement*) é uma parte do serviço contratado que busca garantir que as necessidades mínimas do cliente sejam atendidas.

Por sua vez, a execução de *workflows* de bioinformática normalmente deve manipular grande volume de dados e necessita de grande poder de processamento, o que justificaria a utilização das nuvens computacionais. Apesar disso, as propostas de armazenamento existentes na literatura não consideram as características específicas dos arquivos da bioinformática.

Entre os sistemas de armazenamento encontrados na literatura estão o *Hadoop Distributed File System* [38, 39], o CassandraFS [40], o XtreemFS [41], o Lustre [42, 43] e o Ceph [44]. Estes sistemas de armazenamento, apesar de eficientes, necessitam que uma política de armazenamento para dados biológicos seja estabelecida, a fim de proporcionar maior eficiência em sua manipulação, onde facilmente atinge-se *gigabytes* de armazenamento em apenas um projeto de sequenciamento. Além disso, a utilização dos recursos de armazenamento pode estar condicionada à limitação da largura de banda e do tamanho dos arquivos que devem trafegar. Desta forma, é importante analisar os aspectos que podem influenciar o tempo de resposta do sistema.

Hiden *et al.* [45] apresentam um modelo onde é possível aos cientistas executarem *workflows* e compartilharem resultados com outros pesquisadores. A aplicação pode ser executada em nuvens públicas e privadas. O módulo de armazenamento abstrai as operações nos diversos provedores suportados, entre eles Amazon S3 [46], Azure Blob Store [18] e armazenamento local. Os dados são versionados, o que possibilita que os usuários trabalhem com as versões antigas da informação. Além disso, os dados podem ser agrupados

e compartilhados exclusivamente entre membros do grupo.

Na Tabela 2.1 é possível consultar um resumo dos trabalhos aqui citados, o ano de sua publicação e as principais características.

Tabela 2.1: Tabela Resumo.

Autor	Data (Ano)	Características
Ruiz-Alvarez e Humphrey	2011	Esquema XML para descrever os provedores. O usuário seleciona o critério desejado e através de estatísticas dos provedores de armazenamento o sistema realiza a seleçãof;
Bermbach *et al.*	2011	Armazenamento federado utilizando tabelas hash, atuando na camada acima dos provedores de armazenamento, integrando-os
Bogdan	2010	Gerenciamento dos dados distribuídos e compressão
Lin *et al.*	2012	Preservação dos dados usando HBASE e abstração do sistema de armazenamento
Fetai e Schuldt	2013	Replicação de dados entre os nós
Yuan *et al.*	2010	Agrupamento do conjunto de dados para os provedores selecionados
Stockinger *et al.*	2001	Proposta de modelo de custo para distribuição e replicação de dados através de critérios (Localização física, largura de banda, tamanho do arquivo, protocolo de transmissão)
Zhang *et al.*	2011	Arquitetura em camadas, onde a camada de armazenamento gerencia os dados e faz a compressão
Zeng *et al.*	2009	Abstração do serviço de armazenamento para as aplicações usuárias
Kavalionak e Montresor	2012	Replicação dos dados para garantir a disponibilidade
Hiden *et al.*	2013	Execução de workflows em nuvens federadas, Abstração dos serviços de armazenament, Versionamento dos dados;

2.4 Considerações Finais

Neste capítulo, diversas pesquisas foram apresentadas, muitas delas utilizam os sistemas de armazenamento de terceiros para realizar as operações fundamentais. Em vários casos, há a abstração dos processos de armazenamento e de recuperação. Em alguns casos, como apresentado por [21], há a compressão dos dados, entretanto, sua abordagem está focada na redução do espaço utilizado para o armazenamento. A replicação dos

dados é outro critério muito comum entre as pesquisas para solucionar o problema da disponibilidade.

Notamos que as características dos dados biológicos não são utilizadas para determinar a forma de armazenamento ou as estratégias para isso. Em alguns casos são apresentados trabalhos relacionados aos *workflows* de bioinformática.

No capítulo seguinte é apresentado o BioNimbuZ, detalhando sua arquitetura e seus componentes. Além disso, será possível visualizar o impacto da transferência de informações para a execução do *workflow*.

Capítulo 3

BioNimbuZ

Neste capítulo, é apresentado o BionimbuZ, uma arquitetura para federação de nuvens computacionais, que permite a execução de *workflows* de bioinformática por meio do compartilhamento dos recursos entre os integrantes da federação, fornecendo a ilusão de que os recursos computacionais disponíveis são ilimitados, ou seja, as demandas do usuário sempre serão atendidas.

3.1 Visão Geral

O BioNimbuZ [1, 4] é uma arquitetura para federação de nuvens computacionais, que permite a utilização de diversas ferramentas de bioinformática oferecidas como serviço, de maneira transparente e tolerante a falhas, oferecendo a ilusão de recursos computacionais ilimitados. Assim, os serviços podem ser disponibilizados conjuntamente, mantendo as caracterísitcas e políticas internas de cada nuvem integrante da federação.

A arquitetura BioNimbuZ está divida em módulos (Figura 3.1) que possuem funcionaliades específicas, permitindo simplicidade e eficiência quando novos provedores de nuvens desejam ser incluídos na federação.

Para possibilitar a integração de um provedor de nuvem computacional na federação é utilizado um *plugin* de integracão que realiza a comunicação entre o provedor e os serviços controladores. Além disso, o *plugin* é responsável por coletar informações da nuvem da qual ele é responsável, como recursos computacionais e ferramentas de bioinformática que estão disponíveis neste provedor. O núcleo da arquitetura BioNimbuZ é dividido em serviços, os quais são [1]: Descobrimento, Escalonamento, Armazenamento, Controlador de Tarefas, Tolerância a Falhas, Monitoramento, Segurança e Controlador de Qualidade de Serviço. Estes serviços são mostrados na Figura 3.1.

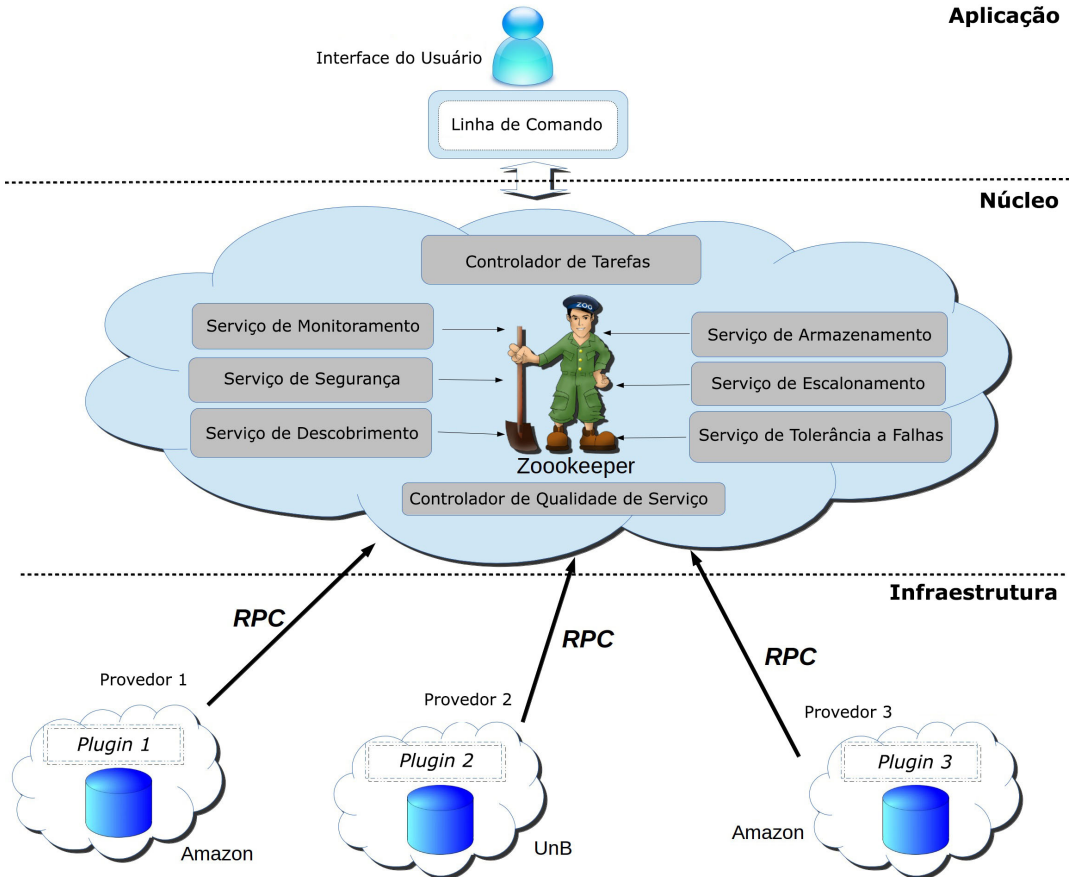

Figura 3.1: Arquitetura BioNimbuZ, adaptado de [4].

3.1.1 Serviço de Descobrimento

Responsável por identificar os provedores de serviços que fazem parte da federação, além de coletar as informações particulares das nuvens, como: latência de rede, capacidade de processamento e armazenamento, ferramentas de bioinformática disponíveis, parâmetros de execução, arquivos de entrada e saída presente nesta nuvem.

Para realizar essa operação, o Serviço de Descobrimento envia mensagem aos integrantes da federação, os quais respondem com as informações existentes. Estas informações são mantidas pelo Serviço de Descobrimento. Assim, constantemente, o serviço envia mensagens aos integrantes da federação para verificar se eles estão presentes, e caso não obtenha resposta, remove o provedor da federação.

3.1.2 Serviço de Escalonamento

Coordena a execução das tarefas, através do recebimento de pedidos, distribuindo dinamicamente as execuções entre os provedores que possuem as ferramentas necessárias para o trabalho.

Este serviço mantém as informações referentes a execução de cada trabalho. Além disso, o escalonador verifica a carga de trabalho dos provedores a fim de evitar a sobrecarga dos mesmos.

Os pedidos de execução são submetidos a política de escalonamento para decidir para qual provedor a tarefa será enviada. Neste momento, o escalonador se comunicará com o Serviço de Escalonamento para obter as informações atualizadas de cada um dos provedores candidatos à execução e escolher, segundo seus critérios, o mais adequado.

3.1.3 Serviço de Armazenamento

Coordena a estratégia de armazenamento de arquivos produzidos e/ou que serão consumidos pelas ferramentas executadas. Entre suas atividades está a distribuição e a replicação dos arquivos entre os provedores da federação.

Dessa forma, quando necessário, o Serviço de Armazenamento solicitará ao Serviço de Descobrimento informações sobre os provedores, para que, baseado nessas informações e utilizando a política de armazenamento, faça a decisão sobre o local para onde o arquivo será enviado ou de onde o arquivo será solicitado.

Este trabalho propõe uma nova política de armazenamento, a qual será detalhada no Capítulo 4.

3.1.4 Controlador de Tarefas

É o serviço responsável por fazer a ligação entre a camada de núcleo da arquitetura e a camada de aplicações. Assim, ele tem a função de gerenciar os pedidos dos usuários, de forma a fazer o controle por usuários, e manter os resultados para posterior consulta.

3.1.5 Serviço de Tolerância a Falhas

Acompanha os provedores da federação na execução dos serviços demandados. Assim, é possível recuperar os serviços ou requisições de execução de ferramentas em caso de falha.

Ao detectar a falha, o Serviço de Tolerância a Falhas solicitará ao Serviço de Escalonamento o reinício das tarefas que estavam em execução no servidor que falhou em outro provedor.

3.1.6 Serviço de Monitoramento

Registra, monitora e armazena dados sobre carga das nuvens da federação e sobre histórico de execução de aplicações.

Quando uma requisição de execução de aplicação é feita pelo Controlador de Tarefas, o Serviço de Monitoramento verifica a disponibilidade da ferramenta necessária na federação, e repassa o pedido ao Serviço de Escalonamento.

Além disso, o Serviço de Monitoramento monitora a federação a fim de que todos os pedidos sejam atendidos e executados com sucesso.

3.1.7 Serviço de Segurança

Gerencia a política de acesso à federação, como as credenciais de acesso dos usuários a cada uma das nuvens. Este serviço cria contextos entre usuários e provedores de serviços atendendo aos requisitos de autenticação, autorização e confidencialidade.

3.1.8 Controlador de Qualidade de Serviço

Responsável por investigar se os requisitos, na solicitação realizada pelo cliente, podem ser atendidos pela federação de nuvens.

