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

Novo Serviço de Armazenamento na Plataforma de Nuvem Federada BioNimbuZ

Lucas Facundo Neiva Santos

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

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

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia da Computação

Coordenador: Prof. Dr. Ricardo Pezzoul Jacobi

Banca examinadora composta por:

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo. (Orientadora) — CIC/UnB
Prof.ª Dr.ª Maria Emília Machado Telles Walter — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB

CIP — Catalogação Internacional na Publicação

Santos, Lucas Facundo Neiva.

Novo Serviço de Armazenamento na Plataforma de Nuvem Federada BioNimbuZ / Lucas Facundo Neiva Santos. Brasília : UnB, 2016.
76 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. Computação em Nuvem; Nuvem Federada; BioNimbuZ; Serviço de Armazenamento.

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

Novo Serviço de Armazenamento na Plataforma de Nuvem Federada BioNimbuZ

Lucas Facundo Neiva Santos

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

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo. (Orientadora)
CIC/UnB

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

Prof.ª Dr.ª Maristela Terto de Holanda
CIC/UnB

Prof. Dr. Ricardo Pezzoul Jacobi
Coordenador do Curso de Engenharia da Computação

Brasília, 8 de julho de 2016

Dedicatória

Dedico este trabalho à minha família, em especial à minha mãe, que me educou e me fez ser quem eu sou hoje. Dedico também aos meus amigos, que sempre me apoiaram e ajudaram durante todo este processo.

Agradecimentos

Agradeço primeiramente a minha orientadora, Aletéia Patrícia, por todo o apoio e paciência ao longo destes dois semestres, inclusive dedicando parte de seu tempo pessoal para auxiliar no desenvolvimento deste trabalho. Agradeço também ao Edward Ribeiro, que me acompanhou e auxiliou durante este último semestre, sempre buscando ajudar no que fosse necessário. Ao grupo do BioNimbuZ, em especial ao Breno Moura, pelo companheirismo e por toda a ajuda prestada durante as dificuldades iniciais que tive neste trabalho. Por fim, agradeço aos meus amigos, que me apoiaram e deram força durante todos os momentos em que precisei.

Resumo

A plataforma de nuvem federada é um cenário no qual múltiplos provedores de nuvem distintos tem seus serviços e recursos integrados para o usuário final de maneira transparente, oferecendo um maior poder de processamento e de armazenamento. O BioNimbuZ é um ambiente de nuvem federada com foco na execução de *workflows* de Bioinformática. Atualmente, o serviço de armazenamento do BioNimbuZ utiliza os serviços de infraestrutura dos provedores para armazenar arquivos nos discos de suas instâncias de máquinas virtuais, o que torna necessárias que ações de replicação de arquivos sejam realizadas pelo BioNimbuZ, aumentando o custo desta ação. A proposta deste trabalho é que este serviço seja alterado para utilizar os serviços de armazenamento já disponibilizados pelos provedores. Assim, o serviço de armazenamento provido pelo BioNimbuZ através da federação de nuvem será mais robusto e eficiente, pois reduzirá custos relacionados ao armazenamento de arquivos na federação e eliminará a limitação da capacidade de armazenamento na federação anteriormente relacionada a capacidade de armazenamento dos discos das instâncias de máquina virtual criadas no BioNimbuZ. Foram realizadas análises dos principais serviços de armazenamento oferecidos pelos grandes provedores de serviço em nuvem para se realizar a escolha dos serviços que compõe o novo serviço de armazenamento implementado no BioNimbuZ. Após a definição e implementação deste novo serviço, foram realizados testes para verificar seu funcionamento e performance, onde foram obtidos resultados comprovando seu funcionamento e que não houve uma perda de performance.

Palavras-chave: Computação em Nuvem; Nuvem Federada; BioNimbuZ; Serviço de Armazenamento.

Abstract

The federated cloud platform is a scenario in which multiple distinct cloud providers have their services and resources integrated to the final user transparently, offering greater processing power and storage capacity. The BioNimbuZ platform is a federated cloud environment with focus on executing Bioinformatics workflows. Currently, the BioNimbuZ's storage service uses the infrastructure services of the cloud providers to store files on the the hard drives of its virtual machine instances. This project's proposal is to modify this service so it uses the storage services already provided by the cloud providers. Thereby, making the storage service provided by the BioNimbuZ platform through the federated cloud more robust and efficient, therefore reducing the cost related to storing files in the federation and eliminating the storage limitation in the federation related to the capacity of hard drives of the instances of virtual machines created in BioNimbuZ. The main storage services offered by the big providers of cloud services to select the services to compose this new storage service implemented in BioNimbuZ. After the definition and implementation of the new service, tests were realized to verify its functioning and its performance, in which results were obtained proving its functionality and that there was no loss of performance.

Keywords: Cloud Computing; Federated Cloud; BioNimbuZ; Storage Service

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o aumento significativo da complexidade das aplicações e dos sistemas computacionais, vem surgindo cada vez mais a necessidade de maior poder de processamento e armazenamento, e com isso o paradigma tradicional de computação, com um único computador executando tarefas localmente, começa a não se mostrar suficiente em determinados cenários, o que deu origem inicialmente a novos paradigmas de programação como os Sistemas Distribuídos. Estes, por sua vez, podem ser classificados em categorias como *Clusters*, *Grids* ou Nuvens Computacionais. Nesse contexto, as nuvens computacionais surgiram através da necessidade de prover serviços e recursos sob demanda, de maneira distribuída e de qualquer lugar do mundo. O objetivo é reunir um conjunto de recursos para que se tenha um ambiente dinamicamente escalável e completamente transparente para seu consumidor final.

Atualmente, há vários provedores que disponibilizam esse tipo de serviço, alguns desses são: a Amazon [1], a Google [2] e a Microsoft [3], que fornecem desde serviços mais complexos como os de Plataforma como Serviço(PaaS) e Software como Serviço (SaaS), até serviços mais específicos como os de Infraestrutura como Serviço (IaaS), cada qual com vantagens e desvantagens em determinados tipos de serviços, além de distintas características quanto a virtualização, a escalabilidade, a interoperabilidade, a elasticidade e a Qualidade de Serviço (QoS), além de obterem distintos Acordos de Nível de Serviço (SLAs) [5].

É neste contexto que surge o conceito de Federação de Nuvens [1], que propõe que diferentes provedores tenham seus serviços disponibilizados de maneira completamente transparente, sob demanda, para o usuário final, provendo características adicionais a este serviço, como a migração de recursos, processamento em paralelo, replicação e fragmentação de dados, quebrando a barreira do modelo existente de nuvens pública, privada

[1] https://aws.amazon.com/
[2] https://cloud.google.com/
[3] https://azure.microsoft.com/

e híbrida [28]. Além disso, tem-se uma ampliação considerável em relação a características já citadas como escalabilidade, interoperabilidade e escalabilidade.

Dentro deste paradigma, é possível destacar algumas vantagens para certos tipos de aplicações, tais como as de Bioinformática. Esses ambientes não só provêm recursos de armazenamento e processamento escaláveis, como também um grau de transparência maior para a utilização de usuários finais.

Nesse contexto, o BioNimbuZ é um ambiente proposto e desenvolvido na Universidade de Brasília, que tem foco em execuções de *workflows* de Bioinformática, que se utiliza das vantagens de um ambiente de Nuvens Federadas, que foram apresentadas no parágrafo anterior.

Diante do exposto, a proposta deste trabalho é de uma melhoria no Serviço de Armazenamento do BioNimbuZ, que atualmente utiliza-se de serviços de infraestrutura dos provedores de nuvem para armazenar arquivos nos discos de suas instâncias de máquinas virtuais. Conforme proposto por Celesti [2] e Cachin [6], alterando-se este cenário para que o serviço de armazenamento do BioNimbuZ passe a utilizar serviços próprios de armazenamento já oferecidos pelos provedores de nuvem. Dessa forma passa-se a ter um serviço mais robusto e eficiente, pois os próprios serviços de armazenamento externos já oferecem um grau de disponibilidade alto o suficiente para os arquivos armazenados, eliminando a necessidade de que a replicação destes arquivos seja feita também pelo BioNimbuZ. Além disso, elimina-se a limitação de armazenamento que o BioNimbuZ tinha relacionada a capacidade dos discos de suas instâncias de máquina virtual, que poderiam ficar saturados, dificultando futuras execuções de *workflows*.

1.1 Motivação

Graças ao crescimento dos provedores de nuvem nos últimos anos, foi possível ter um grande aumento na variedade e na qualidade dos serviços oferecidos pelas mesmas. Portanto, a maneira escolhida para realizar o armazenamento e a replicação de arquivos nas primeiras versões do BioNimbuZ [21, 29] não se faz mais tão eficiente se comparada com as demais possibilidades. Existem serviços direcionados para armazenamento, como o Amazon S3 [4] e o Google *Cloud Storage* [5], que já oferecem funcionalidades como replicação de dados multi-região, transferências de alta velocidade e alta disponibilidade.

Diante deste contexto, a motivação deste trabalho é implementar um novo serviço de armazenamento para a plataforma de nuvem federada do BioNimbuZ que utilize como

[4] http://aws.amazon.com/s3/
[5] https://cloud.google.com/storage/

base os serviços específicos de armazenamento fornecidos pelos provedores externos de nuvem.

1.2 Problema

Atualmente, para armazenar qualquer arquivo em seu ambiente, o BioNimbuZ utiliza exclusivamente de suas instâncias de máquinas virtuais, nas quais os arquivos são salvos nos discos destas instâncias. Isto é ineficiente por que, da maneira como funciona o serviço atual, o *upload* inicial de arquivos que serão utilizados em um determinado *workflow* ocorre antes que o próprio *workflow* seja submetido. Isso significa que nesse momento ainda não há informações suficientes para que se possa escolher em qual instância da federação será executada a tarefa que utilizará este arquivo, fazendo com que, posteriormente, seja necessária a transferência interna de arquivos para outra instância. Além disso, com esta abordagem, existe um acoplamento entre os nós de processamento e de armazenamento na federação de nuvem, o que tipicamente não é desejável, e pode vir a trazer problemas e limitações em diversos cenários, assim como eventuais desperdícios de recursos computacionais. Com o serviço de armazenamento atual, a única maneira de armazenar determinado arquivo na federação até que este seja requisitado é mantendo, pelo menos, uma instância de máquina virtual criada para armazenar este arquivo, o que pode ser muito custoso.

1.3 Objetivos

1.3.1 Objetivo Geral

Este projeto tem como objetivo implementar um novo serviço de armazenamento de dados no ambiente de nuvem federada BioNimbuZ, que utilize serviços de armazenamento específicos oferecidos pelos provedores de nuvem. Desta forma, os arquivos submetidos pelos usuários serão enviados ao serviço de armazenamento próprio e poderão ser acessados de qualquer instância de máquina virtual criada na federação.

1.3.2 Objetivos Específicos

Para que o objetivo geral seja cumprido, faz-se necessário atingir os seguintes objetivos específicos:

- Definir uma arquitetura interna para o novo serviço de armazenamento do BioNimbuZ;
- Implementar o novo serviço de armazenamento no núcleo de serviços do BioNimbuZ;
- Realizar experimentos para avaliar o serviço;
- Comparar com a estratégia definida hoje.

1.4 Organização do Trabalho

Este trabalho foi dividido em mais cinco capítulos, além deste introdutório. No Capítulo 2 são apresentadas as definições e os conceitos fundamentais para a compreensão do cenário atual de computação em nuvem, indo desde os conceitos básicos de sistemas distribuídos até a definição e diferenciação de *clusters*, *grids* e nuvens computacionais. Além disso, este capítulo detalha as principais características da federação de nuvem.

O Capítulo 3 apresenta a plataforma de nuvens federadas BioNimbuZ, detalhando as tecnologias utilizadas em sua concepção, sua arquitetura e o funcionamento de todos os seus serviços internos.

No Capítulo 4 é mostrado o processo de escolha dos serviços de armazenamento externos utilizados na construção do serviço de armazenamento proposto por este trabalho, detalhando as diferenças entre os tipos de serviço de armazenamento encontrados, o armazenamento por blocos e o armazenamento por objetos, e apresentando seus pontos positivos e negativos. Em seguida, este capítulo descreve as responsabilidades e funcionamento do novo serviço de armazenamento.

No Capítulo 5 são apresentados os resultados obtidos nos testes realizados com o novo serviço de armazenamento implementado no BioNimbuZ. O objetivo destes testes é comprovar o funcionamento do serviço proposto e verificar certos aspectos comparados aos do serviço de armazenamento anteriormente presente no BioNimbuZ.

Por fim, o Capítulo 6 apresenta as conclusões obtidas da implementação e teste do serviço proposto e propõe trabalhos futuros, que podem expandir o que foi proposto neste trabalho.

Capítulo 2

Computação em Nuvem

O objetivo deste capítulo é apresentar, definir e diferenciar os principais tipos de sistemas distribuídos. A Seção 2.1 contém uma breve explanação sobre o que são sistemas distribuídos, quais suas características e como eles funcionam. Em seguida, nas Seções 2.2, 2.3 e 2.4, serão explicados os três tipos de sistemas distribuídos, o *cluster*, o *grid* e a computação em nuvem. Por último, na Seção 2.5, serão apresentadas as nuvens federadas, sua definição, seus objetivos e sua arquitetura.

2.1 Sistemas Distribuídos

Juntamente com o rápido avanço dos computadores, vem crescendo a demanda para aplicações cada vez mais pesadas e custosas computacionalmente. Com isto, notou-se que a utilização de arquiteturas monoprocessadas já não estavam mais atendendo as necessidades destas aplicações de maneira satisfatória, gerando tempos de execução extremamente elevados e muitas vezes impraticáveis. Para este tipo de problema existem três abordagens de solução:

- Otimizar o algoritmo;
- Ampliar a frequência de *clock* dos processadores;
- Paralelizar as tarefas.

A primeira opção muitas vezes não é trivial ou possível, especialmente para aplicações que não dependam apenas da otimização do algoritmo, mas envolvam também um grande processamento de dados e variáveis, como é o caso de softwares de previsão climática, tectônica e oceânica, e softwares de à Bioinformática.

Para a segunda opção, mostrou-se que era muito viável sua aplicação durante muitos anos, como descrito pela Lei de Moore [20], que previa que o número de transistores em

um *chip* dobraria a cada 24 meses, impactando, consequentemente, em suas frequências de *clock*. Entretanto, esta abordagem já enfrenta uma limitação física [19], visto que o projeto de processadores vem apresentando dificuldades com a dissipação do calor gerado pela proximidade dos transistores.