Além dos serviços indicados acima, a arquitetura definida por Saldanha [1] também especificou algumas classes de informações que são trocadas entre eles. As classes de informações são:

- *PeerInfo*: informações referentes a um servidor, que é um membro da federação. Nessa classe são armazenadas informações, tais como o endereço de rede, a latência de rede e o tempo que está conectado;
- *PluginInfo*: informações referentes a um *plugin* e o provedor referenciado por ele. Nessa classe são armazenadas todas as informações que serão compartilhadas com outros servidores como a quantidade total de CPUs, a quantidade de CPUs livres, a capacidade de armazenamento e o espaço de armazenamento livre;
- *ServiceInfo*: informações referentes a um serviço oferecido por um provedor que faz parte da federação. Essa classe armazena informações como o nome da ferramenta, os parâmetros de configuração, os arquivos de entrada e os arquivos de saída;
- *JobInfo*: informações referentes a um trabalho, ou seja, uma requisição de execução de um serviço na federação. Nessa classe são armazenadas informações como o identificador do serviço solicitado e os parâmetros para execução do serviço;
- *TaskInfo*: informações referentes a uma tarefa, ou seja, uma instância de um trabalho. Essa classe representa uma execução de um trabalho em um determinado *plugin*. Nessa classe são armazenadas informações, como o *JobInfo*, que possui dados

sobre o trabalho a ser executado, e o *PluginInfo* com informações do local em que será executado. Além disso, essa classe armazena o estado de execução da tarefa;

- *FileInfo*: informações referentes a um arquivo armazenado na federação. Nessa classe são armazenadas informações como o nome e o tamanho do arquivo;
- *PluginFileInfo*: informações referentes a uma instância de um arquivo que pode estar armazenado em vários provedores da federação. Essa classe armazena informações como o *FileInfo* do arquivo, e o *PluginInfo* do local onde o arquivo está armazenado. Essas classes armazenam informações essenciais para a execução do BioNimbuZ, e são utilizadas na comunicação entre os componentes da arquitetura.

As informações armazenadas por essas classes são utilizadas na comunicação entre os componentes da arquitetura, além de serem fundamentais para a execução do BioNimbuZ. Para esta comunicação é utilizado o *framework* Apache ZooKeeper, apresentado na próxima seção.

3.2 Apache ZooKeeper

O ZooKeeper [5] é um serviço de coordenação de alto desempenho para aplicações distribuídas, *opensource*, que através da exposição de serviços comuns como nomeação, gerenciamento de configuração, sincronização e serviços de grupo, são utilizados pelas aplicações sem a necessidade destas implementarem essas funcionalidades.

As aplicações distribuídas, em geral, ao se comunicarem, possibilitam a ocorrência de diversos tipos de falhas:

- Falha Parcial: a rede pode falhar durante o processo e o remetente não consegue entender se o receptor recebeu ou não a mensagem;
- Condição de Corrida: problema relacionado ao gerenciamento da concorrência entre processos simultâneos.

O ZooKeeper fornece ferramentas para que a construção de aplicações distribuídas possam lidar com problemas de falhas de maneira transparente.

Podem existir diversos servidores ZooKeeper que se comunicam para atender às demandas. Enquanto a maioria dos servidores ZooKeeper estiverem disponíveis, o serviço ZooKeeper estará disponível.

3.2.1 Funcionamento

O ZooKeeper é projetado para ser fácil de programar, onde os processos distribuídos se coordenam por meio de espaços de nomes hierárquicos, organizados de forma semelhante a um sistema de arquivos padrão.

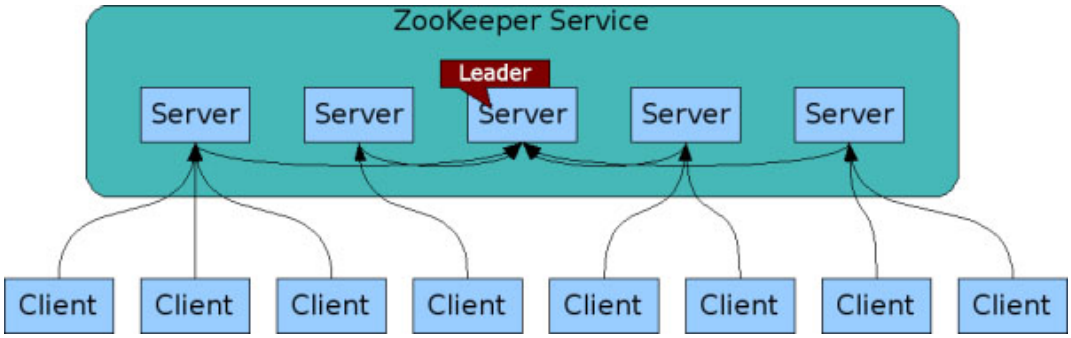

Figura 3.2: Serviço ZooKeeper [5].

O serviço ZooKeeper (mostrado na Figura 3.2) é replicado sobre um conjunto de máquinas onde cada servidor armazena uma cópia dos dados em memória. Entre os servidores, é eleito um líder e os outros são denominados seguidores. No *startup*, ou quando o líder se desconectar, um algoritmo de eleição é executado para definição do novo líder entre os servidores ativos [5].

Os clientes conectam-se a um único servidor ZooKeeper e mantêm uma conexão TCP através da qual enviam pedidos e recebem respostas. A escrita passa sempre pelo líder. Se a conexão for perdida o cliente conecta-se a um servidor diferente.

Assim, todas as transações são numeradas para garantir a ordem entre elas. Operações subsequentes podem utilizar esta ordenação para implementar abstrações de alto nível, como as sincronizações primitivas.

3.2.2 Modelo de Dados e Espaço de Nomes Hierárquicos

O espaço de nomes hierárquicos consiste em um registro de dados onde cada nó é chamado de *ZNode*. Cada *ZNode* tem dados armazenados (array de bytes) em memória, proporcionando baixa latência e alta taxa de transferência. Além disso, podem opcionalmente ter filhos, ou seja, nós abaixo deles.

Um nome é uma sequência de elementos de caminho separados por uma barra (/). Cada nó no espaço nome do ZooKeeper é identificado por um caminho, conforme mostrado na Figura 3.3.

Cada *ZNode* mantém uma estrutura de estatísticas que inclui o número de versão para alterações de dados, mudanças de controle de acesso (ACL) e *timestamps* que permitem validações de *cache* e atualizações coordenadas. Cada *ZNode* armazena no máximo 1Mb

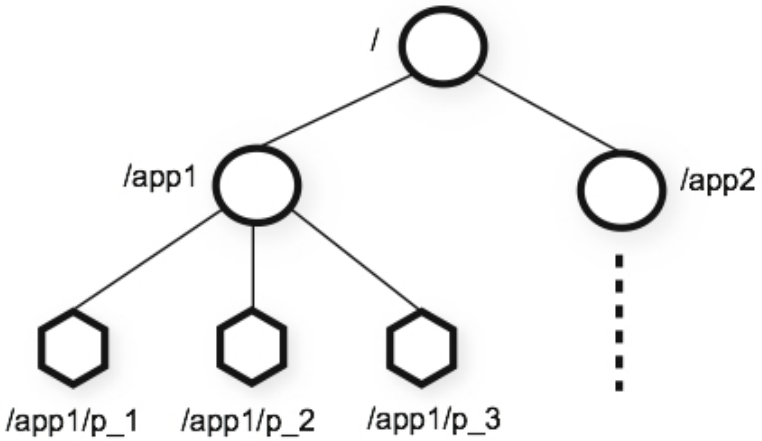

Figura 3.3: Hierarquia de Nomes [5].

de dados. Cada vez que há alterações de dados de um *ZNode*, o número da versão é incrementado. Além disso, sempre que um cliente recupera dados, ele recebe a versão destes. Os dados armazenados em cada *ZNode* em um *namespace* são lidos e escritos atomicamente. Podem existir dois tipos de *ZNode*:

- *ZNodes* Persistentes: existem até que sejam excluídos explicitamente;
- *ZNodes* Efêmeros: existem enquanto a sessão estiver ativa, não podem ter filhos.

3.2.3 Operações

Um dos objetivos do ZooKeeper é fornecer uma interface de programação muito simples. Como resultado, ele suporta apenas as seguintes operações:

- *create*: cria um *ZNode* numa localização na árvore do ZooKeeper;
- *delete*: realiza a exclusão de um *ZNode* no ZooKeeper;
- *exists*: testa se existe um *ZNode* em um determinado local;
- *get data*: faz a leitura dos dados a partir de um *ZNode*. Solicitações de leitura são processadas localmente no servidor ZooKeeper ao qual o cliente está conectado (Figura 3.4);
- *set data*: faz a escrita de dados em um *ZNode*. Solicitações de escrita são encaminhadas para o líder e passa por consenso majoritário, antes que uma resposta seja gerada (Figura 3.4);
- *get children*: recupera a lista de filhos de um *ZNode*;
- *sync*: espera que os dados sejam propagados.

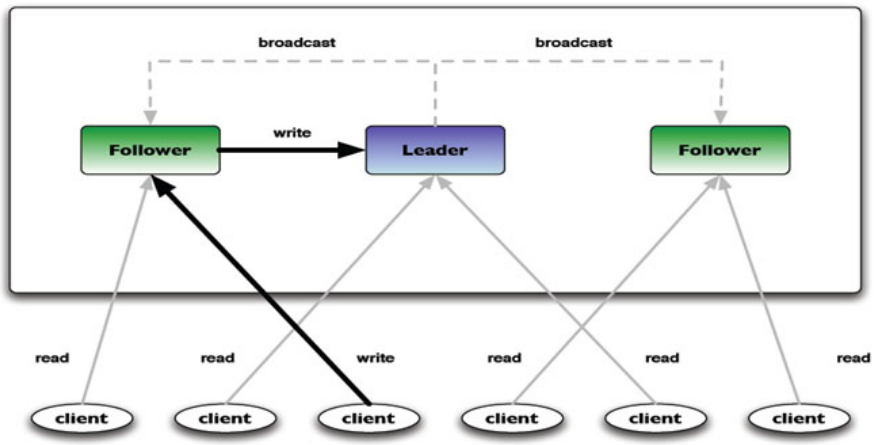

Figura 3.4: Operações de Leitura e Escrita [5].

A Figura 3.4 mostra como são realizadas as operações de leitura e escrita de informações no ZooKeeper. Na operação de escrita, o cliente se conecta a um dos servidores ZooKeeper, que encaminha a solicitação de escrita ao líder, que após a gravação, envia uma mensagem em *broadcast* aos demais servidores ZooKeeper para atualização de seus registros. Na operação de leitura o cliente obtém uma cópia da informação de um dos servidores.

3.2.4 *Watchers*

Em determinados momentos é necessário que o cliente seja informado quando houver alterações no estado do *ZNode*. Para isso, o ZooKeeper possui os *watchers* que serão acionados quando houver alterações na estrutura de arquivos, bem como, na perda da conexão. Quando isso ocorrer, o cliente receberá um pacote dizendo que o *ZNode* mudou. O cliente deverá definir os *watchers*, e somente este receberá o aviso da mudança ou remoção. Os gatilhos disponíveis são:

- *NodeChildrenChanged*: alteração em nó filho;
- *NodeCreated*: *ZNode* for criado;
- *NodeDataChanged*: dados do *ZNode* foram alterados;
- *NodeDeleted*: perda de conexão com um *ZNode*.

Depois de serem utilizados, os *watchers* deixam de existir, sendo necessária a criação de um novo.

3.2.5 Garantias

Apesar de simples e rápido, o ZooKeeper possibilita a construção de serviços mais complexos, tais como sincronização. Para isso, ele fornece um conjunto de garantias que são:

- Consistência Sequencial: atualizações de um cliente serão aplicadas na ordem em que foram enviadas;
- Atomicidade: não existem resultados parciais, atualizações retornam sucesso ou fracasso;
- Sistema Único de Imagem: o ponto de vista do cliente sobre serviço é sempre o mesmo independentemente do servidor que ele estiver conectado;
- Confiabilidade: uma vez que uma atualização foi aplicada, ele irá persistir a partir daquele momento, até que um cliente substitua a atualização;
- Consistência Eventual: a visão dos clientes do sistema é a garantia de ser *up-to-date* dentro de um determinado período de tempo limitado.

3.2.6 Integração com o BioNimbuZ

O ZooKeeper está integrado ao BioNimbuZ auxiliando na realização de diversas tarefas. Quando uma nuvem é incluída na federação, a mesma deve se registrar através da porta 2181 ao serviço ZooKeeper para que esse crie um *ZNode* com seus dados. Neste local serão armazenados, de forma centralizada, as informações referentes a arquivos disponíveis, serviços disponíveis, entre outros. Assim sendo, não é necessário acessar o provedor para obter tais informações.