A terceira opção, que vem se mostrando a mais eficaz ao longo dos últimos anos, teve início com a era dos processadores *multi-core* [3], que consiste na divisão das tarefas maiores e mais complexas em tarefas menores que possam ser distribuídas entre diferentes processadores ou *cores* para que possam ser executadas de maneira simultânea, conforme pode ser visto na Figura 2.1. Essa solução tipicamente gera uma diminuição drástica no tempo total necessário para a realização completa desta tarefa, se comparado à execução mono-processada da mesma tarefa.

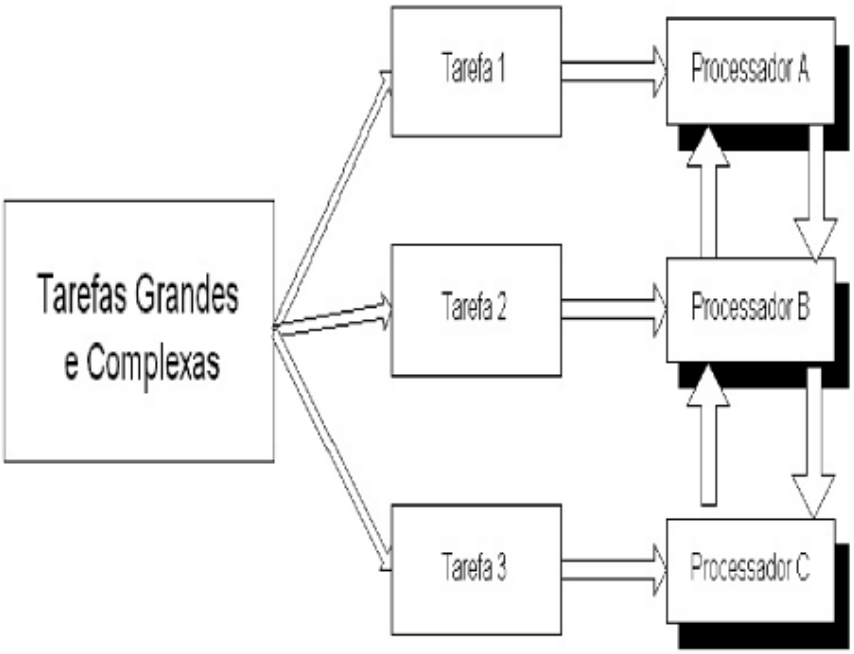

Figura 2.1: Tarefas Paralelizadas [17].

Além disso, este conceito deu origem também aos sistemas distribuídos. Na literatura existem várias definições do que é um sistema distribuído. Segundo Tanenbaum e Steen [18], um sistema distribuído é uma coleção de computadores independentes entre si que se apresenta ao usuário como um sistema único e coerente. Para Pitanga [17], um sistema distribuído é um conjunto de elementos que se comunicam através de uma rede de interconexão e que utilizam software de sistema distribuído. Cada elemento é composto por um ou mais processadores e uma ou mais memórias. Kindberg et al. [8] definiram um sistema distribuído como sendo aquele em que componentes localizados em compu-

tadores interligados em rede se comunicam e coordenam suas ações apenas por troca de mensagens.

Assim, é possível observar através das definições anteriores que existem certos aspectos a serem ressaltados acerca dos sistemas distribuídos, os principais são:

- **Concorrência:** ao dividir tarefas entre múltiplos elementos computacionais, muitas vezes é necessário que haja o compartilhamento de recursos ou uma certa ordenação na realização das tarefas dependentes. Tudo isso deve ser tratado para que não ocorram erros de sincronia ou dependência, o que leva ao aumento do nível de complexidade da programação associada;
- **Tolerância a falhas:** num cenário real, é comum que ocorram falhas em diversos pontos, um sistema distribuído deve ser capaz de lidar com falhas como atrasos de mensagem, ausência de rede ou falha em um determinado computador sem que o sistema seja interrompido como um todo;
- **Ausênsia de *clock* global:** para a cooperação entre mais de um processo em computadores diferentes é necessário que haja uma coordenação proveniente da troca de mensagens para que haja uma noção compartilhada de tempo, visto que não é possível que ocorra um *clock* global entre estes computadores. Porém, existe uma limitação para esta coordenação realizada por troca de mensagens;
- **Heterogeneidade:** em um sistema distribuído, é interessante que exista uma certa flexibilidade em diversos aspectos, como no tipo de comunicação de cada rede, arquitetura e sistema operacional de cada computador e linguagem de programação utilizada em cada aplicação. Para que isso seja possível, é necessária a implementação de diversos protocolos de comunicação, assim como a presença de uma camada intermediária entre o usuário, as aplicações e os sistemas operacionais, responsável por abstrair e mascarar tais diferenças. Esta camada é comumente chamada de *middleware*, e pode ser vista na Figura 2.2;
- **Transparência:** Um aspecto essencial nestes sistemas é a transparência, pois é necessário que para o usuário haja uma abstração de que todos os recursos e os processos estão presentes em apenas uma máquina. A transparência existente em sistemas distribuídos pode ser classificada conforme descrito na Tabela 2.1.

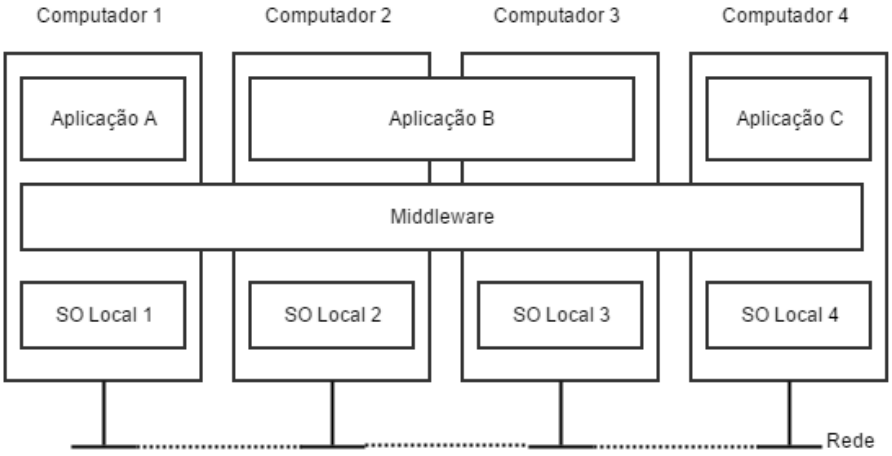

Figura 2.2: Representação de um *Middleware*, adaptado de [18].

Tabela 2.1: Tipos de Transparência em um Sistema Distribuído [18].

Transparência	**Descrição**
Acesso	Esconde diferenças na representação de dados e em como um recurso é acessado.
Localização	Esconde onde um recurso está localizado.
Migração	Esconde que um recurso pode se mover para outra localização.
Realocação	Esconde que um recurso pode ser movido para outra localização enquanto em uso.
Replicação	Esconde que um recurso está replicado.
Concorrência	Esconde que um recurso pode ser dividido entre vários usuários simultaneamente.
Falha	Esconde a falha e recuperação de um recurso.

A seguir, nas próximas seções deste capítulo, serão descritos e diferenciados os principais tipos de Sistemas Distribuídos, que são: o *Cluster* Computacional, o *Grid* Computacional e a Nuvem Computacional.

2.2 *Cluster* Computacional

Cluster é um tipo de Sistema Distribuído, e consiste em um grupo de computadores, chamados de nós, que trabalham juntos. Segundo Sloan [14], um *cluster* possui três elementos básicos:

- Uma coleção de computadores individuais;
- Uma rede conectando estes computadores;
- Um software que permite que um computador divida o trabalho entre os demais computadores através desta rede.

Esta rede se trata tipicamente de uma rede de alta velocidade, permitindo que os computadores trabalhem como uma única máquina lógica para o usuário, e que executem tarefas de alta complexidade de maneira eficiente [9].

Tipicamente em um *cluster*, todos os nós executam instâncias de um mesmo sistema operacional e possuem o mesmo hardware, porém existem exceções para ambos os casos, permitindo uma maior flexibilidade para o *cluster*, mas consequentemente tornando mais complexo o *middleware* presente entre o usuário e os nós.

É comum que se pense que um *cluster* é simplesmente um conjunto composto por vários computadores interconectados. Porém, existem certos detalhes acerca da sua estrutura interna que devem ser levados em consideração, em especial acerca do papel individual de cada nó. Existem distintas abordagens possíveis para um *cluster*, as três abordagens apresentadas por Sloan [14] são: *Cluster* Simétrico, *Cluster* Assimétrico e *Cluster* Expandido.

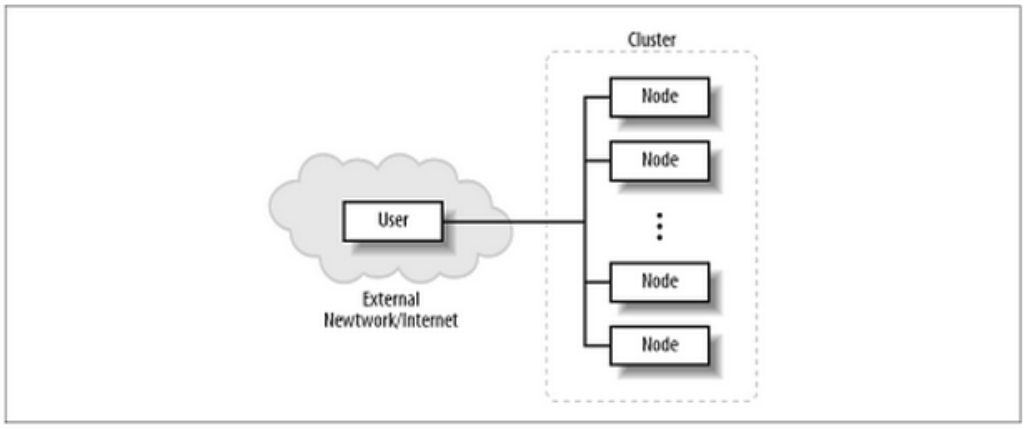

Figura 2.3: Arquitetura de um *Cluster* Simétrico [14].

O *Cluster* Simétrico, que é a abordagem mais simples de todas, onde cada nó funciona como um computador individual. Isto é extremamente fácil de se configurar, porém torna difícil as tarefas de gerenciamento do *cluster*, como distribuição de carga e aplicação de políticas de segurança. É possível ver esta abordagem representada na Figura 2.3.

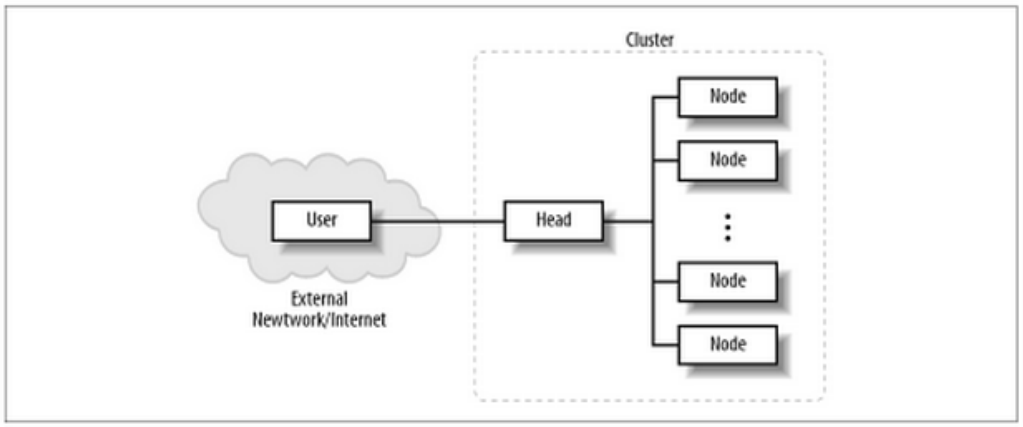

Figura 2.4: Arquitetura de um *Cluster* Assimétrico [14].

Para *cluster*s dedicados, a abordagem assimétrica, apresentada na Figura 2.4, é a mais utilizada. Nela existe um nó denominado *head*, que funciona como uma porta de acesso entre o usuário e os demais nós. Como todo o tráfego deve passar através do nó *head*, as complicações como segurança e distribuição de carga, citadas anteriormente, deixam de ser um problema, porém passa-se a ter complicações relacionadas a desempenho, uma vez que o *head* poderá gerar um gargálo no *cluster* se este não for capaz de transmitir todas as informações do usuário para os demais nós a uma taxa suficientemente alta.

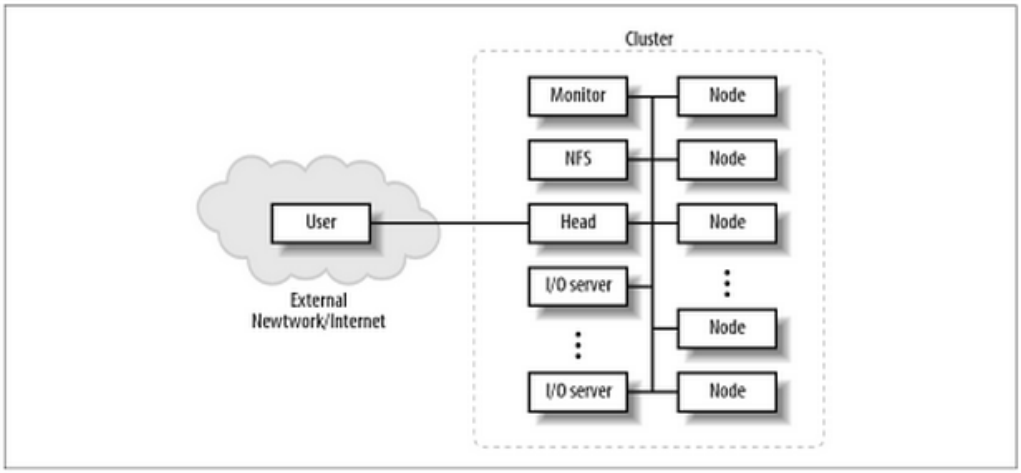

Figura 2.5: Arquitetura de um *Cluster* Expandido [14].