Quando alguma informação é alterada ou excluída de um provedor, os demais integrantes da federação são informados por meio da utilização de *watchers* do ZooKeeper, semelhante a uma mensagem em *broadcast*. No processo de inclusão de dados, o BioNimbuZ cria um *ZNode*, chamando *pending saves* no ZooKeeper para manter essas informações, criando assim, uma lista de arquivos que deverão ser armazenados. Com base nesta lista, a política de armazenamento irá selecionar a melhor nuvem para receber cada arquivo. Depois de concluído o armazenamento, o *StorageService* verificará se o arquivo foi armazenado, e em caso de sucesso, removerá da lista o arquivo.

A troca de mensagens entre os provedores é realizada utilizando-se chamadas RPC (Chamada Remotas de Procedimentos) [47], utilizando para isso o Apache Avro.

3.3 Apache Avro

Apache Avro [48] é um sistema de serialização de dados, para isso o *framework* fornece:

- Rica estrutura de dados;
- Formato compacto, rápido e com dados binários;
- Um *container* de arquivos para armazenar dados persistentes;
- Chamada de procedimento remoto (RPC);
- Integração simples com linguagens dinâmicas, não sendo necessária a geração de código para leitura e escrita de estrutura de dados.

3.3.1 Esquemas

Quando há leitura de dados pelo Avro, o esquema utilizado ao escrever está sempre presente, permitindo que cada dado a ser escrito não tenha gastos com tempo extra, tornando a serialização rápida e pequena. Isso também facilita o uso com linguagens de *script* dinâmicas, pois os dados, juntamente com o seu esquema, são totalmente auto-descritivos.

Quando os dados do Avro são armazenados em um arquivo, seu esquema é armazenado com ele, para que os arquivos possam ser processados, posteriormente, por qualquer programa. Se o programa de leitura dos dados espera um esquema diferente, isso pode ser facilmente resolvido, uma vez que ambos os esquemas estão presentes.

Na utilização de RPC, o cliente e o servidor trocam esquemas no estabelecimento da conexão. Uma vez que tanto o cliente quanto o servidor possuem o esquema do outro, a correspondência entre os mesmos campos nomeados, campos em falta, campos extras, entre outros, podem ser facilmente resolvidos. Assim, os esquemas são definidos com JSON [49]. Isso facilita a implementação em linguagens que já possuem bibliotecas JSON.

3.3.2 Diferenciais do Avro

Comparado a outros sistemas, o Avro oferece funcionalidade semelhante para sistemas como o Thrift [50], Protocol Buffers [51], etc. Entretanto, Avro difere destes sistemas nos seguintes aspectos:

- Tipagem Dinâmica: o Avro não requer que o código seja gerado. Os dados são sempre acompanhados por um esquema que permite o processamento completo de dados sem a necessidade de geração de código, tipos de dados estáticos, etc. Isso facilita a construção de sistemas de processamento de dados genéricos;

- Dados sem etiqueta: desde que o esquema esteja presente quando os dados forem lidos, assim, menos tipos de informações necessitam ser codificados com os dados, resultando em uma serialização menor;
- Atribuição automática de ID's de campos: quando o esquema é alterado, tanto o antigo quanto o novo esquema estão sempre presentes durante o processamento de dados, portanto, as diferenças podem ser resolvidas simbolicamente, usando nomes de campo.

3.3.3 Integração com o BioNimbuZ

As chamadas RPC utilizam o Apache Avro para uma comunicação, de forma mais simples e robusta, entretanto o Avro não consegue garantir a correta execução da chamada em caso de algum problema na rede no momento que uma chamada é realizada.

3.4 Considerações Finais

Neste capítulo foi descrito o BioNimbuZ, uma arquitetura de federação em nuvens para execução de *workflows* de bioinformática, através da distribuição de tarefas e informações entre as diversas nuvens integrantes da federação.

No próximo capítulo será apresentada a política de armazenamento proposta neste trabalho, para otimizar a execução das tarefas que compõem o *workflow*.

Capítulo 4

A Política de Armazenamento

Neste capítulo é apresentada a política de armazenamento proposta para armazenamento de arquivos com dados genômicos em nuvens federadas. Primeiramente, na Seção 4.1 são descritos os aspectos principais referente à política de armazenamento de dados. Posteriormente, a Seção 4.2 apresenta a política proposta neste trabalho, os critérios e os processos realizados. Na Seção 4.3, é mostrada a forma como a política de armazenamento proposta foi implementada e integrada ao BioNimbuZ. Ao final, na Seção 4.4, são feitas as considerações finais referentes à política de armazenamento proposta.

4.1 Visão Geral

O Serviço de Armazenamento é o serviço responsável por coordenar o fornecimento dos arquivos necessários (arquivos de entrada) para a execução de tarefas que fazem parte de um *workflow* e armazenamento dos dados provenientes dessa execução (arquivos de saída). Para que isso aconteça, é necessário que exista uma política de armazenamento definida.

A política de armazenamento de dados é um conjunto de procedimentos que são implementados para controlar e gerenciar os dados dentro de uma organização ou sistema de informação. Esses procedimentos podem variar de acordo com cada política estabelecida, mas devem determinar, para um conjunto de aplicações, como, quando e onde os dados serão coletados e armazenados.

Nas aplicações de bioinformática, devido ao volume de informações, a administração dos dados de forma eficiente é fundamental para otimizar a execução do *workflow*. Esta proposta de armazenamento foi realizada para auxiliar na execução eficiente do *workflow*.

4.2 Política Proposta

A política de armazenamento proposta tem como objetivo escolher as melhores nuvens para armazenar os arquivos, ou selecionar a origem de onde o arquivo será carregado, para completar a execução da tarefa em menor tempo. Para a política proposta, foram considerados alguns critérios, tomando por base Stockinger et al. [34].

Assim, a política de armazenamento proposta fornece três funcionalidades: armazenamento dos arquivos (*upload*), contendo os arquivos de entrada ou o resultado da execução de uma das etapas do *workflow*; recuperação dos arquivos (*download*), necessários para a execução da tarefa de bioinformática, a fim de gerar os arquivos de saída; e, a replicação dos arquivos, a fim de garantir tolerância a falhas. É importante ressaltar que a replicação utiliza a funcionalidade de armazenamento dos arquivos para definir onde a réplica será armazenada.

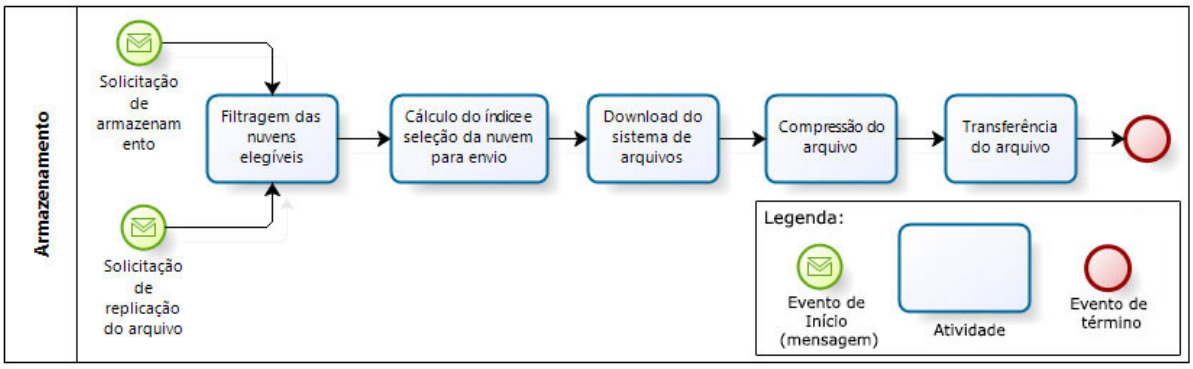

Figura 4.1: Fluxo de *Upload* de Dados.

O fluxo de armazenamento de dados (visto na Figura 4.1) é composto de cinco etapas, as quais são: Filtragem das nuvens elegíveis, Cálculo do índice e seleção da nuvem para o envio, *Download* do sistema de arquivos, Compressão do arquivo e a Transferência do arquivos. Como pode ser observado na Figura 4.1, este processo tem dois tipos de início: o primeiro é a solicitação de armazenamento, que pode ocorrer por demanda do usuário a fim de armazenar um novo arquivo na federação ou um arquivo de saída de uma tarefa executada; o segundo é a solicitação de replicação de um arquivo após ser armazenado. Ambos os processos são controlados pelo Serviço de Armazenamento. Este processo visa selecionar a nuvem que pode receber o arquivo com melhores condições (desempenho e confiabilidade). Cada uma dessas etapas será detalhada a seguir.

O fluxo de recuperação de dados (visto na Figura 4.2), ou seja, o *download* de um arquivo já existente na federação, é composto por quatro etapas: Cálculo do índice e seleção da nuvem para o envio, *Download* do sistema de arquivos, Compressão do arquivo e a Transferência do arquivo. Esse processo busca selecionar a nuvem que poderá responder

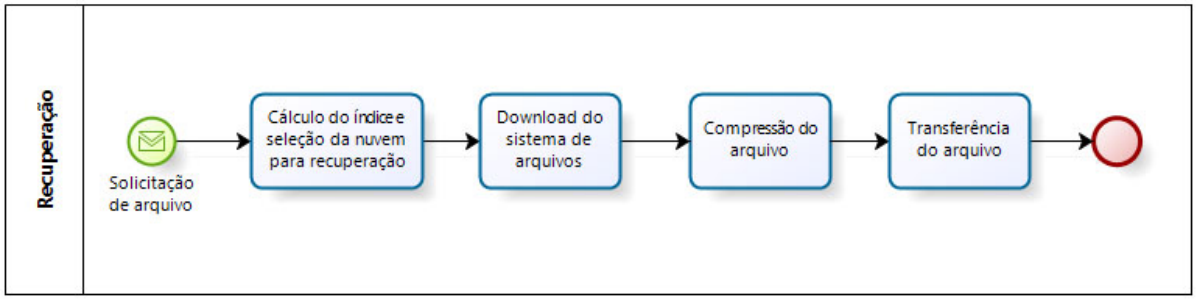

Figura 4.2: Fluxo de *Download* de Dados.

à solicitação de recuperação de um arquivo e realizar a transferência do arquivo no menor tempo possível.

Ambos os processos, Armazenamento e Recuperação, possuem etapas semelhantes (Cálculo do índice e seleção da nuvem para a operação, *Download* do sistema de arquivos, Compressão do arquivo e Transferência do arquivo), tendo como única diferença a execução da fase de filtragem das nuvens elegíveis para o recebimento dos arquivos antes da seleção da nuvem para o envio dos mesmos. Além disso, em ambos os fluxos são realizadas três operações com o arquivo:

- *Download* do sistema de arquivos: quando um arquivo for solicitado, replicado ou armazenado, é necessário que o mesmo seja disponibilizado no sistema de arquivos antes da sua transferência entres as nuvens. Assim, é necessário que a política de armazenamento solicite ao sistema de gerenciamento a cópia do arquivo para o local indicado;
- Compressão do arquivo: após o arquivo estar disponível no local indicado do sistema de arquivos, ele será compactado pela política de armazenamento;
- Transferência do arquivo: a transferência entre a origem e o destino é realizada utilizando o protocolo FTP (*File Transfer Protocol*, através da conexão utilizando um usuário e senha. O local da transferência será informado na mensagem de retorno.

Nas próximas subseções, as atividades do processo de Armazenamento e Recuperação serão apresentadas em detalhes.

4.2.1 Filtragem das Nuvens Elegíveis

Esta fase é aplicada apenas na função de *upload*, pois o espaço disponível nas nuvens candidatas ao recebimento do arquivo deve ser suficiente para o seu armazenamento.

A Filtragem das nuvens elegíveis visa selecionar apenas as nuvens que serão consideradas para a transferência, garantindo assim, que o arquivo não seja enviado para uma nuvem que não possua condições de recebê-lo, ou seja, não tenha espaço suficiente.

A política considera que o servidor possui espaço disponível se consegue armazenar o arquivo que será enviado e além disso, possui pelo menos 10% da capacidade de armazenamento livre. Este valor é definido através da variável *MAXCAPACITY* presente no *StorageService*. A justificativa para o valor escolhido é a necessidade de manter um espaço disponível no servidor para suas atividades básicas e assim evitar o seu travamento.