A terceira abordagem apresentada é a do *Cluster* Expandido, visto na Figura 2.5. Nela procura-se modularizar as tarefas de monitoramento do *cluster*, descentralizando as funções do nó *head* ao se denominar outros nós para funções de monitoramento mais custosas e importantes.

Clusters geralmente são restritos a computadores relativamente próximos, em uma mesma sub-rede ou LAN, que é a sua principal diferença em relação aos *Grids* e Nuvens Computacionais, que serão abordados a seguir neste capítulo.

2.3 *Grid* Computacional

O termo *grid* é frequentemente utilizado para descrever computadores trabalhando juntos através da Internet ou de uma WAN, e vem de uma analogia com as redes elétricas (do inglês *power grids*), por se tratar de recursos presentes em locais espaçados geograficamente e que são acessados de maneira simples e natural, independente de como foram gerados e de onde são provenientes. Segundo Sloan [14], um *grid* é um sistema distribuído descrito como uma coleção de computadores que provêm poder computacional como uma *commoditiy*[1].

As principais diferenças entre um *grid* e um *cluster* está no fato de que *grids* são tipicamente de escala muito superior e disperso geograficamente, o que tende a gerar um acesso de maneira assíncrona e um número maior de acessos, o que também gera problemas de segurança e autenticação. É também mais comum que se tenha uma alta heterogeneidade entre os componentes de um *grid*, tanto em nível de software quanto hardware, o que não era tão comumente visto nos *clusters*.

Apesar de extremamente poderosas e muito utilizadas, aplicações a serem executadas em *grids* possuem um desenvolvimento extremamente complexo, devido à sua alta flexibilidade gerada pela heterogeneidade citada anteriormente, assim como os problemas de assincronia, escalonamento, espaçamento e segurança.

Foster et al. [11] apresentam um arquitetura para um *grid* dividida em camadas, a qual visa simplificar as interações entre aplicações e a infraestrutura. A arquitetura proposta por Foster et al. [11] pode ser vista na Figura 2.6.

A camada mais abaixo representa a infraestrutura, e é nela que se encontram os recursos computacionais físicos presentes no *grid*. Todas as camadas entre a infraestrutura e a aplicação são o que compõe o *middleware*, conforme citado anteriormente na Seção 2.1. A camada de conectividade é a responsável pela comunicação entre os recursos através de protocolos para transmissão de dados em redes. A camada de recursos, por sua vez, é a responsável por inicializar e controlar o compartilhamento de recursos individuais do *grid*. Na camada de serviços coletivos estão presentes serviços como escalonamento, monitoramento e replicação de dados. A camada de aplicação deve prover uma abstração

[1]Chama-se de *commoditiy* neste contexto qualquer mercadoria produzida em massa para venda em transações comerciais.

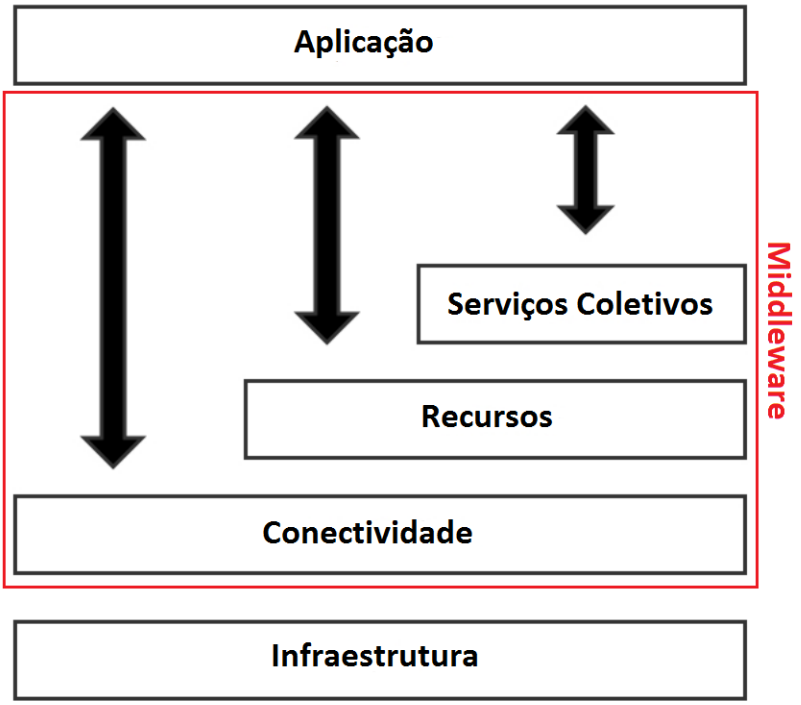

Figura 2.6: Arquitetura de um *Grid*, adaptado de [11].

para que aplicações de usuários possam ser executadas de maneira distribuída, sem que seja necessário um conhecimento profundo de cada componente envolvido nesta tarefa.

2.4 Nuvem Computacional

A computação em nuvem é um paradigma de computação distribuída que surgiu com a necessidade de prover serviços através dos quais seja possível usuários acessarem aplicações e recursos sob demanda de qualquer lugar do mundo [12]. Ainda não existe um consenso no que se diz respeito à definição de computação em nuvem, portanto serão apresentadas a seguir algumas das definições mais aceitas na literatura:

- Para Foster et al. [12], computação em nuvem é um paradigma computacional de larga escala que é impulsionado por economias de escala, no qual um conjunto de poder computacional, plataformas e serviços abstratos, virtualizados, dinamicamente escaláveis e gerenciados são entregues sob demanda para consumidores externos através da internet;
- Segundo Buyya et al. [24], nuvem é um tipo de sistema paralelo e distribuído que consiste em uma coleção de computadores virtualizados e inter-conectados, que são dinamicamente provisionados e apresentados como um único ou mais recursos

computacionais uniformes, baseados em Acordos de Nível de Serviço (SLA), estabelecidos através de negociações entre provedores de serviço e consumidores;

- Caceres et al. [15] propõem através de um estudo envolvendo mais de 20 definições de nuvem, que sua definição abrangente se trata de um grande conjunto de recursos virtualizados de fácil uso e acesso, como hardware, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente reconfigurados para se ajustar a uma carga variável, permitindo também uma otimização da utilização destes recursos. Esse conjunto de recursos é tipicamente explorados através de um modelo *pay-per-use* nos quais existem garantias são oferecidas pelo provedor de infraestrutura através de Acordos de Nível de Serviço;

- A definição oficial do NIST (*National Institute of Standards and Technology*), escrita por Mell e Grance [23], diz que computação em nuvem é um modelo para permitir um acesso a rede onipresente, conveniente e sob demanda para que se possa compartilhar um conjunto de recursos computacionais configuráveis, como redes, servidores, aplicações e serviços, que podem ser rapidamente provisionados e liberados com esforço mínimo de gerência.

Nesse contexto, nota-se através destas definições que existem certas semelhanças entre nuvens e *grids*, o que se deve ao fato de que a origem da computação em nuvem é proveniente do avanço e da evolução da computação em *grid* [12]. Existem, entretanto, diferenças importantes entre os dois. Na computação em *grid*, o principal produto disponibilizado é a infraestrutura. Já na computação em nuvem, infraestrutura é apenas um dos serviços providos, juntamente com diversos outros serviços, como armazenamento, processamento e software.

Assim, na nuvem os recursos são acessados de maneira mais abstrata e transparente, em uma escala ainda maior que nos *grids*, o que permite um acesso global e conveniente destes recursos através da Internet. Além disso, pode-se destacar outras características principais que diferem nuvens computacionais dos outros modelos de sistemas distribuídos, mencionados anteriormente: o modelo de pagamento sob demanda, a elasticidade e a virtualização.

O modelo de pagamento sob demanda significa redução de custos para empresas consumidoras destes serviços, uma vez que só se paga pelo que se consome, ao contrário de taxas fixas tipicamente presentes nos outros sistemas. Elasticidade é a capacidade de um determinado sistema de adicionar e remover recursos de acordo com a necessidade, acompanhando o crescimento ou a diminuição da taxa de consumo destes. Por último, a virtualização é o que torna possível a alta heterogeneidade presente nas nuvens, pois

provê uma abstração do acoplamento entre hardware e software através da virtualização de recursos físicos.

2.4.1 Arquitetura de uma Nuvem

A arquitetura de uma nuvem é composta por uma série de elementos presentes entre a infraestrutura e o usuário, que são responsáveis pela abstração e o gerenciamento do sistema em si. Existem na literatura, várias propostas de arquitetura para nuvem [12, 15]. A arquitetura proposta por Foster [12] pode ser vista na Figura 2.7.

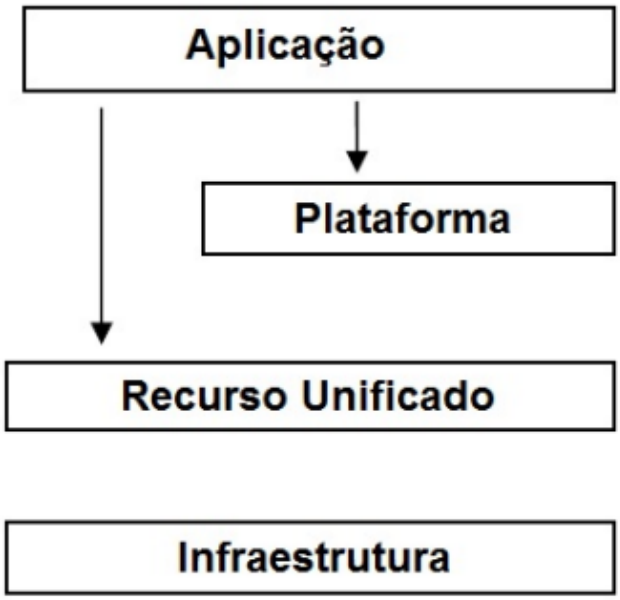

Figura 2.7: Arquitetura de uma Nuvem, segundo Foster [12].

De modo análogo aos *grids*, trata-se de uma arquitetura dividida em camadas, onde a camada de infraestrutura contem os recursos computacionais físicos da nuvem. A camada recursos unificados é onde ficam todos os recursos de forma capsulada, ou seja, virtualizados para que possam ser disponíveis de maneira abstrata tanto ao usuário final quanto a camada superior. A camada de plataforma adiciona uma coleção de ferramentas especializadas, *middleware* e serviços sobre os recursos unificados providos pela camada inferior, como serviços de escalonamento e hospedagem de um ambiente web. Por fim, a camada de aplicação contém as aplicações a serem executadas.

Outra proposta bem aceita é a apresentada por Caceres [15], conforme visto na Figura 2.8. Nesta proposta é possível ver a presença de três atores: Prestadores de Infraestrutura, Provedores de Serviços e Usuários de Serviços. Prestadores de infraestrutura disponibilizam um ambiente completo com recursos físicos para que provedores de serviço

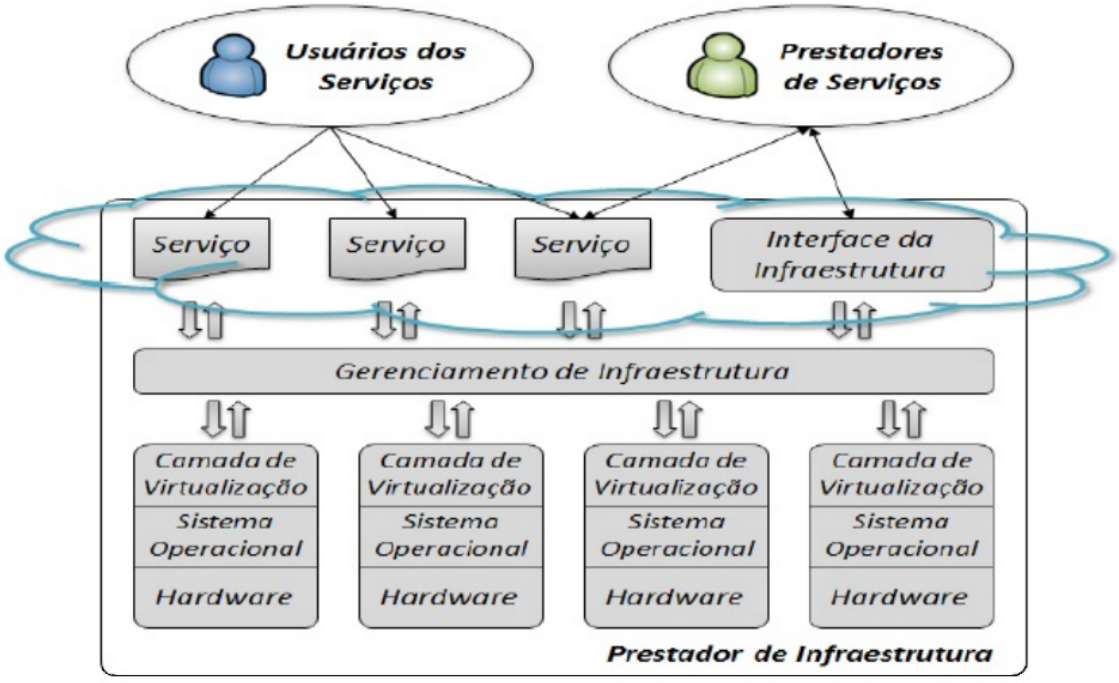

Figura 2.8: Arquitetura de uma Nuvem, segundo Vaquero et al. [15].

hospedem suas aplicações sem que precisem se preocupar com questões de estrutura física. Usuários de serviços acessam aplicações oferecidas através da Internet pelos provedores de serviços [15].

Essa arquitetura é dividida em três camadas, as quais são a camada inferior, a camada superior e a camada intermediária. A camada inferior é a camada de infraestrutura, que de maneira análoga a proposta anterior, é onde ficam todos os componentes físicos da nuvem. A camada superior é a camada de aplicação, cujo nível de abstração é maior, onde ficam as aplicações disponíveis a serem executadas na nuvem. A camada intermediária é a camada de plataforma, que oferece um ambiente para o provedor de serviço, sem que esse precise se preocupar com recursos físicos ou softwares, tornando mais simples a tarefa de controlar os sistemas e os ambientes necessários para que um serviço seja desenvolvido, testado e hospedado na web para usuários de serviços.

2.4.2 Serviços em Nuvem

Serviços providos em ambientes de computação em nuvem são classificados, a fim de tornar tais serviços uniformes para aplicações de nuvens. Embora seja possível encontrar na literatura classificações divididas em mais de três tipos [5], como pode ser visto na Figura 2.9, existem três tipos mais comumente aceitos e abordados [15, 12, 23], e estes são:

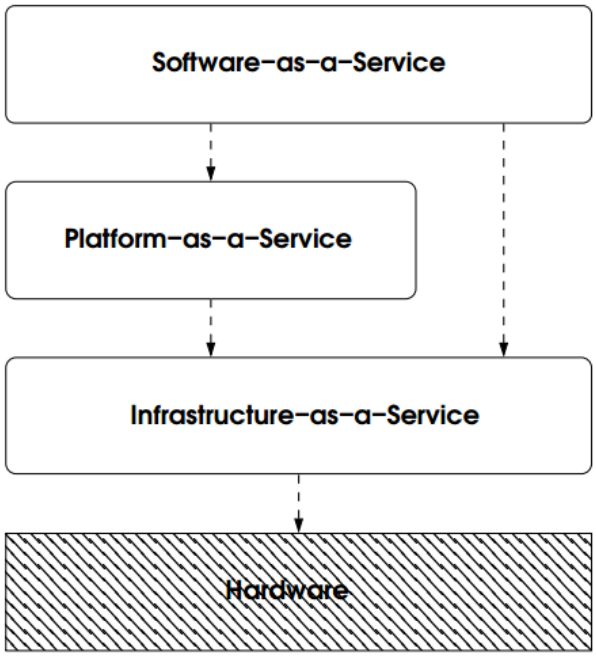

Figura 2.9: Arquitetura de Serviços em Camada da Computação em Nuvem [29].

- ***Infrastructure as a Service* (IaaS):** serviços que fornecem um ambiente, composto tanto de hardware quanto de software, para que uma aplicação seja executada, baseado no Acordo de Nível de Serviço equivalente ao uso da infraestrutura. O pagamento é proporcional ao uso dos recursos, que podem ser de processamento, de armazenamento ou de largura de banda. Exemplos de provedores de serviço IaaS são a *Amazon Elastic Compute Cloud* (EC2) [2], o GoogleFS [3], o Google Drive [4] e o Dropbox [5].
- ***Platform as a Service* (PaaS):** oferecem um ambiente de alto nível com kits de desenvolvimento para criação, testes e manipulações de aplicações. Usuário não ne-

[2]http://aws.amazon.com/pt/ec2/
[3]http://research.google.com/archive/gfs.html
[4]https://drive.google.com/
[5]http://www.dropbox.com/

cessitam ter tais kits instalados em suas máquinas localmente, possibilitando assim o desenvolvimento de aplicações para nuvens, que utilizam APIs fornecidas pelos provedores de serviços. Tais serviços geram uma escalabilidade muito alta a desenvolvedores de maneira instantânea, sem necessidade de trabalho com aquisição, instalação, configuração e manutenção de hardware e de software específicos. Exemplos de provedores de serviço PaaS são a Microsoft Azure [6] e a Google AppEngine [7].

- ***Software as a Service* (SaaS):** aplicações disponibilizadas para acesso e utilização de usuários através da Internet fornecidas como serviço, de maneira remota, sem a necessidade de instalação local. Esse tipo de serviço pode utilizar ambientes de programação fornecidos por serviços PaaS, ou podem utilizar diretamente a infraestrutura fornecida por serviços IaaS. Assim, pode haver uma cobrança relacionada ao uso de tais serviços. Exemplos de provedores de serviço SaaS são o Google Drive e a Oracle Cloud [8].

2.4.3 Tipos de Nuvens

Nuvens computacionais podem ser classificadas em tipos, dependendo da necessidade da aplicação a ser implementada e do nível de segurança desejado. Os três tipos são nuvens públicas, nuvens privadas e nuvens híbridas [28].

- **Nuvens Públicas:** o modelo de nuvens públicas permite o acesso de vários usuários através de interfaces web utilizando *browsers* comuns. Trata-se de um modelo tipicamente *pay-per-use*, mais flexível. Nuvens públicas são menos seguras que os demais modelos, visto que a necessidade de oferecer aplicações e dados a um grande volume de usuários de maneira pública é inversamente proporcional ao esforço necessário com autenticações e medidas de segurança para impedir ataques maliciosos. Portanto, acordos de confiança e de segurança, assim como Acordos de Nível de Serviço são essenciais para modelos de nuvens públicas.
- **Nuvens Privadas:** são nuvens configuradas de maneira interna no *datacenter* de uma empresa. É mais fácil de se realizar o alinhamento de questões de segurança, conformidade e de requisitos regulamentares, e provê à empresa um controle mais completo da implementação e uso da nuvem. Em modelos privados, recursos escaláveis e aplicações virtuais fornecidas pelo fornecedor da nuvem são agrupados e tornados disponíveis para uso e compartilhamento entre os usuários da nuvem. Di-

[6] https://azure.microsoft.com/
[7] https://cloud.google.com/appengine/docs/whatisgoogleappengine
[8] https://cloud.oracle.com/home

fere da nuvem pública pois aplicações e recursos apenas são gerenciados pela própria empresa, de maneira semelhante ao que ocorre nas Intranets.

Figura 2.10: Diagrama de uma Nuvem Híbrida.

- **Nuvens Híbridas:** trata-se de uma nuvem privada interligada com um ou mais serviços em nuvem externos, gerenciados de maneira centralizada, provisionado como uma única unidade, e contido em uma única rede segura. Nuvens híbridas oferecem soluções virtuais através de uma mistura de ambas, privadas e públicas. Oferecem um nível de controle e segurança intermediário posicionado entre redes públicas e privadas. Possuem uma arquitetura aberta que permite interfaces com outros sistemas de gestão. A Figura 2.10 mostra um diagrama exemplo de uma núvem híbrida, utilizado pelo *VMware vCloud Connector* [9].

2.5 Federação de Nuvens

Com o passar dos anos e com o aumento exponencial no número de consumidores de serviços em nuvem no Mundo, a utilização de nuvens de maneira individual passou a não ser mais suficiente. Grandes empresas passaram a estabelecer *datacenters* ao longo do Mundo, em múltiplas localizações geográficas, a fim de fornecer redundância e confiabilidade em casos de falhas, e também visando diminuir o atraso de solicitações de requisições, reduzindo a distância entre os servidores e os usuários. Surgiu então uma necessidade de interconexão entre diferentes nuvens para que fosse possível continuar fornecendo tais serviços de maneira mais robusta, eficiente, escalável e rápida.

[9]http://www.vmware.com/br/products/vcloud-connector/features

Bittman et al. [30] sugerem que a computação em nuvem vêm seguindo uma evolução em fases, conforme exemplificado na Figura 2.11. Na primeira fase, existiam apenas grandes *datacenters* isolados, sem presença de nuvens. A segunda fase é chamada de monolítica, onde existem apenas nuvens individuais e isoladas, compostas apenas de serviços de grandes empresas. A terceira fase é composta de uma cadeia vertical, onde o foco ainda está presente em nuvens individuais de grandes proprietários, porém existe uma troca de serviços entre as nuvens, com um princípio de integração. Por fim, a quarta fase é composta por uma federalização das nuvens, isto é, pequenos e grandes provedores cooperam de maneira a oferecer serviços entre si, de maneira a aumentar a flexibilidade, a escalabilidade e a eficiência de soluções, em nuvem.

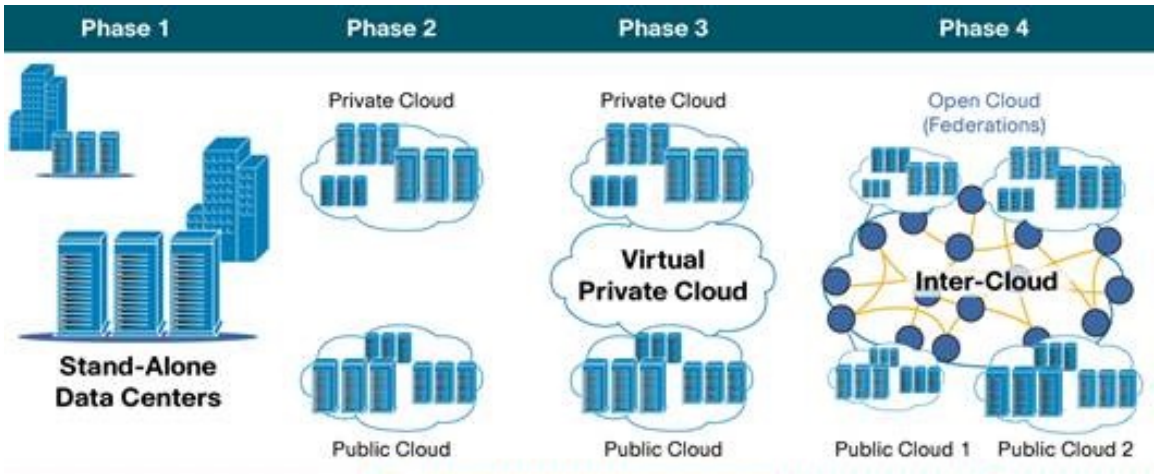

Figura 2.11: Fases da Computação em Nuvem [30].

Portanto, pode-se definir a federação de nuvens computacionais, também chamada em alguns contextos de *inter-cloud* ou *cross-cloud* [1], como um conjunto de provedores de nuvens públicos e privados, conectados através da Internet [29]. Além disto, seus principais objetivos são:

- Maximizar a eficiência na utilização de recursos, permitindo juntamente com a elasticidade que exista a impressão para os usuários de que tais recursos são ilimitados;
- Eliminar a dependência de um único provedor de infraestrutura, gerando um nível maior de confiabilidade, utilizando redundância e maior distribuição de serviços.

2.5.1 Requisitos e Desafios

Segundo Celesti et al. [1], dentre os requisitos necessários para que exista a federação, estão:

- **Automatismo e Escalabilidade:** deve existir um mecanismo de descoberta capaz de escolher qual servidor de nuvem dentre os presentes na federação é aquele que atende de melhor maneira as necessidades do usuário e capaz de se adaptar a mudanças dinamicamente;
- **Segurança Interoperável:** é necessário que haja uma integração entre as diferentes políticas e tecnologias de segurança, de forma que não seja necessário nenhuma adequação para que uma nuvem se junte à federação.

Buyya et al. [25] destacam também alguns outros problemas tipicamente enfrentados em cenários de federação:

- **Previsão de Comportamento da Aplicação:** é importante que o sistema seja capaz de prever o comportamento das aplicações para que ele possa tomar decisões eficientes relacionadas a alocação de recursos, dimensionamento dinâmico, armazenamento e largura de banda. Deve haver um modelo para realizar tais previsões, este modelo deve utilizar dados estatísticos e dados contendo padrões de uso do sistema para ajustar variáveis sempre que preciso, fazendo com que o modelo torne-se o mais próximo possível da realidade;
- **Mapeamento Flexível de Recursos e Serviços:** é essencial que o sistema seja eficiente quando for necessário encontrar a melhor configuração de hardware e de software para atender uma determinada demanda estabelecida através de um Acordo de Nível de Serviço. O comportamento imprevisível de aplicações em nuvem torna isto uma tarefa consideravelmente complexa;
- **Modelo Econômico Impulsionado por Técnicas de Otimização:** decisões orientadas ao mercado geram um problema de otimização combinatória, buscando encontrar a melhor combinação entre serviços e planos. Modelos de otimização tem como tarefa otimizar tanto a utilização e disponibilidade de recursos, como também tempo de resposta e de orçamento;
- **Integração e Interoperabilidade:** manter todos os dados de um sistema na nuvem pode não ser a escolha mais segura para o cenário de uma grande empresa, pois podem haver riscos de vazamentos de dados sigilosos através de ataques maliciosos, portanto pode ser mais sensato que se mantenham tais dados fora da nuvem;
- **Monitoramento Escalável dos Componentes do Sistema:** existe um monitoramento centralizado para gerenciamento de componentes na nuvem. Isto pode gerar um gargalo caso haja um fluxo de requisições muito elevado. Além disto, pode gerar falhas de segurança mais facilmente, pois ataques ao ponto de gerenciamento

centralizado poderiam gerar falhas em todo o restante da nuvem. Logo, nota-se que em um ambiente largo como uma federação, é necessário que tal monitoramento seja realizado de maneira distribuída.

2.5.2 Arquitetura

Para que todos os requisitos e desafios mencionados acima pudessem ser atingidos de maneira aceitável e eficiente, surgiram na literatura propostas de arquiteturas para uma federação de nuvens. Serão apresentadas a seguir duas destas propostas:

Buyya [25] propõe uma arquitetura, vista na Figura 2.12, na qual o usuário utiliza um componente externo aos provedores para realizar qualquer tipo de interação com a federação, chamado de *Cloud Broker* (CB). Este componente é responsável por intermediar a comunicação entre o usuário e os provedores da federação, e por identificar quais recursos estão disponíveis em cada provedor para que seja possível atender os recursos de qualidade de serviço (QoS) acordados previamente.

Para conseguir os dados necessários dos provedores de nuvem, o *Cloud Broker* consulta um outro componente da arquitetura, chamado de *Cloud Exchange* (CEx). A função do *Cloud Exchange* é funcionar como um registro, que será consultado por todos os *Cloud Brokers* a fim de obter informações sobre a infraestrutura, como custos de utilização, padrões de execução e recursos disponíveis, além de oferecer também serviços de mapeamento de requisições dos usuários aos provedores da nuvem que melhor as atenderiam.

Além disso, em cada infraestrutura oferecida pelos provedores existe um outro componente, chamado de *Cloud Coordinator*, que é o componente responsável por incluir a infraestrutura na federação e tornar disponíveis seus recursos aos usuários da federação. Para atender às requisições dos usuários, o *Cloud Coordinator* realiza uma autenticação e um estabelecimento de acordo de qualidade de serviço com cada um dos *Cloud Brokers*, e a partir disto, escalona as requisições para a infraestrutura. Por fim, o *Cloud Coordinator* também identifica e monitora os recursos disponíveis na federação e sua utilização, para que tais informações sejam consumidas pelo *Cloud Exchange*.

A outra proposta a ser apresentada foi proposta por Celesti [1], e pode ser vista na Figura 2.13. Nela, ao contrário do que pôde ser visto na proposta anterior, o usuário interage diretamente com um dos componentes da arquitetura disponível na infraestrutura de um provedor. Nessa arquitetura é sugerido que existem dois tipos de nuvens: local e estrangeira. Uma nuvem local consiste num provedor que está com seu recurso saturado, logo não tem mais capacidade de atender às requisições da demanda, portanto deve repassar tais requisições para as nuvens estrangeiras. Estas, por sua vez, se tratam de provedores que irão ceder seus recursos ociosos para atender à demanda que lhe foi repassada, de maneira gratuita ou não.