4.2.2 Cálculo do Índice e Seleção da Nuvem

Tendo a relação das nuvens candidatas, é necessário escolher a melhor entre elas para a operação desejada: *Download* ou *Upload*. Para isso, as nuvens serão classificadas segundo alguns critérios:

- Latência: uma medida do atraso de um provedor a outro. Possuir latência mais elevada indica atrasos mais longos. A latência nunca pode ser eliminada inteiramente, e é usada como uma medida do desempenho da rede. Durante a execução do *workflow*, a latência pode variar dependendo da carga imposta à rede [52].

  O cálculo da latência pode ser realizado de um único ponto por meio da utilização do comando *ping*, disponível em diversos sistemas operacionais. Por não processar os pacotes, o *ping* apenas retorna uma mensagem quando recebe um pacote, assim é um meio preciso para medir a latência. Este critério é o que apresenta maior impacto na transferência dos dados entre as nuvens;

- Custo de armazenamento: como a arquitetura BioNimbuZ possibilita a integração de diversos tipos de nuvens, o custo de armazenamento será considerado na escolha da nuvem, priorizando, dessa forma, nuvens federadas que não cobrem taxas pela transferência e o armazenamento dos dados.

  O custo de armazenamento servirá, nesta política, apenas como critério de desempate. Havendo nuvens que apresentem condições semelhantes para realizar a operação, será priorizada aquela que não envolva custo.

  Normalmente, existem dois tipos de custos em nuvens computacionais relativos ao armazenamento: (a) custo de armazenamento, pago por *gigabyte* armazenado; e, (b) custo de transferência de dados, pago por *gigabyte* trafegado, podendo haver valores diferentes para determinadas faixas e localização do destino/origem. Na política proposta foi assumido o valor para transferência de um *gigabyte*;

- Núcleos de processadores: o maior número de núcleos de processadores na nuvem candidata possibilita a resposta mais rápida à solicitação (*download* do sistema de arquivos e compressão), o que atende tanto às necessidades na operação de *download*, e a execução mais rápida dos trabalhos futuros, quanto às necessidades na operação de *upload*. Neste último caso, objetiva-se a diminuição das transferências de arquivos entre as nuvens, ou seja, pretende-se que a tarefa seja executado nas máquinas que possuem o arquivo.

  Além disso, uma nuvem que possui maior poder computacional tem condições de atender de forma mais eficiente à solicitação realizada, além de concluir tarefas em andamento em tempo menor, possibilitando a execução de um número maior de tarefas;

- Carga de trabalho: caso a nuvem esteja sobrecarregada, a possibilidade de executar determinada tarefa, assim que os dados estiverem disponíveis, é baixa, podendo desta forma, influenciar o tempo de execução de todo o *workflow*.

  Este critério visa garantir que a solicitação será atendida assim que for recebida pela nuvem. Havendo sobrecarga, o *download* do sistema de arquivos e a sua compressão estará comprometida, visto que não haverá recursos para serem utilizados nestas operações.

  Além disso, a combinação de alta carga de trabalho com baixo número de núcleos de processadores levará à execução mais demorada das operações e por consequência de todo o *workflow*.

Dessa forma, o cálculo do índice é realizado pela política de armazenamento, por meio de informações fornecidas pelo *plugin* de cada nuvem integrante da federação. Cada critério é padronizado, podendo variar entre 0 e 1. Após esta normalizado, cada critério será multiplicado pelo peso e ordenado de forma decrescente pelo índice. Desta forma, a nuvem que possuir o melhor índice (valor mais baixo) será escolhida para o armazenamento ou a recuperação do arquivo, como mostrado na Equação 4.1.

$$I = (\frac{l}{max(l)} * wl) + (\frac{ca}{max(ca)} * wca) + (\frac{ct}{100} * wct) - (\frac{np}{max(np)} * wnp) \tag{4.1}$$

Na Equação 4.1, I é o índice para seleção da nuvem, l é a latência de rede entre a origem e o destino, ca é o custo de armazenamento da nuvem candidata, ct é a carga de trabalho da nuvem candidata, np é a quantidade de núcleos de processadores na nuvem. Os pesos são: wl - latência, wca - custo de armazenamento, wct - carga de trabalho, wnp - núcleos de processadores.

Os critérios, carga de trabalho e núcleos de processadores foram utilizados por causa da necessidade de completar as operações na nuvem, baixar o arquivo do sistema de arquivos e realizar a compressão, estes dois processos são afetados pelas características do provedor. Quando um arquivo é solicitado, a nuvem que recebeu a solicitação deve descarregar o arquivo, entretanto, se esta nuvem estiver sobrecarregada ou possui capacidade de computacional baixa, a solicitação levará mais tempo para ser atendida, e assim, o tempo da política será afetado positivamente.

Os critérios latência, carga de trabalho e custo de armazenamento são prejudiciais à execução da operação, ou seja, quanto maior os seus valores, maior tempo será dispendido pela política de armazenamento. Por este motivo, seus valores são somados na Equação 4.1. Por outro lado, o critério núcleo de processadores traz benefícios à execução da política de armazenamento, assim, seu valor é subtraído para garantir que a nuvem que tiver o índice mais baixo seja considerada a melhor.

4.2.3 *Download* do Sistema de Arquivos

O *download* do sistema de arquivos é necessário para disponibilizar os dados em um local onde as outras nuvens tenham acesso. Além disso, em nuvem, uma das principais tecnologias adotadas para a execução de ferramentas de bioinformática é o *framework* Apache Hadoop [53], em que o modelo MapReduce [54] e seu sistema de arquivos distribuídos (HDFS) [38] são utilizados como infraestrutura para distribuir o processamento em grande escala, e realizar o armazenamento dos dados. Isto é devido ao fato da paralelização não exigir a comunicação entre tarefas processadas simultaneamente, uma vez que são independentes umas das outras.

Na implementação da política proposta, o Hadoop foi utilizado para armazenamento dos dados e execução distribuída entre os nós. Desta forma, é necessário que o arquivo seja copiado para o sistema de arquivos do servidor, para posteriormente ser comprimido.

4.2.4 Compressão do Arquivo

Após a escolha da nuvem, será realizada a compressão do arquivo. Este processo proporciona a redução do seu tamanho e, consequentemente, diminui o tempo de transferência do arquivo.

Arquivos biológicos, que são utilizados pelo BioNimbuZ, possuem características específicas que possibilitam alta taxa de compressão. Isso decorre dos arquivos serem compostos por apenas quatro caracteres distintos (A, C, G, T) que representam as bases, respectivamente, Adenina, Citosina, Guanina e Timina [55], além de espaços vazios. Ru-

bin [56] apresenta a abordagem básica para a compressão de texto, dividir o texto original em *substrings* e substituir cada uma dessas *substrings* por um código.

Isso é muito bem aplicado em arquivos com as características citadas anteriormente, onde o universo de caracteres é limitado e há muitos espaços vazios. Assim, a compressão dos arquivos biológicos apresenta grande redução relativamente aao seu tamanho.

A biblioteca utilizada em nossa política de armazenamento é a Snappy [57]. Esta biblioteca não objetiva a compressão máxima do arquivo, entretanto, sua estratégia é a rapidez na operação e uma compressão moderada, ou seja, o arquivo não terá a redução máxima do seu tamanho, mas esta operação será realizada de forma rápida, assim o custo-benefício da operação proporciona vantagens em relação à máxima compressão.

4.2.5 Transferência do Arquivo

A transferência dos arquivos, após a sua compressão, é realizada utilizando o protocolo FTP, que é um protocolo padrão para a transferência de arquivos para ou a partir de máquinas remotas.

Entre as vantagens deste protocolo estão a autenticação de usuários por meio de nome de usuário e senha, além da navegação nos diretórios de sistemas de arquivos locais e remotos.

4.3 A Implementação da Política Proposta

O Serviço de Armazenamento, por meio da política de armazenamento apresentada, irá coordenar as estratégias de armazenamento dos dados produzidos pelas tarefas do *workflow*, selecionar a origem de onde os arquivos serão carregados e quando e onde a replicação dos dados será realizada.

O Serviço de Armazenamento é parte do núcleo da arquitetura BioNimbuZ, da mesma forma que o Serviço de Escalonamento, Serviço de Descobrimento, Controlador de Tarefas, entre outros. Estes serviços comunicam-se através da troca de mensagens, onde o serviço demandante envia uma mensagem com os dados da solicitação, e recebe uma resposta contendo os dados solicitados. Esta troca de mensagem é realizada utilizando chamadas RPC através do *framework* Apache Avro. Toda a arquitetura foi desenvolvida utilizando a linguagem de programação Java. Assim, os passos a serem realizados no momento do armazenamento dos arquivos são (veja Figura 4.3):

1. o Serviço de Escalonamento envia uma mensagem do tipo *StoreReq* ao Serviço de Armazenamento a fim de obter o local para onde o arquivo deverá ser enviado;

2. o Serviço de Armazenamento irá solicitar ao Serviço de Descobrimento, através da mensagem *CloudReq*, informações sobre todas as nuvens disponíveis na federação, e assim escolher a melhor entre elas;

3. o Serviço de Descobrimento irá buscar as informações necessárias no *plugin* de integração;

4. de posse das informações, o Serviço de Descobrimento enviará ao Serviço de Armazenamento uma mensagem do tipo *CloudReply* contendo uma coleção de dados da classe *PluginInfo*, com os dados coletados pelo serviço até o momento, incluindo informações referente as condições de armazenamento de cada provedor integrante da federação. Entre as informações constantes nesta classe estão: a latência, a carga de trabalho, o número de processadores e o custo de armazenamento;

5. através dos dados recebidos o Serviço de Armazenamento escolherá a nuvem que irá receber o arquivo e responderá ao Serviço de Escalonamento por meio de uma mensagem do tipo *StoreReply*, contendo o local onde o arquivo deve ser armazenado;

6. após receber esta informação o Serviço de Escalonamento realiza a transferência do arquivo ao plugin indicado;

7. ao final deste processo, o *plugin* envia uma mensagem do tipo *StoreAck* para o solicitante confirmando o armazenamento.

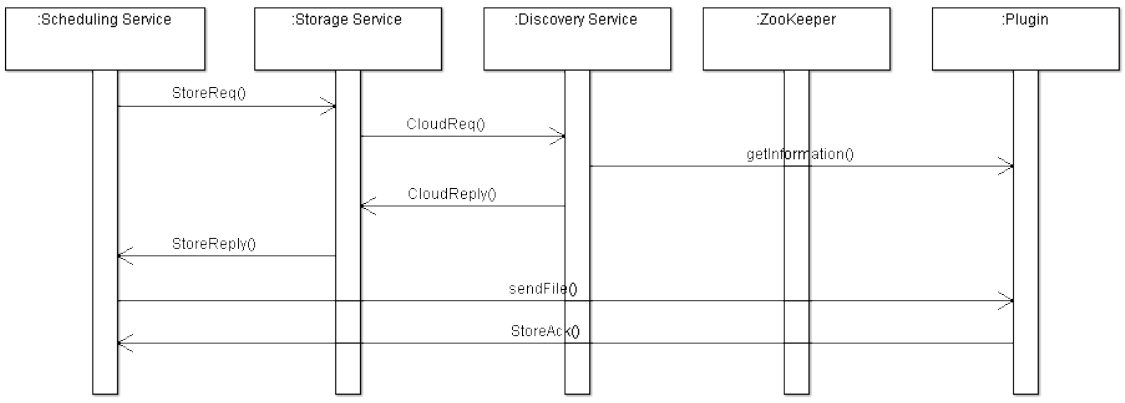

Figura 4.3: Diagrama de Sequência da Troca de Mensagens no Armazenamento de um Arquivo.

O processo de replicação dos arquivos é semelhante ao apresentado acima, e pode ser observado na Figura 4.4. Este processo também é iniciado em outros dois momentos: (a) no armazenamento de um novo arquivo proveniente de uma das etapas do *workflow* e

saída de algum provedor da federação. Neste último caso, após receber a mensagem de saída, algum dos outros provedores carregará a lista de arquivos registrados no ZooKeeper [5], verificará a necessidade de replicação e, em caso positivo, replicará estes arquivos em outras nuvens, até o limite configurado no Serviço de Armazenamento.

A Figura 4.4, apresenta o diagrama de atividade quando um usuário carrega um novo arquivo para a nuvem e este precisa ser replicado. Os passos para essa operação são:

1. o usuário solicita ao Serviço de Armazenamento o carregamento de um novo arquivo para a nuvem;
2. o Serviço de Armazenamento fará o armazenamento do arquivo, enviando uma solicitação ao *plugin* de integração;
3. o *plugin*, após o armazenamento, enviará a confirmação de sucesso da operação;
4. o Serviço de Armazenamento fará o registro do arquivo no ZooKeeper;
5. após o armazenamento, o Serviço de Armazenamento verificará o número de cópias existentes na federação, caso este número seja inferior ao configurado, ele solicitará a replicação;
6. o Serviço de Armazenamento solicitará a transferência de uma cópia do arquivo para a nuvem selecionada;
7. o *plugin*, após o armazenamento, enviará a confirmação de sucesso da operação;
8. o Serviço de Armazenamento fará o registro da réplica do arquivo no ZooKeeper.

No recebimento do pedido via Serviço de Armazenamento, a política de armazenamento irá decidir o melhor provedor para atender a solicitação, isto é feito por meio da *interface PluginInfo chooseStorage(FileInfo file)*, definida pela arquitetura BioNimbuZ [1]. Assim, a arquitetura possibilita a existência de diversas políticas de armazenamento.

Uma política de armazenamento deve implementar esta interface de forma que o Serviço de Armazenamento repasse as informações sobre o arquivo (classe *FileInfo*) e o provedor de infraestrutura (classe *PlugInfo*) para onde será feito o *upload* ou de onde será feito o *download* do arquivo. Na implementação desta interface é que a política proposta irá executar as etapas: filtragem das nuvens elegíveis e, cálculo do índice e seleção da nuvem para o envio.

Para realizar essas funções, o Serviço de Armazenamento possui três grupos de mensagens. O primeiro grupo trata do processo de decisão sobre o local de armazenamento de um arquivo:

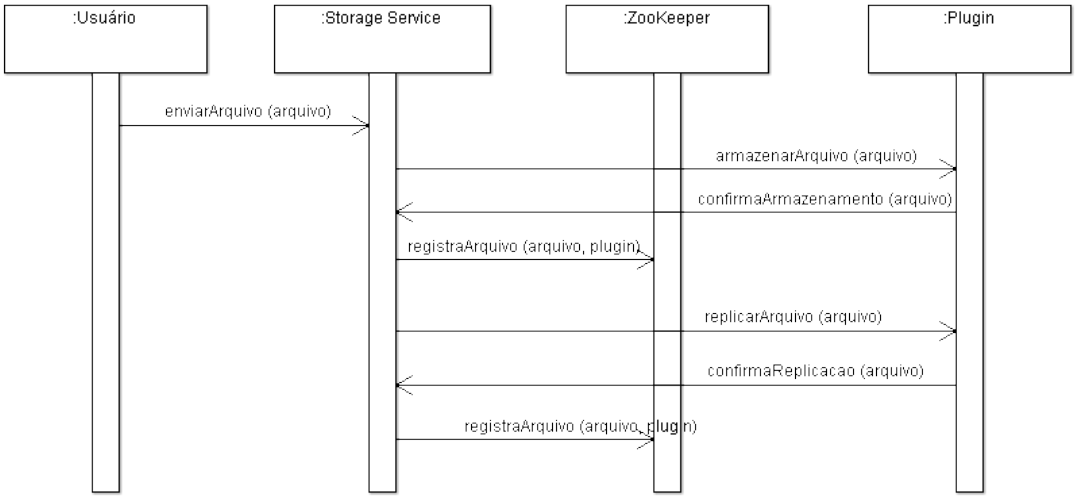

Figura 4.4: Replicação de Dados.

- *StoreReq*: requisição de escolha do local de armazenamento de um arquivo. Deve conter dados da classe *FileInfo* para que sejam usados pela política de armazenamento;
- *StoreReply*: resposta do Serviço de Armazenamento indicando o local de armazenamento para o provedor que solicitou o armazenamento. Deve conter, além dos dados da requisição, dados da classe *PluginInfo*;
- *StoreAck*: mensagem enviada por um *plugin* de integração após a finalização do *upload* de um arquivo à sua infraestrutura. Ela contém dados da classe *PluginFileInfo* que serão armazenados nas tabelas de arquivo do Serviço de Armazenamento.

O segundo grupo de mensagens refere-se às consultas sobre os arquivos armazenados na federação de nuvens:

- *ListReq*: requisição de listagem de arquivos armazenados na federação de nuvens;
- *ListReply*: resposta do Serviço de Armazenamento com uma coleção de dados da classe *FileInfo*, que representa a consolidação de todos os arquivos mantidos na federação naquele momento.

O último grupo de mensagens enviadas ao serviço de armazenamento refere-se à indicação do local de armazenamento de um arquivo, a partir do qual pode ser feito um *download*:

- *GetReq*: requisição sobre o local de onde pode ser feito o *download* de um dado arquivo. Deve conter, pelo menos, o identificador único do arquivo;
- *GetReply*: resposta do Serviço de Armazenamento com dados da classe *PluginInfo* sobre o provedor, para onde deve ser enviada uma mensagem do tipo *FilePrepReq* para iniciar o processo de *download*.

Tendo as informações do provedor que atenderá a solicitação, e esta sendo para armazenamento de um arquivo, o Serviço de Armazenamento enviará uma mensagem do tipo *FilePrepReq* que executará as etapas: *download* do Hadoop e compressão do arquivo. Após o término destas etapas, o solicitante poderá efetuar o *downlaod* do arquivo via FTP. Após recebê-lo o arquivo será descompactado.

Caso a operação seja de *upload*, o Serviço de Armazenamento fará a compressão do arquivo localmente e enviará uma mensagem do tipo *StoreReq* para armazenamento do arquivo. Após a transferência, o arquivo será descompactado no local de destino e carregado no sistema de arquivos do Hadoop.

4.4 Considerações Finais

A política de armazenamento proposta neste trabalho busca selecionar a melhor nuvem para o armazenamento dos dados, com o objetivo de garantir maior disponibilidade da informação e diminuir a transferência dos dados em decorrência do arquivo estar armazenado na nuvem que irá receber a execução do trabalho. Além disso, no caso de *download*, a política tem por objetivo responder com maior rapidez a solicitação realizada, e assim disponibilizar o arquivo no local desejado com o menor tempo.

Assim, a política de armazenamento, por meio dos critérios definidos (latência, núcleo de processadores, carga de trabalho e custo de armazenamento) tenta determinar qual é a melhor nuvem. Além disso, a fase de compressão do arquivo causa grande impacto no tamanho do arquivo, e por consequência no tempo necessário para a sua transferência.

No próximo capítulo serão apresentados os resultados obtidos com a utilização da política proposta.

Capítulo 5

Estudo de Caso

Neste capítulo, é apresentado o estudo de caso realizado para validar a política de armazenamento proposta. Primeiramente, na Seção 5.1 é apresentado o *workflow* utilizado no estudo de caso. Posteriormente, a Seção 5.2 apresenta o ambiente dos testes, as configurações de hardware e as ferramentas instaladas. Na Seção 5.3, são discutidos os resultados obtidos com a política de armazenamento.. Ao final, na Seção 5.4, são feitas as considerações finais referente aos resultados apresentados.

5.1 *Workflow* dos Testes

O *workflow* de bioinformática escolhido para o estudo de caso tem como objetivo identificar genes diferencialmente expressos em células humanas cancerosas do rim e do fígado [58, 59], com fragmentos sequenciados por sequenciadores Illumina [6]. O *workflow* consiste em quatro fases, como mostrado na Figura 5.1.

Esse *workflow* foi escolhido porque utiliza ferramentas e arquivos comuns da bioinformática, além disso, ele também foi utilizado por Saldanha [1] quando propôs a primeira versão da arquitetura BioNimbuZ [1].

Na primeira fase, de mapeamento, os fragmentos são mapeados para os 24 cromossomos humanos de referência. O objetivo é identificar a região do genoma de referência onde cada fragmento está localizado. Um conjunto de fragmentos mapeados para a mesma região nos permite inferir que eles possuem a mesma organização estrutural do genoma de referência. A ferramenta utilizada para fazer esse mapeamento foi o Bowtie [60]. A saída desta etapa é um arquivo em formato SAM [61], o qual é um formato de mapeamento delimitado por TAB, consistindo de um cabeçalho (opcional) e uma seção de alinhamentos.

Na segunda etapa, o formato SAM de saída do mapeamento é convertido para o formato BED com um *script* de conversão chamado *sam2bed*, implementado exclusivamente

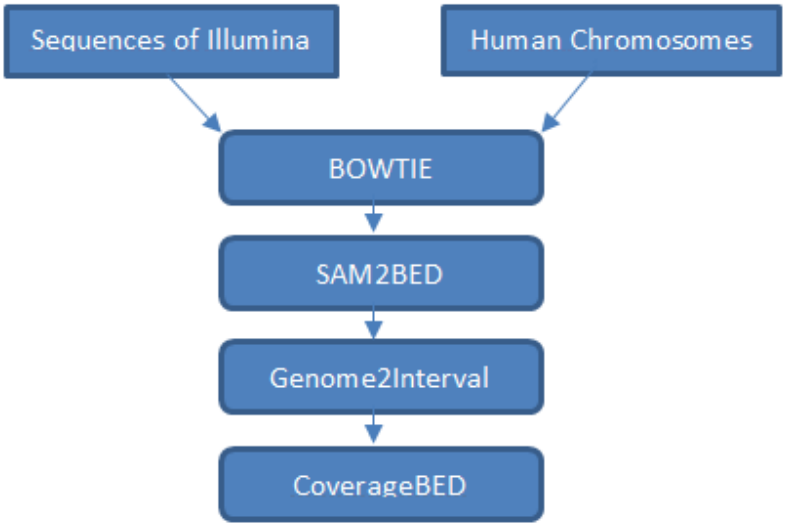

Figura 5.1: *Workflow* Utilizado para Identificar o Nível de Expressão de Genes em Células Cancerosas do Rim e do Fígado, adaptado de [1].

para este *workflow*. O formato BED foi criado pelo UCSC [62] e oferece uma maneira flexível para definir as linhas de dados que serão exibidas em uma faixa de anotação. Ele é um formato texto, com colunas separadas por caracteres TAB que representa intervalos no genoma associados à anotação.

Na terceira fase, com um *script* chamado *genome2interval*, são gerados intervalos de tamanho fixo, baseados no tamanho de cada cromossomo que será utilizado na fase seguinte.

Por fim, na quarta etapa, a partir das saídas da segunda e terceira etapas, são gerados histogramas que indicam o número de fragmentos mapeados para a faixa de cromossomos com a ferramenta *coverageBED* da suíte BEDtools [63]. Esta ferramenta também permite calcular a profundidade e a amplitude de cobertura dos recursos de um arquivo A em todos os recursos de um arquivo B.

Assim, os 24 cromossomos devem passar pelas quatro etapas cada um, totalizando 96 tarefas no *pipeline*. Os arquivos de entrada dos 24 cromossomos totalizaram 2,9 *gigabytes*, sendo o maior cromossomo com 248 *megabytes* e o menor com 51 *megabytes*. Os arquivos compactados totalizaram 893 *megabytes*, o que representa uma redução de 70%.

Tabela 5.1: Configuração de Hardware dos Nós das Nuvens.

	Tipo 1	Tipo 2 (Pequeno)	Tipo 3 (Grande)
Processador	Core(TM)2 Duo CPU E7300	Variável	Intel Xeon E5-2670
Núcleos	2	1	4
Memória	2GB	615MB	7,5GB
Armazenamento	160GB	20GB	500GB

5.2 Ambiente Computacional e Detalhes da Implementação

A federação de nuvens usada nos testes foi composta por três nuvens (veja a Figura 5.2). Cada nuvem possuia três nós, cada um com o sistema operacional Linux Ubuntu 12.04 instalado, o *framework* Apache Hadoop [53] e todas as ferramentas utilizadas no *workflow*. Assim, uma nuvem estava localizada na Universidade de Brasília e duas nuvens na Amazon, das quais uma estava localizada na Costa Oeste e outra na Costa Leste dos Estados Unidos.

A configuração de hardware dos nós das nuvens utilizadas no ambiente de testes é apresentado na Tabela 5.1. Como pode ser observado foram definidos três tipos de configuração de servidores: o **tipo 1** são as máquinas presentes na nuvem da Universidade de Brasília, estas máquinas não são virtualizadas, e possuem configurações intermediárias, se comparado aos outros dois tipos, quanto ao número de núcleos de processadores e memória. O **tipo 2** são microinstâncias, máquinas virtualizadas mais simples da Amazon. Nessas máquinas os recursos são limitados, por exemplo, 615 *megabytes* de memória, a configuração de CPU é variável pois possui baixa capacidade, entretanto, pode ser aumentada automaticamente em pequenos ciclos quando houver recursos disponíveis. Essa operação é controlada pelo sistema de virtualização. O **tipo 3** são máquinas virtualizadas otimizadas para a computação em geral, e possuem grande capacidade computacional, principalmente, CPU e memória.