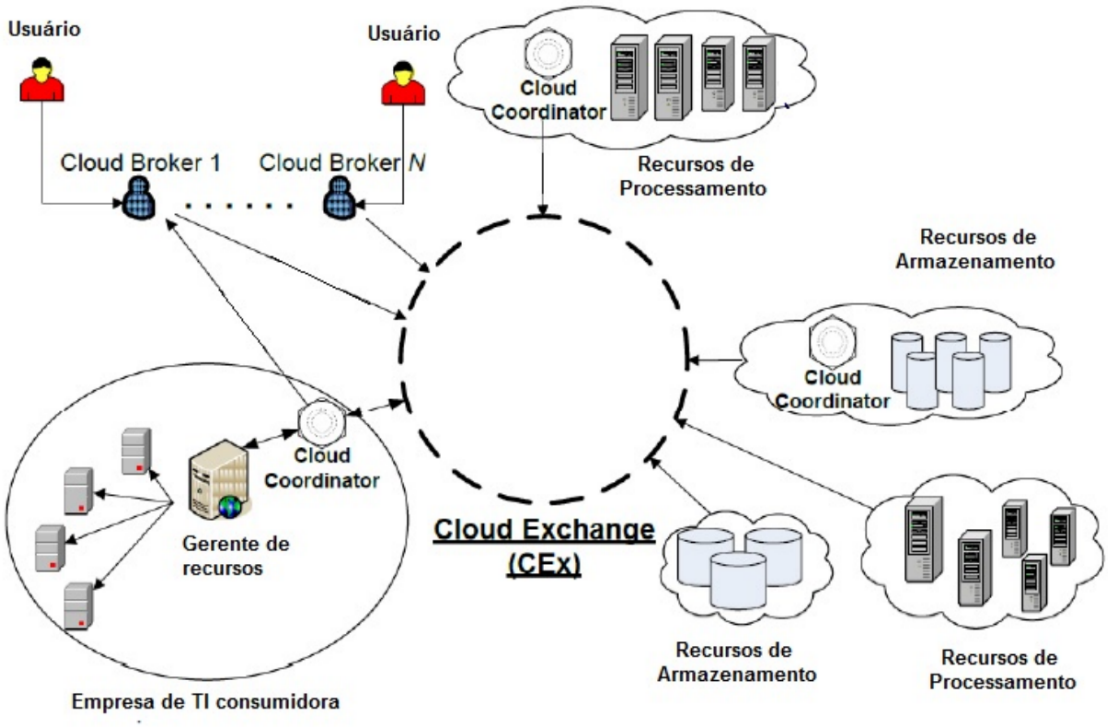

Figura 2.12: Arquitetura de uma Federação de Nuvens, segundo Buyya [25].

É interessante destacar que uma nuvem pode ser considerada tanto local quanto estrangeira em um mesmo instante de tempo, basta que ela esteja com um recurso saturado, mas possua outro determinado recurso ocioso ao mesmo tempo. Este repasse de requisições entre provedores deve ocorrer de maneira transparente para o usuário, de forma que para ele, só seja necessário um contrato de Acordo de Nível de Serviço.

Nesta arquitetura, propõe-se que haja um gerenciador, chamado *Cross-Cloud Federation Manager* (CCFM), em cada uma das infraestruturas disponibilizadas pelos provedores na federação. Tal gerenciador é responsável por realizar a gerência da nuvem e seus recursos, verificando se algum deles esteja saturado e buscando nuvens que estejam dispostas a colaborar com seus recursos que estejam ociosos. O *Cross-Cloud Federation Manager* é composto por três agentes, que realizam os três passos do procedimento de atendimento a requisição de um usuário. O *Discovery Agent* é o responsável por realizar o processo de descoberta na federação, ou seja, identificar as nuvens que estão presentes na federação e quais são os seus recursos. Tal processo deve ocorrer de maneira dinâmica e distribuída, portanto, é necessário que cada provedor de nuvem disponibilize suas informações para um ponto da nuvem federada que será consultado pelos demais provedores caso necessário. O *Match-Making Agent* é o agente cuja função é realizar a escolha de quais nuvens são mais adequadas para a realização de uma requisição de usuário. Além disso,

ele também é responsável por garantir que Acordos de Nível de Serviço sejam cumpridos. Por fim, o *Authentication Agent* é o agente responsável pela criação do canal seguro entre uma nuvem local e uma nuvem estrangeira, de forma que seja capaz que a nuvem local use os recursos da nuvem estrangeira sem que ocorram falhas na política de segurança de nenhuma das duas. Esta é uma tarefa bem complicada, uma vez que cada nuvem possui o seu próprio mecanismo de autenticação, que podem ser diferentes entre si, fazendo com que este agente realize esta interação entre tais mecanismos.

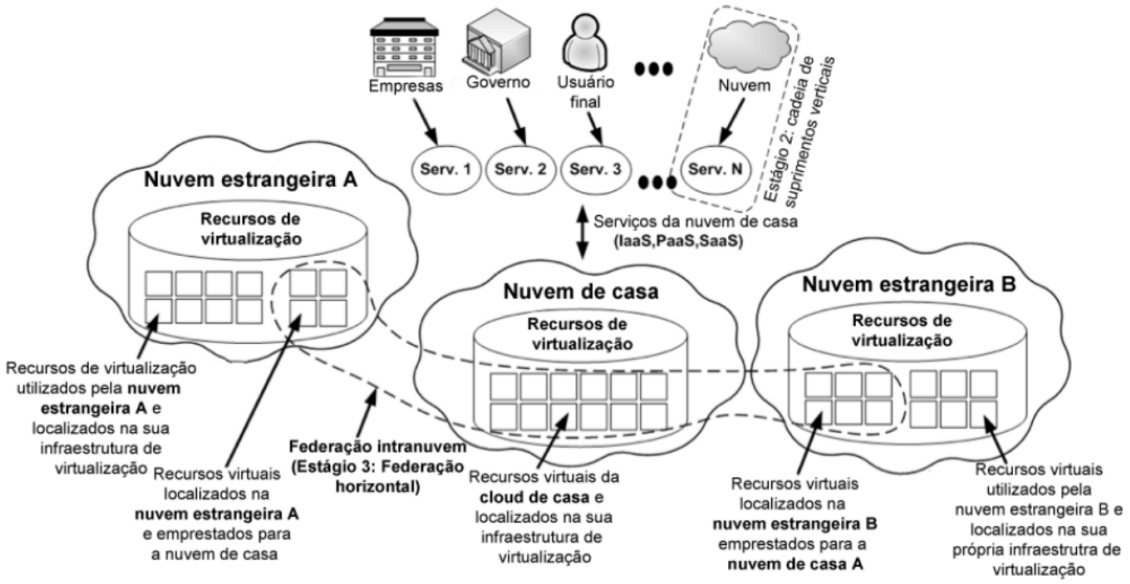

Figura 2.13: Arquitetura de uma Federação de Nuvens, segundo Celesti [1].

2.6 Considerações Finais

Neste capítulo foi apresentado o histórico da computação em nuvem, indo desde os conceitos básicos da origem da computação distribuída, até os conceitos atuais modernos sobre federações de nuvens.

No Capítulo 3 será apresentado o BioNimbuZ, plataforma de federação de nuvens, utilizada na implementação deste projeto. Para isso, serão apresentadas sua arquitetura, suas dependências e ferramentas utilizadas, e em seguida, detalhado o funcionamento de seus serviços.

Capítulo 3

Plataforma BioNimbuZ

O objetivo deste capítulo é apresentar a plataforma para federação de nuvens BioNimbuZ. Na Seção 3.1, será apresentadas a sua visão geral e as ferramentas utilizadas para sua implementação em um ambiente de nuvem distribuída. Na Seção ref3.2 sua arquitetura e seus serviços. Na Seção 3.3 será apresentada sua organização lógica e funcionamento básico. Por fim, na Seção 3.4, será apresentado com um detalhamento um pouco maior o funcionamento do serviço de armazenamento, que será explorado mais a fundo neste trabalho.

3.1 Visão Geral

Inicialmente proposto por Saldanha [29] com o nome de Bionimbus, e atualmente, após várias iterações e melhorias [26, 7, 21, 10], o BioNimbuZ é uma arquitetura de federação de nuvens com foco em aplicações de Bioinformática. O BioNimbuZ surgiu com a crescente demanda de processamento em cenários de nuvem nos últimos anos, que passou a não ser mais facilmente atendida com o uso de provedores de serviços em nuvem individuais. Isso ocorre em especial nos casos de aplicações de Bioinformática, que tipicamente exigem um poder de processamento alto e, consequentemente, longos períodos de execução.

O BioNimbuZ permite a integração e o controle de múltiplos provedores de infraestrutura em nuvem, com suas ferramentas de Bioinformática oferecidas como serviço. Isso ocorre de maneira transparente, flexível e tolerante a falhas, proporcionando, dessa forma, em teoria, um ambiente de recursos supostamente ilimitados para processamento e armazenamento. Para atingir tais objetivos, o BioNimbuZ permite que provedores de nuvens sejam independentes, heterogêneos, privados ou públicos, possam oferecer seus serviços de maneira conjunta, mantendo suas características e políticas internas, mas agindo como uma única entidade para o usuário, sem que exista uma distinção de qual provedor está sendo de fato utilizado.

Outro fator importante no BioNimbuZ é a sua flexibilidade na inclusão de novos provedores de serviço em sua federação, isso se dá devido ao uso de *plugins* de integração, que são mecanismos capazes de mapear requisições internas para o formato de requisição externa utilizado por cada provedor especificamente. Além disso, esses serviços permitem coletar informações sobre os recursos computacionais disponíveis em cada provedor. Isso é de suma importância para que se atinja determinados objetivos de uma federação de nuvens, especialmente, no que se diz respeito a escalabilidade.

Atualmente, federações de nuvens [1] são implementadas através do uso de uma camada intermediária entre a aplicação e as demais nuvens, que deve permitir a interação conjunta de várias nuvens, de maneira transparente a aplicação. Assim sendo, a proposta do BioNimbuZ é de realizar o papel desta camada intermediária.

Em suas primeiras propostas, toda a comunicação existente no BioNimbuZ era realizada por meio de uma comunicação *Peer-to-Peer* (P2P), porém as limitações relacionadas a este tipo de comunicação no que diz respeito a flexibilidade e a escalabilidade do sistema não estavam mais atendendo às demandas do BioNimbuZ, que necessitava de um maior grau de transparência na troca de informações entre diferentes máquinas virtuais da federação. Portanto, nos trabalhos de Oliveira [22] e Moura e Bacelar [21] foi proposta uma melhoria nesta comunicação. A melhoria veio através do uso de Chamadas de Procedimento Remoto, conhecidas como RPC (do inglês *Remote Procedure Call*) e seu objetivo é realizar a comunicação de forma transparente, pois tais chamadas permitem que procedimentos localizados em outra máquina sejam chamados de maneira transparente, sem que o usuário perceba [18]. O sistema RPC escolhido para tal tarefa foi o Apache Avro, da Fundação Apache, e para auxílio na coordenação e na organização dos arquivos e serviços do BioNimbuZ, utilizou-se o serviço voltado a sistemas distribuídos Apache Zookeeper, também da Fundação Apache. A seguir será explicado o funcionamento de cada um destes sistemas.

3.1.1 Apache Avro

O Apache Avro [1] é um sistema de RPC e de serialização de dados desenvolvido pela Fundação Apache como parte do projeto Hadoop [2]. O sistema oferece estruturas de dados ricas, formatos compactos e rápidos de dados binários, ou alternativamente, em formato JSON [3].

Além disso, faz uso de um *container* de arquivos para o armazenamento de dados persistentes. Existe também uma integração de maneira simples com diversas linguagens

[1] https://avro.apache.org/
[2] https://hadoop.apache.org/
[3] http://www.json.org/json-pt.html

de programação, não sendo necessário que se gere código específico para leitura e escritura de arquivos de dados ou para a implementação dos protocolos RPC. Toda a geração de código funciona como uma opção de otimização, sendo melhor aproveitadas em implementações para linguagens que utilizam tipos estáticos de dados. No que se diz respeito a protocolos de rede, existem as possibilidades de uso tanto do protocolo HTTP como de um protocolo próprio do Avro.

O sistema Avro foi criado com o conceito de utilização de grandes volumes de dados, e foi escolhido como *middleware* a ser utilizado para RPC no BioNimbuZ por se tratar de um software livre e flexível, e também por sua grande facilidade na integração com diversas linguagens, o que tipicamente é algo vantajoso em cenários de sistemas em nuvem, devido a possíveis heterogeneidades. A figura 3.1 representa o funcionamento de uma operação de *upload*, na qual se observa [21]:

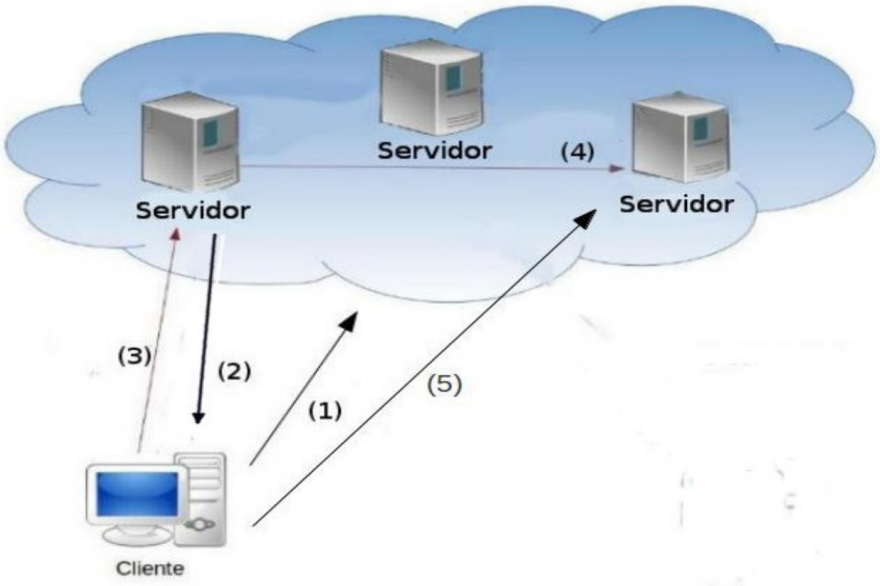

Figura 3.1: Chamadas RPC Passo-a-Passo no BioNimbuZ [21].