Desta forma, e com as configurações de hardware das máquinas da nuvem da UnB

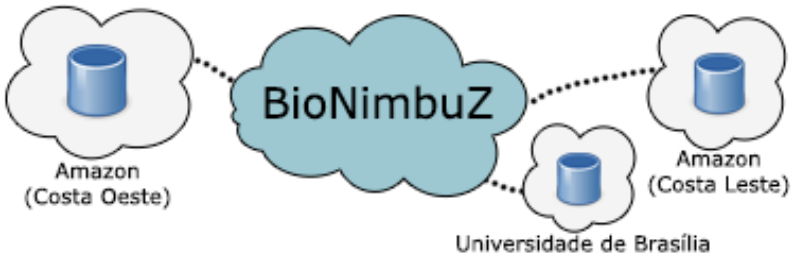

Figura 5.2: Federação Utilizada nos Testes.

sendo fixas, existiram duas configurações de hardware possíveis, mantendo sempre as mesmas configurações nas máquinas das duas nuvens da Amazon:

- Configuração de hardware I - uma nuvem com três máquinas tipo 1 e duas nuvens com três máquinas tipo 2 cada uma. Essa configuração possui baixa capacidade computacional dentro da federação;
- Configuração de hardware II - uma nuvem com três máquinas tipo 1 e duas nuvens com três máquinas tipo 3 cada uma. Essa configuração possui alta capacidade computacional dentro da federação;

Além das configurações de hardware, para testar a política de armazenamento proposta, dois escalonadores (o escalonador é o serviço da nuvem federada responsável por indicar onde o serviço será executado) foram utilizados no BioNimbuZ, os quais foram:

- *Round Robin*: um algoritmo simples de escalonamento de processos em sistemas operacionais, atribuindo fatias de tempo para cada processo em partes iguais e de forma circular, e uso dos processos sem prioridade;
- AcoSched - *Ant Colony Optimization (ACO)* [64]: um algoritmo de busca aleatória que imita o comportamento de uma colônia de formigas reais em busca de alimento, utilizando feromônios para traçar os caminhos, adaptando as execuções a medida que são realizadas.

  O algoritmo utiliza heurísticas para realizar o escalonamento e fazer a utilização de dados computacionais mais próximos da realidade. Além disso, ele considera a localização do arquivo como fator para sua decisão. Para otimizar o tempo de execução das tarefas, este escalonador prioriza tarefas que possuem arquivos de entrada de tamanho maior.

  Outra característica importante deste escalonador é que ele considera as execuções anteriores de cada recurso, utilizando-se desses dados para calcular o tempo aproximado que será utilizado para a execução de uma tarefa nesses recursos.

Os testes foram executados em quatro cenários distintos. Cada cenário combinou diferentes configurações de hardware (Tabela 5.1) e escalonadores. Assim, os quatro cenários usados nos testes foram:

- Cenário 1: configuração de hardware I executando o escalonador *Round Robin*;
- Cenário 2: configuração de hardware II executando o escalonador *Round Robin*;
- Cenário 3: configuração de hardware I executando o escalonador *AcoSched*;

- Cenário 4: configuração de hardware II executando o escalonador *AcoSched*.

Para cada cenário apresentado, cinco execuções de testes foram realizadas, totalizando 20 execuções. Em cada uma das execuções, os 24 cromossomos passaram pelas quatro fases do *workflow*, totalizando 96 tarefas por execução. Assim, o total de tarefas executadas foi igual a 1.920. Este número de execuções foi realizado para obter a média das operações e assim evitar que alguma exceção comprometesse os resultados.

5.3 Resultados e Discussão

O objetivo dos testes foi analisar o comportamento da política de armazenamento proposta nos cenários apresentados. Os diversos comportamentos servem como parâmetro para a definição dos pesos de cada critério, além do comparativo com as execuções utilizando a política de armazenamento original.

A Tabela 5.3 apresenta o tempo de execução total do *workflow* (TET), tempo de *download* do arquivo do Hadoop (TDH), tempo de compressão do arquivo (TC), tempo de transferência do arquivo (TT), tempo total da política de armazenamento (TTP) e o percentual que a política de armazenamento representou no tempo total de execução (PP). Devido ao número de tarefas, foram escolhidas, de forma arbitrária, apenas seis tarefas por cenário, sendo três pares de tarefas, ou seja, três arquivos diferentes em cada cenário, contendo sempre a execução da nuvem da UnB, pois a mesma não sofreu alteração nas configurações de hardware entre os cenários. Estas tarefas servem para mostrar as variações entre as configurações de hardware e latência, e seu impacto no tempo total de execução.

Nesse cenário, a Equação 5.1 apresenta o tempo total da política de armazenamento (TTP):

$$TTP = TDH + TC + TT \tag{5.1}$$

Na Tabela 5.2, é apresentado o tamanho dos cromossos utilizados nas tarefas. Na Tabela 5.3, pode-se observar que as tarefas executadas na nuvem da UnB apresentaram o mesmo padrão, alta representatividade da política de armazenamento (percentual do tempo total) e rápido processamento, como pode ser observado nas linhas 2, 6, 12, 14, entre outras. Nestas execuções, o tempo da política de armazenamento representou grande parte do tempo total de execução, chegando a 55% na execução do arquivo *chr15.fa* (linha 14). Isso deve-se ao fato da latência de rede desta nuvem (UnB) ser bastante superior às outras nuvens, porém seu processamento era superior nos cenários 1 e 3. Essa mesma tarefa, em outra execução, foi alocada para a nuvem da Amazon Costa Oeste (Amazon CO), e o

Tabela 5.2: Tamanho dos Cromossomos (em *Megabytes*).

Cromossomo	Arquivo	Tamanho (MB)
1	chr1.fa	248
2	chr2.fa	242
3	chr3.fa	197
4	chr4.fa	190
5	chr5.fa	180
6	chr6.fa	170
7	chr7.fa	158
8	chr8.fa	145
9	chr9.fa	140
10	chr10.fa	135
11	chr11.fa	134
12	chr12.fa	133
13	chr13.fa	114
14	chr14.fa	106
15	chr15.fa	102
16	chr16.fa	90
17	chr17.fa	80
18	chr18.fa	77
19	chr19.fa	58
20	chr20.fa	62
21	chr21.fa	47
22	chr22.fa	51
X	chrX.fa	154
Y	chrY.fa	59

Tabela 5.3: Tempos de Execução de Algumas Tarefas.

Linha	Cenário	Nuvem de Exec.	Arquivo	TET	TDH	TC	TT	TTP	PP
1	Cenário 1	Amazon CO	chr11.fa	3699	82	15	63	160	4%
2	Cenário 1	UnB	chr11.fa	1505	35	22	624	681	45%
3	Cenário 1	Amazon CO	chr21.fa	1661	32	8	9	49	3%
4	Cenário 1	UnB	chr21.fa	980	19	2	412	433	44%
5	Cenário 1	Amazon CO	chr7.fa	3774	49	67	26	142	4%
6	Cenário 1	UnB	chr7.fa	2174	21	20	1117	1158	53%
7	Cenário 2	Amazon CO	chr12.fa	766	5	3	13	21	3%
8	Cenário 2	UnB	chr12.fa	2606	5	2	926	933	36%
9	Cenário 2	Amazon CO	chr15.fa	567	3	1	9	13	2%
10	Cenário 2	UnB	chr15.fa	953	3	1	370	374	39%
11	Cenário 2	Amazon CO	chr7.fa	604	4	3	14	21	3%
12	Cenário 2	UnB	chr7.fa	2160	3	2	814	819	38%
13	Cenário 3	Amazon CO	chr15.fa	1455	19	10	10	39	3%
14	Cenário 3	UnB	chr15.fa	1566	100	46	722	868	55%
15	Cenário 3	Amazon CO	chr16.fa	2040	14	10	22	46	2%
16	Cenário 3	UnB	chr16.fa	943	18	4	455	477	51%
17	Cenário 3	Amazon CO	chr7.fa	5146	89	12	1180	1281	25%
18	Cenário 3	UnB	chr7.fa	2716	111	58	1134	1303	48%
19	Cenário 4	Amazon CO	chr10.fa	1067	15	7	118	140	13%
20	Cenário 4	UnB	chr10.fa	1145	3	1	565	569	50%
21	Cenário 4	Amazon CO	chr11.fa	602	3	1	16	20	3%
22	Cenário 4	UnB	chr11.fa	1661	4	3	353	360	22%
23	Cenário 4	Amazon CO	chr12.fa	706	2	3	23	28	4%
24	Cenário 4	UnB	chr12.fa	1197	3	2	451	456	38%

tempo da política representou apenas 3% do tempo total de execução. Entretanto, se as duas execuções forem observadas, será notado que houve uma variação de apenas 8%, ou seja, apesar da alta latência de rede da nuvem UnB, o tempo total foi compensado pelo maior poder de processamento.

Nos cenários 2 e 4, onde o poder de processamento era alto, a latência teve maior impacto na execução total do trabalho. Por exemplo, na execução do trabalho do arquivo *chr7.fa* (linhas 11 e 12), os tempos de *download* do Hadoop e o tempo de compressão do arquivo foram semelhantes. Entretanto, o tempo de transferência do arquivo foi bem distinto, sendo 58 vezes maior na execução na nuvem da UnB. Essa diferença se deu em decorrência da latência de rede da nuvem da UnB.

Assim, o impacto causado pela latência de rede na política de armazenamento é determinante para a escolha da nuvem de onde o arquivo será recuperado. Assim, considerou-se que seu peso no cálculo do índice (Equação 4.1) deveria ser de 50%.

Outro critério que contribui negativamente para a política de armazenamento é o poder de processamento das nuvens computacionais da federação. Quando executadas com baixos recursos (cenários 1 e 3), o tempo de *download* do Hadoop e compressão do arquivo foram bastante superiores. Vale observar que para a execução do trabalho do arquivo *chr11.fa* (linhas 1 e 2 no cenário 1, e linhas 21 e 22 no cenário 4), o tempo destas operações na execução no cenário 1, foi até 27 vezes superior a da execução no cenário 4.

Este mesmo padrão ocorreu comparando-se as outras execuções, assim, a importância do poder de processamento é seu impacto direto na realização de duas etapas, entre três, da política de armazenamento (Equação 5.1). Desta forma, o peso no cálculo do índice foi considerado 30%.

Associado a este critério, a carga de trabalho também influencia as duas operações citadas (*download* do Hadoop e compressão do arquivo). Em algumas execuções, como é o caso do arquivo *chr10.fa* no cenário 4 (linha 19), apesar de possuir alto poder de processamento, o tempo para realização das duas operações foi superior ao apresentado em outros arquivos semelhantes. Isso pode ser explicado pela sobrecarga do sistema no momento da operação, fazendo com que o mesmo demorasse maior tempo para responder à solicitação. Desta forma, o peso no cálculo do índice foi considerado 15%.

O último critério do índice, o custo de armazenamento, foi utilizado apenas como critério de desempate, a fim de dar preferência a nuvens que não possuiam custos na operação, por isso seu peso no cálculo do índice foi de 5%. Assim temos a Equação 5.2.

$$I = (\frac{l}{max(l)} * 0.5) + (\frac{ca}{max(ca)} * 0.05) + (\frac{ct}{100} * 0.15) - (\frac{np}{max(np)} * 0.3) \tag{5.2}$$

Tabela 5.4: Tempos de Execução.

Cenário	Política de Armazenamento	Tempo Médio de Execução do *Workflow* (hh:mm:ss)	Tempo da Política de Armazenamento (hh:mm:ss)
Cenário 1	Política Proposta	04:40:24	00:53:37
Cenário 1	Política Original	05:22:11	01:35:46
Cenário 2	Política Proposta	01:35:47	00:33:11
Cenário 2	Política Original	02:12:41	00:58:54
Cenário 3	Política Proposta	03:34:21	00:27:12
Cenário 3	Política Original	04:05:10	01:08:26
Cenário 4	Política Proposta	01:24:33	00:20:48
Cenário 4	Política Original	01:50:54	00:49:32

Tabela 5.5: Tempos Médios da Política de Armazenamento Proposta (em segundos).

Passo	Original	Cenário 1	Cenário 2	Cenário 3	Cenário 4
Passo 1 - Bowtie	52	44	28	38	16
Passo 2 - SAM2BED	15	9	4	9	5
Passo 3 - Genome2Interval	14	5	4	5	4
Passo 4 - CoverageBED	13	8	5	10	4

Utilizando-se desses pesos, foi realizada uma bateria de testes. Em cada cenário foi executado o BioNimbuZ com a política de armazenamento proposta e a política de armazenamento original, presente na arquitetura proposta por Saldanha em [1], como pode ser observado na Tabela 5.4. Nossa política de armazenamento teve um ganho médio de 40%, quando comparado com a execução da política de armazenamento original. Assim, nessa tabela, na coluna Tempo Médio de Execução do *Workflow* é apresentado o tempo total para a execução das 96 tarefas do *workflow*, e na coluna Tempo da Política de Armazenamento, o tempo utilizado apenas pela política na execução das três etapas do fluxo (*download* do hadoop, compressão do arquivo e transferência do arquivo).

O *workflow* utilizado nos testes possuia arquivos de tamanho e formatos diferentes. Assim, outra análise foi realizada a fim de verificar se havia impactos diferentes da política de armazenamento no formato e no tamanho do arquivo. A Tabela 5.5 mostra os tempos médios da política de armazenamento nas execuções em todas os cenários. A coluna Original apresenta a média de valores das execuções utilizando a politica de armazenamento original no cenário 4, que apresentou os melhores tempos. Nas demais colunas são apresentados os valores médios de cada cenário utilizando a política de armazenamento proposta.

Devido ao tamanho dos arquivos de entrada, o passo 1 do *workflow* (Bowtie) necessitou de maior tempo para a realização das operações em todos os cenários. O tempo médio

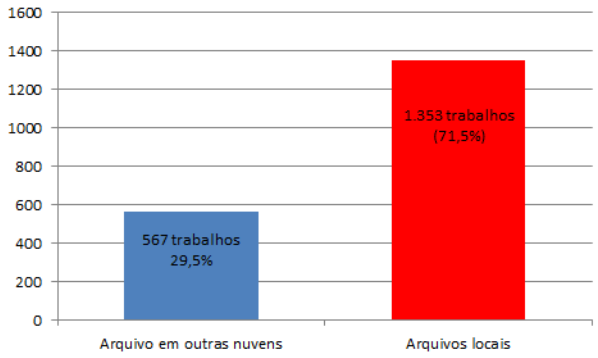

Figura 5.3: Número de Trabalhos que Necessitaram de Transferência do Arquivo.

utilizado pela política de armazenamento para a execução das tarefas com este tipo de arquivo foi de 52 segundos na política original, e de 16 segundos no cenário 4 utilizando a política de armazenamento proposta.

No tempo da política de armazenamento original estão contidas duas etapas do processo: *download* do arquivo do Hadoop e a transferência do arquivo, sendo que a primeira etapa utilizou 6 segundos e a segunda etapa 86 segundos. Na política de armazenamento proposta existem três etapas, além das duas da política de armazenamento original, o arquivo é compactado antes de ser transferido. No cenário que apresentou os melhores resultados para a política de armazenamento, cenário 4, o tempo utilizado na primeira etapa foi de 5 segundos, na compressão, 3 segundos e na transferência 8 segundos.

Conforme apresentado, a conribuição da nova política de armazenamento proposta é a adição do tempo de compressão e a redução significativa no tempo de transferência do arquivo. Apesar do tempo utilizado para a compressão do arquivo, o ganho proporcionado pela operação é constatado com a redução no tempo de transferência do mesmo.

Nas execuções nos quatro cenários foram processadas 1.920 tarefas, sendo que 567 (29,5%) necessitaram de transferência de arquivos que estavam em outras nuvens (veja Figura 5.3). Em execuções feitas com a política de armazenamento original, 32,2% dos arquivos tinham sido transferidos. Esta redução é explicada pela replicação de arquivos feita pela política de armazenamento. No entanto, a redução não foi tão significativa devido ao *Scheduler Service* realizar as suas tarefas mais rápido do que a replicação dos arquivos, ou seja, a replicação proporciona benefícios maiores para a tolerância a falhas do que para o tempo total de execução do *pipeline*.

Na Figura 5.4 é possível observar a quantidade de tarefas que necessitaram transferir arquivos quando utilizaram nuvens com hardware do tipo 3. Estas tarefas foram agrupadas por faixas, representando o percentual referente a política de armazenamento no tempo total de execução. Nestas configurações, 300 tarefas necessitaram de transferência de arquivos, e a maioria delas (129) gastou até 10% do tempo total de execução do *work-*

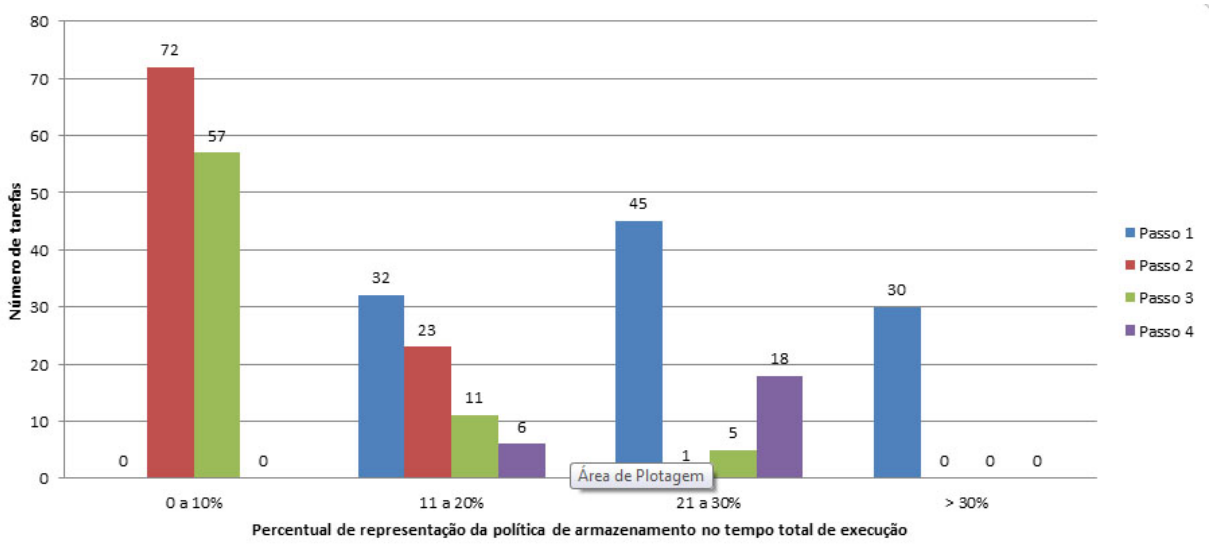

Figura 5.4: Número de Trabalhos que Necessitaram de Transferência do Arquivo nas Nuvens do Tipo 3.

flow com a política de armazenamento, respectivamente, 72 tarefas no Passo 2 e 57 no Passo 3, e apenas 30 tarefas necessitaram de mais de 30% do tempo total.

Nas execuções com hardware tipo 2 (Figura 5.5), o número de tarefas que necessitaram transferir arquivos foi de 267, sendo que 136 tarefas necessitaram até 10% do tempo total de execução.

O padrão do gráfico foi equivalente entre as Figuras 5.4 e 5.5, com a maioria das tarefas utilizando, no máximo 10%, do tempo total de execução pela política de armazenamento.

O tempo da política de armazenamento também afetou mais as execuções em máquinas com maior poder de processamento, ou seja, nas máquinas do tipo 3. Isso porque com grande poder de processamento o número de operações, nas faixas superiores do gráfico, é maior do que o número da mesma faixa nas máquinas do tipo 2. Por exemplo, enquanto que nas máquinas do tipo 3 foram 69 trabalhos, onde o tempo da política de armazenamento representou entre 20% e 30% do tempo total, nas máquinas do tipo 2, este número é igual a 56, ou seja, aumento de 23%. Este aumento é justificado pela rapidez na execução dos trabalhos em nuvens com maior poder computacional, pois as ferramentas de bioinformática requerem mais recursos dos servidores. Desta forma, quando este recurso é limitado, o tempo de execução aumenta consideravelmente, assim, o tempo da política de armazenamento tem menor impacto.

Em decorrência da execução paralela, não é possível garantir a ordem, bem como o local de execução (Universidade de Brasília, Amazon CO ou Amazon Costa Leste). Entretanto, analisando os tempos de transferência de arquivos nos *logs* de execução, foi possível descobrir que as tarefas que tiverem maior impacto no tempo da política foram os

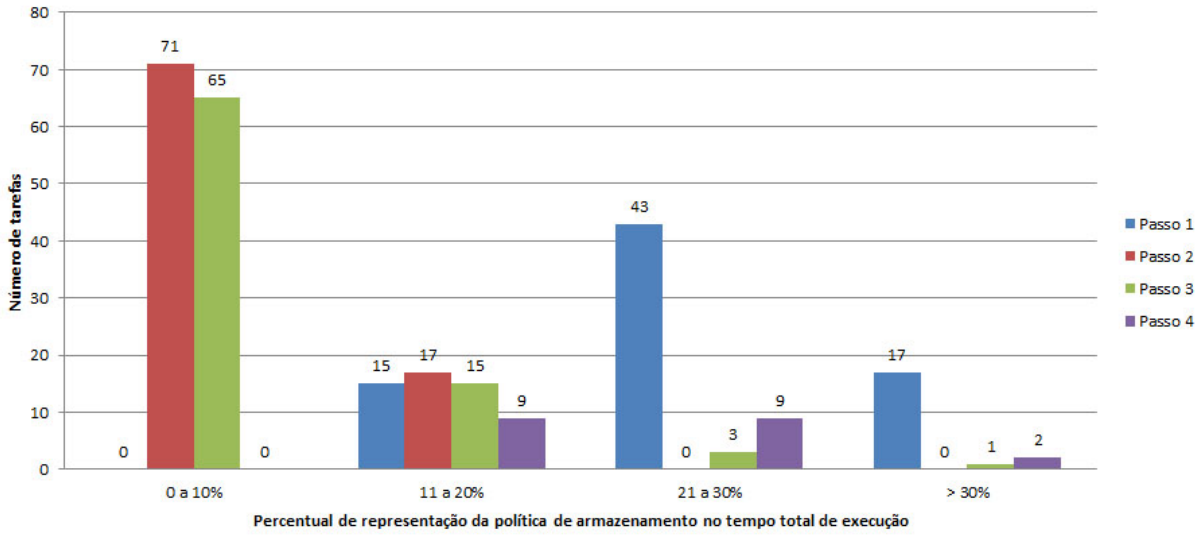

Figura 5.5: Número de Trabalhos que Necessitaram de Transferência do Arquivo nas Nuvens do Tipo 2.

que tiveram a execução enviada para a Universidade de Brasília, onde todos os trabalhos executados mostraram as mesmas características, baixa taxa de transferência e execução rápida.

Em comparação ao trabalho de Hiden et al. [45], este trabalho tem o diferenciar de realizar as operações com os arquivos, já que no trabalho dos autores citados, esta operação é realizada pelos provedores de armazenamento.

5.4 Considerações Finais

A política de armazenamento proposta conseguiu diminuir o tempo total de execução do *workflow* por meio de melhorias no processo de armazenamento e na recuperação das informações.

A seleção da melhor nuvem foi importante para que a operação terminasse no menor tempo possível. Outra operação que contribuiu significativamente para a transferência mais rápida dos arquivos foi a compressão dos dados. Por meio dela, o arquivo teve seu tamanho diminuido consideravelmente, o que proporcionou menor tempo de transferência. Assim, o tempo gasto com a operação de compressão não causou impacto negativo na política de armazenamento.

Assim, os testes mostraram que a política de armazenamento teve a mesma eficiência em arquivos grandes, utilizados na primeira fase do *workflow*, e também nos arquivos menores, utilizados nas demais fases. Isso ocorreu devido às características dos arquivos serem semelhantes.

A replicação dos dados proporcionou menor tempo de transferência de arquivos entre as nuvens. O resultado desta operação não foi melhor em decorrência da velocidade do agendamento das tarefas, o Serviço de Escalonamento executa mais rapidamente que o processo de replicação dos arquivos. Entretanto, proporcionou maior disponibilidade do arquivo.

Capítulo 6

Conclusão e Trabalhos Futuros

No presente trabalho, foi proposta uma política de armazenamento para dados genômicos em federações de nuvens computacionais. O objetivo da política de armazenamento é diminuir o tempo de transferência dos dados, e assim contribuir para a diminuição do tempo total de execução do *workflow*.