1. O cliente faz uma conexão com o servidor na federação;
2. O cliente recebe como resposta uma lista com os servidores disponíveis para receber o arquivo;
3. O cliente calcula a latência entre ele e os servidores da lista e envia uma chamada RPC para o servidor em que ele está conectado contendo os dados gerados referentes a esta ação de *upload*, e que são em seguida utilizados por este servidor para determinar qual o servidor que irá receber o arquivo;

4. O servidor conectado ao cliente faz uma chamada RPC para o servidor escolhido como destino do *upload* contendo os dados do arquivo que o cliente pretende enviar;
5. Por fim, o cliente abre uma conexão RPC com o servidor de destino e envia o arquivo.

3.1.2 Apache Zookeeper

O Apache Zookeeper [4] é um sistema de coordenação para sistemas distribuídos de código-aberto criado pela Fundação Apache, também como parte do projeto Hadoop. Trata-se de um sistema simples que permite que processos presentes em uma estrutura distribuída coordenem-se por meio de uma estrutura hierárquica, semelhante a uma estrutura de diretórios padrão.

O modelo de serviço do Zookeeper consiste em vários servidores que podem se conectar e comunicar entre si através de um servidor denominado líder, no qual acontece a centralização de toda a comunicação do sistema, assim é possível ver este modelo exemplificado na Figura 3.2. Caso o servidor líder seja encerrado por algum motivo, é imediatamente feita uma nova eleição através de um algoritmo do Zookeeper e um dos servidores restantes passa a ser o novo líder.

No Zookeeper, os dados presentes na estrutura hierárquica são chamados de *znodes* e estes funcionam de maneira semelhante aos arquivos e diretórios. Cada *znode* pode armazenar até 1 Megabyte (MB) de informação útil, e podem ser identificados através de seu caminho completo na estrutura hierárquica de árvore do Zookeeper, como mostrado na Figura 3.3. Diferente de como ocorre em um sistema de arquivos comum, no Zookeeper os dados são mantidos na memória, e não no disco rígido, tornando, consequentemente, suas operações de leitura e escrita extremamente mais rápidas. Assim, podem existir dois tipos de *znodes*:

- ***Znodes* Persistentes:** são os *znodes* que continuam a existir mesmo após o encerramento do servidor, de maneira persistente, permitindo que se armazene informações importantes e que se deseje manter em tais cenários. Estes *znodes* serão excluídos apenas através de exclusões explícitas;
- ***Znodes* Efêmeros:** são *znodes* que são excluídos a cada vez que se encerra o servidor. Estes são úteis para armazenar informações que não se deseja manter após quedas ou encerramentos no servidor, ou até que seja interessante que sua exclusão possa ser percebida pelos demais servidores do sistema.

[4] https://zookeeper.apache.org/

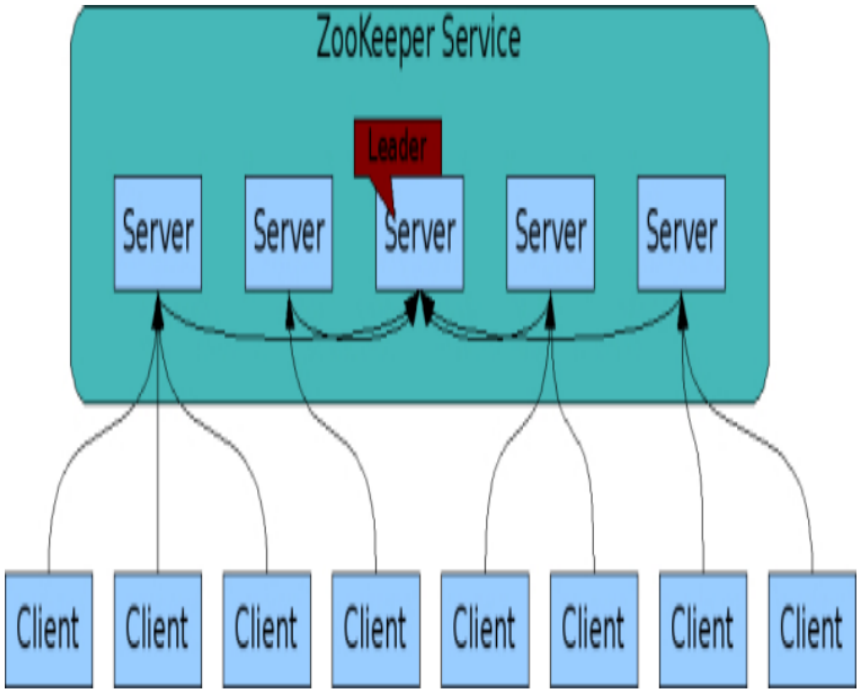

Figura 3.2: Modelo de Serviço do Apache Zookeeper.

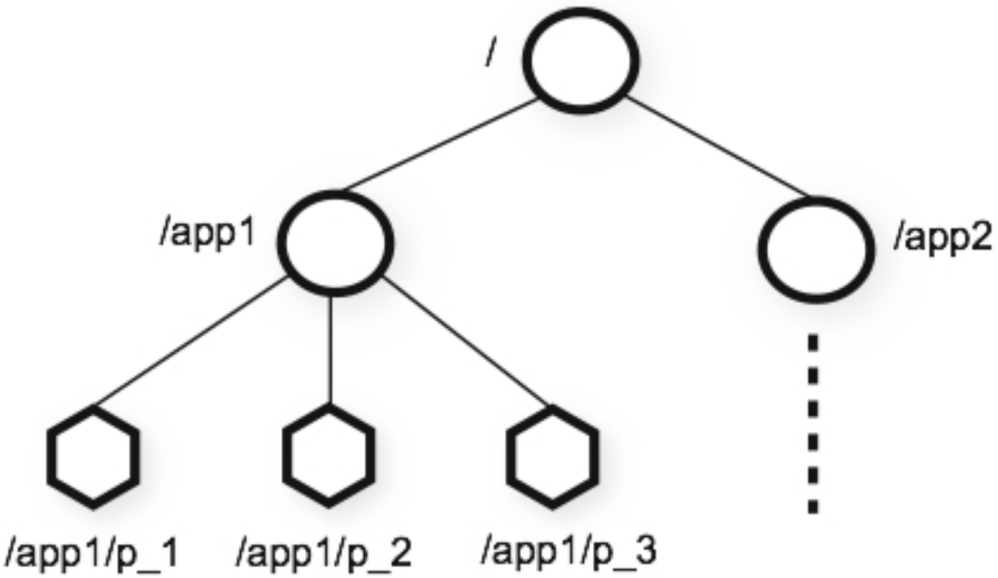

Figura 3.3: Estrutura Hierárquica dos *Znodes*.

Além disso, outra importante e poderosa funcionalidade do Zookeeper é o seu conceito de *watchers*, que são entidades observadoras definidas pelos clientes, e que lançam avisos a cada alteração ou exclusão de um determinado *znode* que se deseje observar. Isso ocorre

apenas uma vez, ou seja, cada *watcher* criado irá deixar de existir uma vez que este for disparado e o mesmo já tenha informado seu cliente solicitante sobre a alteração, sendo então necessária uma criação de um novo *watcher* a partir deste instante caso ainda se deseje observar alterações neste *znode* em questão. Isto é muito útil em diversos cenários, pois permitem um monitoramento passivo do sistema, não sendo necessárias constantes checagens manuais nestes determinados pontos.

Outra importante característica do Zookeeper é que o mesmo apresenta uma interface de programação (do inglês *Application Programming Interface* - API) extremamente simples e funcional, oferecendo grandes funcionalidades através de operações simples e compactas, tais como:

- ***create:*** cria um *znode*;
- ***delete:*** exclui um *znode*;
- ***getData:*** recupera os dados armazenados em um *znode*;
- ***setData:*** armazena dados em um *znode*;
- ***getChildren:*** obtém uma lista contendo todos os *znode* filhos de um determinado *znode*;
- ***exists:*** verifica a existência de um *znode*;

3.2 Arquitetura do BioNimbuZ

O BioNimbuZ faz uso de uma arquitetura hierárquica distribuída composta por quatro camadas, as quais são a camada de aplicação, a camada de integração, a camada de núcleo e a camada de infraestrutura, conforme visto na Figura 3.4.

Nestas camadas estão dispostas todas as funções do BioNimbuZ, iniciando na interação com o usuário, atualmente realizada através de uma aplicação *web*, que recebe os *workflows* a serem executados na federação de nuvens, o que ocorre na camada de aplicação. A camada de integração é a responsável por integrar as camadas de aplicação e de núcleo. A camada de núcleo é a responsável por realizar todo o gerenciamento interno do sistema, como os serviços de escalonamento e de armazenamento. E por fim, a camada de infraestrutura é responsável por mapear as requisições feitas pelos serviços do núcleo para os provedores de nuvens externos através dos *plugins* de integração, que são os mecanismos utilizados pelo BioNimbuZ para mapear as requisições internas da federação para o formato específico de requisição externa a ser enviado para cada provedor. A seguir serão descritas, com um detalhamento um pouco maior, as funções e características de cada uma destas camadas.

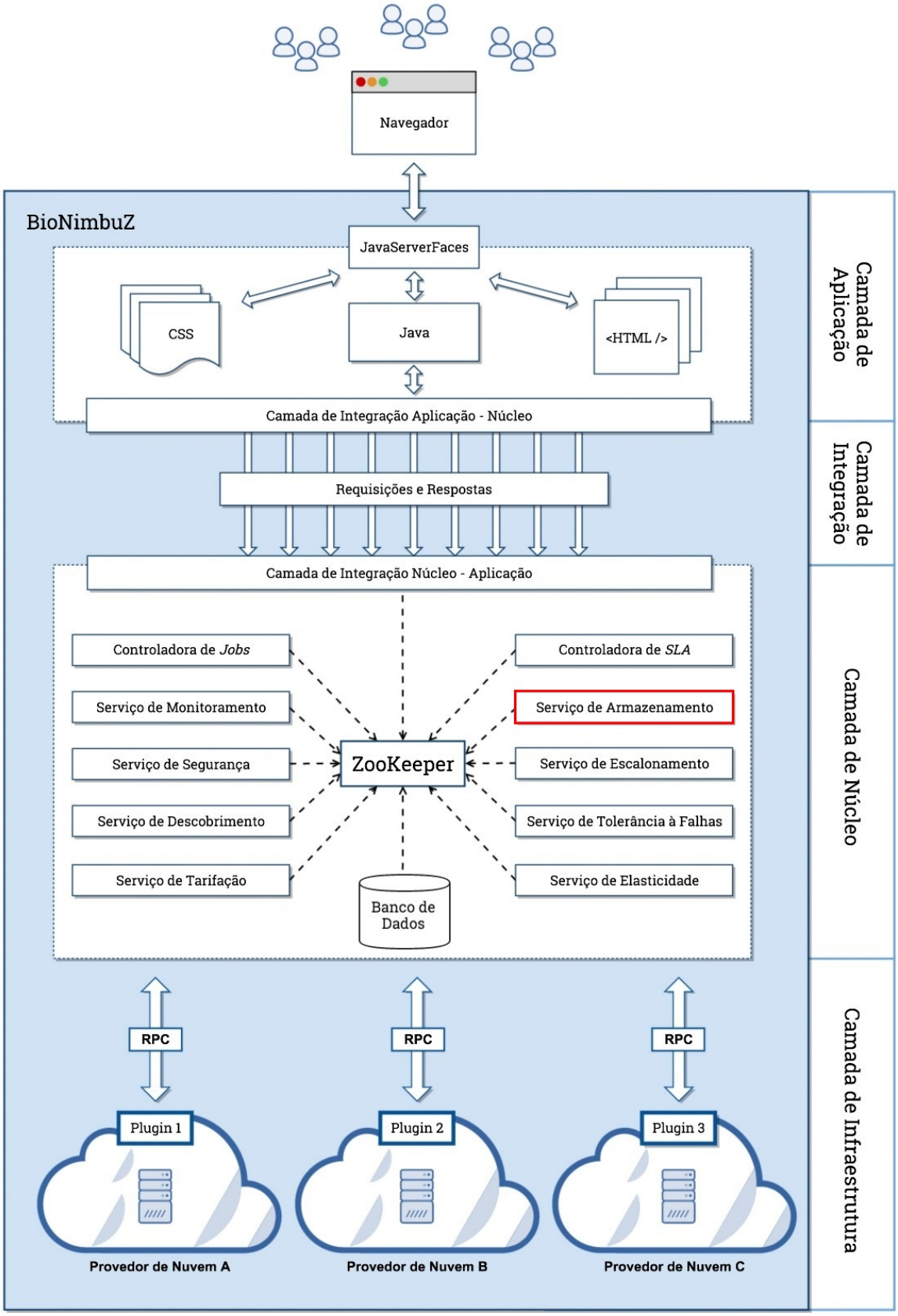

Figura 3.4: Arquitetura do BioNimbuZ [26].

3.2.1 Camada de Aplicação

A camada de aplicação é a responsável por toda a interação com o usuário. Atualmente, é composta por uma aplicação *web* que implementa um sistema gerenciador de *workflows* científicos. Essa interação inclui tarefas como *uploads* de arquivos e envio de *workflows* a serem executados. Além disso, também é responsabilidade da camada de aplicação exibir informações de *feedback* do sistema para o usuário, como listagem de arquivos armazenados e informações sobre uma determinada execução, como tempo, preço e horário de termino.

3.2.2 Camada de Integração

A camada de integração é responsável por integrar as camadas de aplicação e de núcleo. Ela realiza esta tarefa através da troca de mensagens entre essas camadas, utilizando serviços *web*, enviando requisições da camada de aplicação para o núcleo e recebendo respostas do núcleo para a camada de aplicação [26].