Na especificação da política de armazenamento, foram analisados critérios que influenciavam na transferência dos dados e na execução dos trabalhos. Os critérios selecionados foram a latência, o número de núcleos de processamento, a carga de trabalho e o custo de armazenamento dos dados. Este critérios receberam, respecitavamente, os seguintes pesos, 50%, 30%, 15% e 5%.

Além disso, foi realizado um estudo de caso da política de armazenamento proposta com a arquitetura BioNimbuZ, utilizada para a execução de ferramentas de bioinformática. Neste estudo de caso foram utilizadas três nuvens computacionais, sendo duas públicas e uma privada, com o intuito de verificar a funcionalidade e o seu desempenho na prática. Nesse estudo de caso, foi utilizado um *workflow* para a identificação de diferenças de expressão em genes de células cancerosas do rim e do fígado humano. Os dados utilizados foram fragmentos produzidos por um sequenciador de alto desempenho Illumina e os 24 cromossomos humanos. Para efetivar a análise, o *workflow* foi executado na arquitetura BioNimbuZ com o uso da política de armazenamento original e, posteriormente, com a política de armazenamento proposta.

Com os resultados obtidos, foi possível observar a efetividade da política de armazenamento, ou seja, a diminuição do tempo de transferência dos arquivos além da diminuição do tempo total de execução do *workflow*. A política de armazenamento foi mais efetiva em nuvens com maior poder de processamento e latência, onde o tempo da política de armazenamento utiliza maior parte do tempo total de execução do trabalho. Além disso, a política de armazenamento apresentou o mesmo comportamento tanto em arquivos pequenos (50 megabytes) como em arquivos maiores (270 megabytes). Também houve menos

replicação de arquivos entre os provedores.

Como trabalhos futuros, a implementação do SLA *Controller* para aplicar e monitorar níveis de serviço, pode contribuir para que o usuário selecione suas necessidades e assim a política de armazenamento poderia configurar o peso de cada critério, aumentando a velocidade da operação, diminuindo custos, entre outros. O Escalonador poderia considerar a localização física do arquivo como critério para seleção da nuvem de execução do trabalho. Outro aspecto que pode constribuir para diminuir o tempo de transferência do arquivo é o particionamento do mesmo. E, não associado a transferência, ainda é necessário a melhor da usabilidade do sistema para os usuários a qual se destina a ferramenta.

Finalmente, vale ressaltar que o presente trabalho resultou em um artigo aceito na *11th International Conference Applied Computing 2014 (AC 2014)* [65] a ser realizada em Outubro. Houve também a participação na escrita de um capítulo de livro (Bioinformatics |ISBN 980-953-307-202-4) [66].

Referências

[1] H. V. Saldanha, E. Ribeiro, M. Holanda, A. Araujo, G. Rodrigues, M. E. M. T. Walter, J. C. Setubal, and A. Davila, "A cloud architecture for bioinformatics workflows," in *1st International Conference on Cloud Computing and Services Science* (L. INSTICC, ed.), CLOSER 2011, pp. 1–8, 2011. iii, vii, 1, 17, 20, 36, 39, 40, 47

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 50–55, 2008. vii, 1, 6, 7, 8

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, pp. 1–10, 2008. vii, 5, 9

[4] D. Lima, B. Moura, E. Ribeiro, A. Araujo, M. Holanda, and M. E. Walter, "A storage policy for a hybrid federated cloud platform: A case study for bioinformatics," in *14th Simpósio em Sistemas Computacionais, WSCAD-SSC'2013*, 2013. vii, 1, 17, 18

[5] ZooKeeper, "Apache zookeeper." `http://zookeeper.apache.org/`, Abril 2014. vii, 21, 22, 23, 24, 36

[6] Illumina, "Illumina life sciences." `http://www.illumina.com`, Abril 2014. 1, 39

[7] 454, "454 life sciences. roche diagnostics corporation." `http://www.454.com/`, Abril 2014. 1

[8] Solid, "Life technologies & applied biosystems." `http://www.appliedbiosystems.com/`, Abril 2014. 1

[9] Amazon, "Amazon elastic compute cloud (amazon ec2)." `http://aws.amazon.com/ec2/`, Abril 2014. 1, 8, 10

[10] T. Redkar, *Windows Azure Platform*, vol. 1. Berkeley, CA, USA: Apress, 2th ed., 2011. 1

[11] D. Sanderson, *Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure*. O'Reilly Media, Inc., 1st ed., 2009. 1

[12] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010)*, pp. 21–23, Springer, 2010. 1, 5, 7, 10, 11

[13] T. J. Bittman, "The evolution of the cloud computing market." http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market, November 2008. 1

[14] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *3rd IEEE International Conference on Cloud Computing (IEEE Cloud 2010)*, (Miami, Florida, USA), pp. 337–345, Jul 2008. 1, 10

[15] P. Mell and T. Grance, "The NIST definition of cloud computing," tech. rep., National Institute of Standards and Technology, July 2009. 6, 9

[16] Google, "Google file system." http://research.google.com/archive/gfs.htm, Abril 2014. 8

[17] Google, "Google app engine." https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=pt-br, Abril 2014. 8

[18] Microsfot, "Windows azure." http://www.windowsazure.com/pt-br/, 2014. 8, 14

[19] Google, "Google drive." http://drive.google.com, Abril 2014. 8

[20] Zoho, "Zoho." http://www.zoho.com/, Abril 2014. 8

[21] B. Nicolae, "High throughput data-compression for cloud storage," in *Proceedings of the Third international conference on Data management in grid and peer-to-peer systems*, Globe'10, (Berlin, Heidelberg), pp. 1–12, Springer-Verlag, 2010. 10, 12, 15

[22] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, (Berkeley, CA, USA), pp. 15–15, USENIX Association, 2006. 12

[23] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, (New York, NY, USA), pp. 29–43, ACM, 2003. 12

[24] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 205–220, Oct. 2007. 12

[25] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, (New York, NY, USA), pp. 1–14, ACM, 2010. 12

[26] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of windows azure," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, (New York, NY, USA), pp. 367–376, ACM, 2010. 12

[27] A. Ruiz-Alvarez and M. Humphrey, "An automated approach to cloud storage service selection," in *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing*, ScienceCloud '11, (New York, NY, USA), pp. 39–48, ACM, 2011. 12

[28] D. Bermbach, M. Klems, S. Tai, and M. Menzel, "Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, CLOUD '11, (Washington, DC, USA), pp. 452–459, IEEE Computer Society, 2011. 12

[29] H.-Y. Lin, C.-M. Chen, Y.-L. Chen, Y.-H. Chang, and T.-W. Pai, "Construction of a taiwan biodiversity genetic database in cloud environments," in *Biometrics and Security Technologies (ISBAST), 2012 International Symposium on*, pp. 108–112, March 2012. 12

[30] T. A. S. Foundation, "Apache hbase." http://hbase.apache.org/, Junho 2014. 12

[31] I. Fetai and H. Schuldt, "So-1sr: Towards a self-optimizing one-copy serializability protocol for data management in the cloud," in *Proceedings of the Fifth International Workshop on Cloud Data Management*, CloudDB '13, (New York, NY, USA), pp. 11–18, ACM, 2013. 13

[32] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Gener. Comput. Syst.*, vol. 26, pp. 1200–1214, Oct. 2010. 13

[33] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, (Berkeley, Calif.), pp. 281–297, University of California Press, 1967. 13

[34] H. Stockinger, K. Stockinger, E. Schikuta, and I. Willers, "Towards a cost model for distributed and replicated data stores," in *Parallel and Distributed Processing, 2001. Proceedings. Ninth Euromicro Workshop on*, (Wien Univ., Austria), pp. 461–467, IEEE Computer Society, 2001. 13, 29

[35] Z. Jian-hua and Z. Nan, "Cloud computing-based data storage and disaster recovery," in *Future Computer Science and Education (ICFCSE), 2011 International Conference on*, (Chengdu, China), pp. 629–632, IEEE Computer Society, 2011. 13

[36] W. Zeng, Y. Zhao, K. Ou, and W. Song, "Research on cloud storage architecture and key technologies," in *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, (New York, NY, USA), pp. 1044–1048, ACM, 2009. 14

[37] H. Kavalionak and A. Montresor, "P2p and cloud: A marriage of convenience for replica management," in *Self-Organizing Systems* (F. Kuipers and P. Heegaard, eds.), vol. 7166 of *Lecture Notes in Computer Science*, pp. 60–71, Springer Berlin Heidelberg, 2012. 14

[38] D. Borthakur, "The Apache Software Foundation: HDFS Architecture." http://hadoop.apache.org/common/docs/r0.20.2/hdfs_design.pdf, 2008. 14, 33

[39] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2010. 14

[40] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 35–40, Apr. 2010. 14

[41] F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti, and E. Cesario, "The xtreemfs architecture—a case for object-based file systems in grids," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 2049–2060, Dec. 2008. 14

[42] Lustre, "Lustre file system." http://lustre.opensfs.org/, Maio 2014. 14

[43] J. Piernas, J. Nieplocha, and E. J. Felix, "Evaluation of active storage strategies for the lustre parallel file system," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC '07, (New York, NY, USA), pp. 28:1–28:10, ACM, 2007. 14

[44] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, (Berkeley, CA, USA), pp. 307–320, USENIX Association, 2006. 14

[45] H. Hiden, S. Woodman, and P. Watson, "Developing cloud applications using the e-science central platform," *PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A-MATHEMATICAL PHYSICAL AND ENGINEERING SCIENCES*, vol. 371, no. 1, pp. 17–32, 2013. 14, 50

[46] A. W. S. LLC, "Amazon simple storage service (amazon s3)." http://aws.amazon.com/s3/, Abril 2014. 14

[47] K.-J. Lin and J. Gannon, "Atomic remote procedure call," *Software Engineering, IEEE Transactions on*, vol. SE-11, pp. 1126–1135, Oct 1985. 25

[48] Avro, "Apache avro." http://avro.apache.org/, Abril 2014. 26

[49] Json, "Json." http://json.org/, Abril 2014. 26

[50] Thrift, "Apache thrift." http://thrift.apache.org/, Abril 2014. 26

[51] ProtoBuf, "Protocol buffers." https://code.google.com/p/protobuf/, Abril 2014. 26

[52] S. Cheshire, "Latency and the quest for interactivity." White paper, 1996. 31

[53] Apache, "Apache Hadoop." http://hadoop.apache.org/, Abril 2014. 33, 41

[54] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *6th Conference on Symposium on Operating Systems Design & Implementation*, (Berkeley, CA, EUA), pp. 10–10, USENIX Association, 2004. 33

[55] D. L. Nelson and M. M. Cox, *Lehninger princípios de bioquímica*, vol. 1. São Paulo: Sarvier, 3th ed., 2002. 33

[56] F. Rubin, "Experiments in text file compression," *Commun. ACM*, vol. 19, pp. 617–623, Nov. 1976. 34

[57] Google, "Snappy." http://code.google.com/p/snappy, Abril 2014. 34

[58] J. C. Marioni, C. E. Mason, S. M. Mane, M. Stephens, and Y. Gilad, "RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays.," *Genome research*, vol. 18, pp. 1509–1517, Sept. 2008. 39

[59] M. Robinson and A. Oshlack, "A scaling normalization method for differential expression analysis of RNA-seq data," *Genome Biology*, vol. 11, pp. R25+, Mar. 2010. 39

[60] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, no. 3, p. R25, 2009. 39

[61] S. Tools, "Formato SAM." http://samtools.github.io/hts-specs/SAMv1.pdf, Maio 2014. 39

[62] UCSC, "Ucsc." http://genome.ucsc.edu/, Maio 2014. 40

[63] A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features," *Bioinformatics*, vol. 26, pp. 841–842, Mar. 2010. 40

[64] G. de Oliveira, E. Ribeiro, D. Ferreira, A. Araujo, M. Holanda, and M. Walter, "Acosched: A scheduling algorithm in a federated cloud infrastructure for bioinformatics applications," in *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pp. 8–14, Dec 2013. 42

[65] R. F. Gallon, M. T. Holanda, A. P. F. Araújo, and M. E. M. T. Walter, "Storage policy for hybrid federated clouds: A case study for the bionimbuz platform." http://www.computing-conf.org/, Junho 2014. 53

[66] H. Saldanha, E. Ribeiro, C. Borges, R. G. A. Araujo, M. Holanda, R. T. M. E. Walter, and J. C. Setubal, "Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows," *Bioinformatics*, pp. 51–0878, 2012. 53