3.2.3 Camada de Núcleo

O núcleo do BioNimbuZ é o responsável por toda a gerência da federação, tendo como principais responsabilidades o descobrimento de novos provedores e recursos, o controle de acesso de usuários, o escalonamento de tarefas (chamadas de *jobs* neste contexto), o gerenciamento destes *jobs*, assim como todo o controle de armazenamento de arquivos presentes na federação. Estas responsabilidades estão distribuídas ao longo de diversos módulos internos, como pode ser visto na Figura 3.4, compostos principalmente de serviços, e também contendo alguns controladores. A seguir será descrito o funcionamento de cada um destes módulos, assim como suas responsabilidades:

- **Controlador de *Jobs*:** é o responsável por fazer a ligação entre o restante do núcleo da arquitetura e a camada de aplicação, recebendo as requisições vindas desta última e realizando o controle de acesso de usuário por meio do serviço de segurança, para permitir ou não a execução das tarefas. Além disso, o controlador de *jobs* também é responsável por gerenciar todos os pedidos de todos os usuários, mantendo um controle para que os resultados possam ser entregues e consultados, posteriormente, por cada usuário.
- **Serviço de Monitoramento:** é o responsável por verificar se um serviço requisitado pelo usuário está disponível na federação, juntamente ao Zookeeper, através do uso de *watchers*, tratando os alertas enviados pelos mesmos. Além disso, deve

acompanhar e garantir que todas as tarefas de um processo sejam executadas completamente, e ao seu término, deve informar ao serviço de escalonamento. Por fim, tem como responsabilidade permitir a recuperação de dados utilizados pelo restante dos módulos do servidor BioNimbuZ armazenados na estrutura hierárquica do Zookeeper, para que sejam possíveis reconstruções e garantias de execução de servidores solicitados.

- **Serviço de Tarifação:** é o responsável por realizar os cálculos relativos ao custo de utilização dos usuários ao realizar suas tarefas no BioNimbuZ. Esse cálculo é realizado a partir da troca de informações entre este serviço e o serviço de monitoramento, de maneira a manter atualizadas informações como quantidade de instâncias alocadas, quantidade de tarefas executadas e tempo de execução por tarefa, dentre outras. Além, é claro, de manter informações atualizada a respeito dos valores de utilização associados aos serviços contratados dos provedores de nuvem, através de seus *plugins*.

- **Serviço de Segurança:** é o serviço que deve garantir a segurança dos usuários e sua autenticação, garantindo que cada usuário acesse apenas suas informações através da interface. Além disso, este serviço deve garantir as permissões internas, de forma que cada usuário só seja capaz de realizar apenas ações sobre as quais ele tenha autorização.

- **Serviço de Descobrimento:** responsável por identificar os provedores de serviço e consolidar as informações acerca da capacidade de armazenamento, de processamento, de latência de rede e de recursos disponíveis na federação. O serviço de descobrimento envia mensagens para todos os provedores da federação, que respondem com informações referentes a infraestrutura e ferramentas disponíveis no mesmo. Para que seja possível consolidar esses dados, o serviço de descobrimento mantém uma estrutura de dados atualizada a cada nova resposta dos provedores, removendo desta as entradas referentes a provedores que não responderem as requisições.

  Por fim, tendo consolidada a informação acerca do estado atual da federação de nuvens, o serviço de descobrimento é capaz de atender as requisições enviadas pelo restante dos módulos da federação, repassando tais informações armazenadas.

- **Controlador de SLA:** para toda tarefa submetida para a federação por meio da camada de aplicação, o usuário deve preencher um *template* de SLA, que representa, de maneira geral, os parâmetros de qualidade de serviço, do inglês *Quality of Service* (QoS), que o usuário deseja durante sua execução na plataforma da federação. Tais parâmetros podem descrever requisitos funcionais, como número de núcleos de CPU desejados, frequência de *clock* de CPU, tamanho da memória, tamanho do

disco rígido, ou requisitos não-funcionais, como tempo de resposta, custo a pagar, taxa de disponibilidade ou tempo máximo de execução. Isso tudo ocorre para que haja um acordo de nível de serviço, do inglês *Service Level Agreement* (SLA), seja estabelecido previamente entre o usuário e a federação, que caso violado, resulta em multas através de crédito de serviço ou descontos na tarifação.

Assim sendo, o Controlador de SLA tem a responsabilidade de investigar se os requisitos especificados pelo usuário no *template* do SLA podem ser suportados pela federação naquele dado momento. Isso é realizado através dos dados de SLA obtidos pelo *plugin* de cada provedor presente.

Após a negociação bem sucedida do acordo, o controlador de SLA deve manter a sessão deste acordo através de um identificador único, que é repassado para o controlador de *jobs*, que acopla essa informação na sessão do usuário. A partir deste momento, é necessário que se mantenha um acompanhamento da execução deste usuário, verificando para que o acordo não seja violado em momento algum, lançando exceções caso contrário.

- **Serviço de Elasticidade:** o serviço de elasticidade é o responsável por ajustar dinamicamente a quantidade de recursos disponíveis na federação em um dado momento. Isso pode ser feito através simplesmente da adição ou remoção de determinadas instâncias de máquinas virtuais, chamada elasticidade horizontal, ou através da reconfiguração de parâmetros de determinadas instâncias de máquinas virtuais, alterando a capacidade de processamento, de armazenamento e de largura de banda disponíveis na mesma, chamada de elasticidade vertical.

- **Serviço de Escalonamento:** este módulo é o responsável por distribuir dinamicamente todas as tarefas recebidas pela camada de aplicação através do Controlador de *Jobs* entre as instâncias de máquinas virtuais disponíveis de todos os provedores de nuvem presentes na federação. Para realizar essa escolha, existe uma política de escalonamento interna ao serviço de escalonamento, que calcula, através de diversas métricas como tempo de execução e custo pela execução, atribuindo determinados pesos a cada uma dessas métricas, quais instâncias são as mais apropriadas para aquela determinada tarefa. Existem, na realidade, múltiplas políticas de armazenamento distintas, que operam de uma mesma maneira através de interfaces pré-definidas, tornando fácil e flexível que se realize alterações neste serviço.

  De uma maneira geral, o serviço de escalonamento envia para a política de escalonamento uma lista de tarefas a serem executadas e recebe como retorno um mapeamento de todas estas tarefas com suas respectivas instâncias alocadas para execução, de acordo com as métricas definidas pela política de escalonamento uti-

lizada. Além disso, o serviço de escalonamento deve manter um registro de quais tarefas estão executando em qual instância a todo momento, assim como o estado de execução das mesmas.

- **Serviço de Tolerância a falhas:** o serviço de tolerância a falhas tem como objetivo garantir que todos os demais serviços do BioNimbuZ estejam sempre disponíveis, e no caso da ocorrência de falhas, que tais serviços possam ser recuperados o mais rápido possível. Para realizar isso, é necessário que todos os objetos afetados por tal falha sejam restaurados a um estado prévio a esta. Este serviço ocorre no BioNimbuZ de uma maneira distribuída, estando presente cada um dos outros serviços.

  No serviço de monitoramento, a tolerância a falha ocorre ao receber um alerta informando que um determinado recurso está indisponível, o que dispara um processo de verificação nos demais módulos para checar se já foi feita a recuperação de todos os dados armazenados no servidor do Zookeeper. Neste cenário, também deve ser feita a verificação caso haja uma falha ao se realizar o escalonamento ou execução de um determinado *job*, ou na inclusão de algum arquivo que o serviço de armazenamento não tenha reconhecido.

  No serviço de armazenamento, a tolerância a falha está presente também no recebimento de alertas indicando indisponibilidade de algum recurso, que inicia um processo de replicação dos arquivos nos servidores do BioNimbuZ, visto que é necessário que haja uma quantidade específica de duplicadas de todo arquivo, a qualquer momento na federação.

  No serviço de escalonamento, a tolerância a falhas ocorre quando é recebido um alerta de indisponibilidade de recurso, o que inicia a recuperação de todas as tarefas que estão atualmente escalonadas para execução neste recurso.

  Tais recuperações ocorrem através do uso de *watchers* presentes nos *znodes* de *STATUS* que estão em todos os serviços do BioNimbuZ.

- **Serviço de Armazenamento:** é o serviço responsável pelas decisões relacionadas ao armazenamento de arquivos utilizados pelas aplicações a serem executadas e que devem ser mantidos na federação. Esse armazenamento deve acontecer de maneira eficiente, de forma a possibilitar que tais aplicações possam utilizar os arquivos de maneira transparente, ou com o menor custo possível. Esse custo é calculado através de uma política de armazenamento que utiliza métricas como a capacidade de armazenamento, a latência de rede, o *uptime* da máquina, a largura de banda e a distância entre a fonte e o destino da transferência. O BioNimbuZ já teve três políticas de armazenamento diferentes. Inicialmente era utilizada a política proposta

por Moura e Bacelar [21], em seguida foi utilizada a política proposta por Gallon [10] e atualmente se utiliza a política proposta pelo trabalho de Azevedo e Júnior [4].

Dada a importância para este trabalho, o serviço de armazenamento será melhor detalhado futuramente na Seção 3.4, na qual serão abordados também suas limitações e problemas, que este trabalho busca solucionar.

3.2.4 Camada de Infraestrutura

A camada de infraestrutura consiste nos recursos disponíveis na federação pelos seus respectivos provedores de nuvem, acessados e gerenciados através dos seus repectivos *plugins*. Para cada requisição proveniente do restante do núcleo do BioNimbuZ, o *plugin* deve mapeá-las para ações a serem realizadas na infraestrutura da federação. Como cada provedor de nuvem pode implementar a execução de ações em suas instâncias de maneiras distintas, é necessário que haja uma implementação diferente do *plugin* para cada provedor de serviço.

Existem básicamente três tipos de requisições que devem ser tratadas pela camada de infraestrutura, e estas são: informações sobre a infraestrutura do provedor; gerenciamento de tarefas; e transferência de arquivos.

3.3 Organização Lógica do BioNimbuZ

Nesta seção será descrita a maneira lógica como é estruturado o sistema do BioNimbuZ. Para garantir que seja prestado um serviço eficiente e robusto, mesmo estando distribuída dentre vários provedores de nuvem, a plataforma do BioNimbuZ foi implementada de forma hierárquica, onde são definidos três níveis lógicos:

- ***Master* Global:** o mestre global é o responsável por todas as tarefas de escalonamento, armazenamento, comunicação e decisão de todos os outros membros da federação. Isso ocorre de maneira distribuída, onde tais tarefas são coordenadas entre o mestre global e os mestres locais. Existe apenas um mestre global ativo na federação;
- ***Master* Local:** o mestre local é o responsável por receber tarefas provenientes do mestre global da federação e repassar aos seus respectivos escravos para que estas sejam executadas. Além disso, também é responsabilidade do mestre local realizar o escalonamento local, uma vez recebidas tais tarefas, e ao seu término, enviar as informações de *feedback* da execução ao mestre global. Existe apenas um mestre local para cada provedor de nuvem na federação;

- ***Slave*:** o escravo é aquela instância responsável por receber as tarefas de um dos mestres locais e executá-las. Toda instância de máquina virtual existente na federação executa o BioNimbuZ no nível hierárquico escravo, independente do provedor de nuvem relacionado.

Essas divisões hierárquicas, no entanto, são completamente transparentes para o usuário, pois a camada de aplicação, que contém toda a interface com o usuário, mantém contato apenas com o mestre global da federação, como pode ser observado na Figura 3.5.

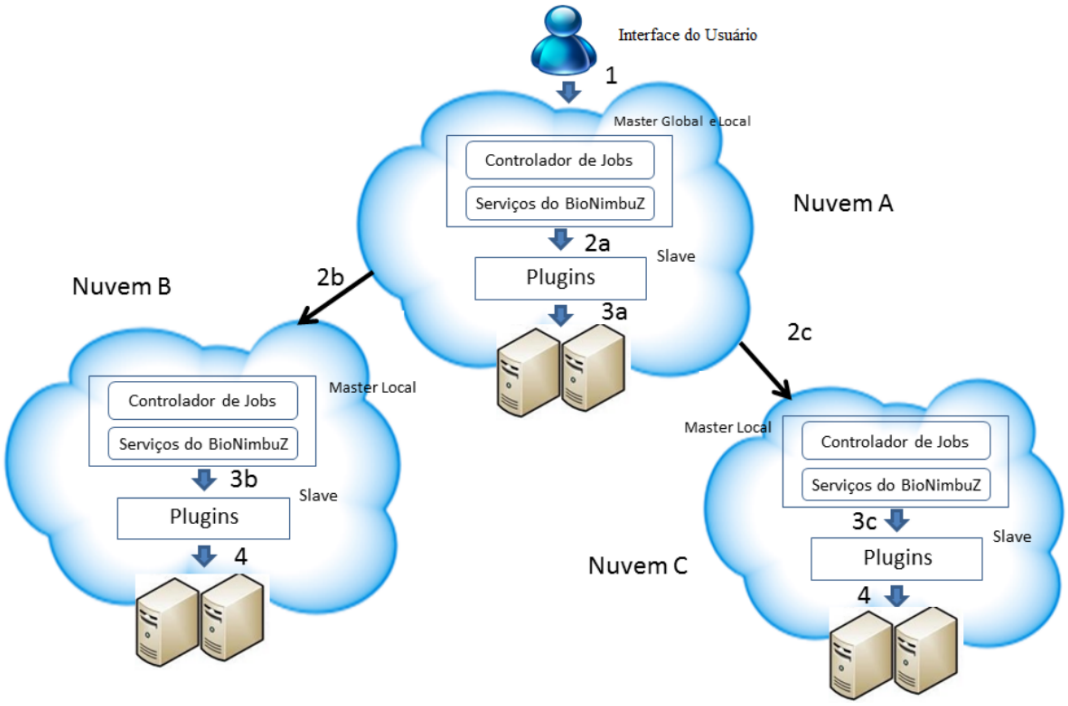

Figura 3.5: Organização Lógica do BioNimbuZ [7].

Primeiramente, ocorre o contato entre o módulo de interação com o usuário, presente na camada de aplicação no BioNimbuZ, com o controlador de *jobs* do mestre global da federação. O controlador de *jobs* por sua vez deverá ativar o serviço de segurança para verificar e autenticar as credenciais daquele usuário. Após esta etapa, o usuário será capaz de solicitar a execução de tarefas na federação. Estas tarefas serão escalonadas entre os recursos disponíveis na federação através de escolhas realizadas pelo serviço de escalonamento. Neste momento, dependendo da decisão tomada pelo serviço de escalonamento, o mestre global deverá encaminhar esta tarefa para alguma de suas instâncias escravas, que estão presentes na mesma nuvem, ou para uma instância mestre local presente em outra nuvem, que será responsável por repassar a tarefa em questão para a instância escrava escolhida.

3.4 O Serviço de Armazenamento

O serviço de armazenamento é o serviço responsável pelas decisões relacionadas ao armazenamento de arquivos utilizados pelas aplicações que compõem os *workflows* a serem executados e que devem ser mantidos na federação. Esse armazenamento deve acontecer de forma a possibilitar que tais arquivos possam ser acessador de maneira transparente, ou com o menor custo possível. Esse custo é calculado, atualmente, pelo serviço de armazenamento, através de sua política de armazenamento [4], que utiliza métricas como a capacidade de armazenamento, a latência de rede, o *uptime* da máquina, a largura de banda e a distância entre a fonte e o destino da transferência.

Para realizar sua função, o serviço de armazenamento comunica-se com o serviço de descobrimento, visando obter acesso às informações necessárias sobre os recursos disponíveis na federação. Com isso, o serviço é capaz de saber as condições atuais de armazenamento de cada um dos provedores presentes na federação de nuvens, num certo momento.

Atualmente, o BioNimbuZ realiza o armazenamento em sua federação através dos discos presentes em suas instâncias de máquinas virtuais da federação de nuvens, fazendo com que todas as transferências de arquivos entre instâncias na federação sejam realizadas através do protocolo SFTP. Além disso, adota-se no serviço de armazenamento uma política de redundância de arquivos, buscando garantir um maior nível de confiabilidade e impedir que se perca acesso a arquivos em caso de falhas em instâncias específicas. Por isso, em todo momento, este serviço assegura-se de que todos os arquivos que deseja-se manter armazenados na federação estejam replicados em, pelo menos, mais duas instâncias da mesma [21].

Além disso, o serviço de armazenamento é o responsável por manter no Zookeeper as informações referentes a todos os arquivos armazenados na federação, assim como em quais instâncias estes estão. Por fim, o serviço de armazenamento mantém uma tabela localmente em cada instância contendo informações de todos os arquivos presentes naquela mesma instância.

A proposta deste trabalho é de uma melhoria no serviço de armazenamento atual do BioNimbuZ, conforme proposto por Celesti [2] e Cachin [6]. Assim sendo, a ideia é alterar o cenário apresentado para que o serviço de armazenamento passe a utilizar serviços próprios de armazenamento oferecidos pelos mesmos provedores de nuvem. Dessa forma, o BioNimbuZ passará a ter um serviço de armazenamento mais robusto e eficiente, visto que os próprios serviços de armazenamento externos já oferecem um grau de disponibilidade alto o suficiente para os arquivos armazenados, eliminando a necessidade de que a replicação destes arquivos seja feita também pelo BioNimbuZ. Além disso, elimina-se a limitação de armazenamento que o BioNimbuZ tinha relacionada a capacidade dos discos

de suas instâncias de máquina virtual, que poderiam ficar saturados. Essa proposta será apresentada no próximo capítulo.

3.5 Considerações Finais

Neste capítulo foram apresentados detalhes da plataforma de federação de nuvens BioNimbuZ. Inicialmente, apresentou-se a sua arquitetura e as ferramentas utilizadas para sua implementação em um ambiente de nuvem distribuída. Em seguida, foram detalhados o objetivo e o funcionamento de seus serviços.

O Capítulo 4 irá apresentar com detalhes a proposta deste trabalho para o novo serviço de armazenamento implementado na plataforma de nuvens federadas do BioNimbuZ.

Capítulo 4

Novo Serviço de Armazenamento do BioNimbuZ

Neste capítulo serão apresentados, na Seção 4.1, os diferentes tipos de serviços de armazenamento disponíveis dentre os principais provedores de nuvens e a maneira como estes serviços funcionam, ressaltando suas principais características e limitações. Em seguida, na Seção 4.2, será descrito, de maneira ampla e detalhada, o funcionamento do serviço de armazenamento proposto neste trabalho para o BioNimbuZ.

4.1 Tipos de Serviços de Armazenamento Disponíveis

No cenário atual, os principais provedores da computação em nuvem são a Amazon, a Google e a Microsoft, por isso estes são os provedores abordados em trabalhos anteriores relacionados ao BioNimbuZ. Portanto, esses foram os serviços de armazenamento oferecidos por estes provedores analisados e levados em consideração na hora de compor o novo paradigma para o serviço de armazenamento do BioNimbuZ. Desta forma, com o estudo realizado, notou-se que existem, atualmente, dois tipos de serviços de armazenamento oferecidos pelos grandes provedores de nuvem, os quais são o Armazenamento por Blocos e o Armazenamento por Objetos. As Seções 4.1.1 e 4.1.2 irão detalhar estes tipos de serviço, respectivamente, detalhando a maneira como funcionam, suas limitações e suas vantagens.

4.1.1 Armazenamento por Blocos

Este serviço oferece um modelo de armazenamento no qual os arquivos são salvos em blocos de armazenamento de tamanho uniforme, de maneira semelhante ao que acontece

em um sistema de arquivos de um disco rígido comum. No Armazenamento por Blocos, apenas o conteúdo útil dos arquivos é salvo em seu respectivos endereço, sem que se guarde nenhuma informação adicional sobre estes arquivos, como seus metadados. Estes blocos são dispostos em volumes, que atuam como discos rígidos que podem ser anexados virtualmente a uma de suas instâncias de máquina virtual [16].

Entretanto, este modelo de serviço apresenta uma considerável limitação, pois cada volume só pode ser anexado a uma instância de máquina virtual de maneira simultânea. Assim, em um ambiente de escala elevada, onde existe um grande número de instâncias requisitando arquivos em um mesmo momento, continuará sendo necessário que se realize replicações em nível de volumes, pois não será possível que se tenha uma alta taxa de disponibilidade para um arquivo, caso este só esteja armazenado em um volume.

Além disso, o único provedor utilizado pelo BioNimbuZ que oferece este modelo de serviço é a Amazon, através de seu Amazon *Elastic Block Store* (EBS) [1]. Isto tem algumas outras limitações, como o fato de que os volumes de armazenamento oferecidos pelo Amazon EBS só podem ser anexados em instâncias de seu próprio serviço de máquinas virtuais, o Amazon *Elastic Compute Cloud* (EC2) [2]. Então, um volume de armazenamento do Amazon EBS não pode ser anexado a instâncias de máquinas virtuais dos serviços oferecidos pela Google ou pela Microsoft, por exemplo. Isso fere um dos principais objetivos de uma federação de nuvens, que se baseia em serviços de múltiplos provedores de nuvem sendo oferecidos ao seu usuário final de maneira transparente.

Além das desvantagens já citadas, o Amazon EBS só permite que seus volumes sejam anexados a instâncias de máquinas virtuais do Amazon EC2 que estejam executando na mesma região geográfica em que o volume foi criado, tornando ainda mais complexa e custosa a replicação necessária para que se obtenha uma alta taxa de disponibilidade, pois para tornar um arquivo disponível na federação do BioNimbuZ, seria necessário armazená-lo em múltiplos volumes em cada uma das regiões geográficas utilizadas pela federação.

4.1.2 Armazenamento por Objetos

Este tipo de armazenamento, que é encontrado nos serviços da Amazon S3 [3], do Google *Cloud Storage* [4] e do Azure *Blob Storage* [5], consiste em um armazenamento que, de forma diferente ao que ocorre no Armazenamento por Blocos, não particiona os arquivos para salvá-los, mas os salva de maneira integral em objetos. Cada objeto salvo contém, além do

[1] https://aws.amazon.com/ebs/
[2] http://aws.amazon.com/pt/ec2/
[3] http://aws.amazon.com/s3/
[4] https://cloud.google.com/storage/
[5] https://azure.microsoft.com/services/storage/blobs/

conteúdo útil do arquivo, informações extras como seus metadados e o seu identificador único.

Os serviços de Armazenamento por Objetos disponíveis pelos principais provedores de nuvem oferecem um serviço baseado em *buckets*, que agem como recipientes nos quais podem ser armazenados diversos objetos e que podem ser acessados de maneira distribuída de qualquer local, sob demanda. Estes *buckets* podem ser criados em localidades chamadas de multi-regionais, como por exemplo América do Sul, América do Norte, Europa e Ásia; ou regionais, como por exemplo Centro da América do Norte, Leste da América do Norte, Oeste da Europa, dentre outras [16].

A replicação necessária para garantir um grau de disponibilidade alto é fornecida pelo próprio serviço, e regida pelo seu próprio Acordo de Nível de Serviço (SLA). Esta replicação ocorre dentro da localidade determinada para o *bucket* na hora de sua criação. Então, um objeto armazenado em um *bucket* criado na região da Europa, será replicado em múltiplas sub-regiões daquela mesma macro-região, porém um objeto armazenado em um *bucket* criado na região do Leste da Europa será replicado apenas naquela mesma sub-região.

A maior vantagem que pode ser observada no Armazenamento por Objetos, quando comparado ao Armazenamento por Blocos, está no fato de que no Armazenamento por Objetos é possível que sejam feitos acessos simultâneos, sejam de escrita ou de leitura, a um mesmo *bucket*, o que permite que duas máquinas virtuais acessem, em um mesmo momento, arquivos que estejam em um mesmo *bucket*. Isso faz com que a utilização do Armazenamento por Objetos seja muito mais eficiente para o ambiente de nuvem federada do BioNimbuZ, no qual tipicamente são necessários muitos acessos simultâneos a um mesmo arquivo ou volume de armazenamento.

Considerando estas vantagens, o tipo de serviço escolhido para compor a base do módulo do novo serviço de armazenamento do BioNimbuZ foi o Armazenamento por Objetos, onde foram utilizados mais especificamente os serviços da Amazon S3 e do Google *Cloud Storage*. O Azure *Blob Storage* não foi considerado por que apresentou grandes limitações de autenticação de seus *buckets*, ao serem acessados por instâncias de máquinas virtuais utilizando um sistema operacional do tipo Unix, que é justamente o tipo padrão de sistema operacional utilizado pelas instâncias presentes na federação do BioNimbuZ.

Inicialmente, a fim de medir o grau de usabilidade e de eficiência dos serviços escolhidos, foram criados *buckets* nos dois serviços e em múltiplas localidades geográficas, que podem ser vistos na Tabela 5.2. É importante ressaltar que não foi criado um *bucket* na região do Brasil no serviço da Google, por que essa região não está incluída em seus serviços de nuvem.

Em seguida, foram feitas medições de banda para acessos a estes *buckets* através de

máquinas virtuais instanciadas em todos os três provedores de nuvem utilizados pelo BioNimbuZ: Amazon, Google e Microsoft. A fim de buscar as limitações para acesso nos piores casos, utilizou-se todas as combinações possíveis de acesso entre as regiões do Brasil, Estados Unidos e Europa. Para cada caso de teste foi realizado um conjunto de vinte medições, com o intuito de representar os resultados da melhor maneira possível, atenuando possíveis oscilações nos mesmos. Os resultados destas medições podem ser observados na Tabela 4.1, na qual constam, para cada caso de teste, não apenas os valores médios, mas também os valores de desvio padrão. A ideia é mensurar o nível de oscilação obtido em cada caso de teste.

Como pode ser visto ver na Tabela 4.1, o melhor resultado obtido para escritas ocorreu no caso de teste número 15, no qual foi utilizada uma máquina virtual da Amazon, instanciada na região da Europa (EU), para realizar escritas no *bucket* da Google, criado na região também da Europa. Este caso de teste apresentou um valor médio para escrita de 24.88 MB/s, com um desvio padrão de apenas 1.55 MB/s. Para leituras, o melhor resultado obtido pode ser visto no caso de teste 16, no qual utilizou-se uma máquina virtual da Amazon, instanciada na região dos Estados Unidos (US), para realizar leituras ao *bucket* também da Google e instanciado na mesma região. Neste caso de teste obteve-se uma taxa média de escrita de 25.43 MB/s, com um desvio padrão de 1.98 MB/s.

Por outro lado, o destaque negativo nas escritas pode ser observado no caso de teste 28, com uma máquina virtual da Azure, instanciada na região do Brasil (BR), realizando escritas em um *bucket* da Amazon criado na região da Europa, no qual o valor médio de escrita foi de 2.44 MB/s, com um desvio padrão de 0.24 MB/s. Por fim, o pior resultado obtido para leituras ocorreu no caso de teste 23, no qual utilizou-se uma máquina virtual da Google, instanciada na região da Europa, para realizar acessos ao *bucket* da Amazon criado na região do Brasil. O valor médio de leitura para este caso de teste foi de 2.43 MB/s, com um desvio padrão de 0.24 MB/s.

Assim sendo, é possível observar que um dos fatores que mais influenciou nos resultados obtidos foi a distância geográfica entre as regiões em que o *bucket* foi criado e que máquina virtual foi instanciada. Além disso, pode-se observar que os casos de teste que utilizaram *buckets* da Google apresentaram valores consideravelmente maiores do que os apresentados nos casos de teste cujos *buckets* utilizados foram da Amazon.

As velocidades de transferência obtidas nos melhores casos de teste, em que o acesso ocorreu dentro de uma mesma região, foram extremamente satisfatórios, com valores médios atingindo até 24.88 MB/s na escrita e 25.43 MB/s na leitura, velocidades estas que podem ser consideradas boas até mesmo em um cenário de transferência local.

Apesar das baixas velocidades obtidas nos piores casos de teste, que chegaram a apenas 10% daquelas obtidas nos melhores casos, esta é uma limitação que sempre estará

Tabela 4.1: Tabela de Medição de Banda para os Casos de Teste (valores em MB/s).

Teste	**Provedor da VM**	**Região da VM**	**Provedor do *Bucket***	**Região do *Bucket***	**Escrita (Média)**	**Escrita (Desvio Padrão)**	**Leitura (Média)**	**Leitura (Desvio Padrão)**
1	Amazon	BR	Amazon	BR	12.99	0.95	9.37	0.27
2	Amazon	BR	Amazon	US	7.93	1.35	5	0.44
3	Amazon	BR	Amazon	EU	7.29	0.14	4.18	0.16
4	Amazon	BR	Google	US	8.96	0.38	8.81	0.52
5	Amazon	BR	Google	EU	8.47	0.37	8.35	0.26
6	Amazon	US	Amazon	BR	6.17	0.39	4.58	1.85
7	Amazon	US	Amazon	US	9.05	0.75	8.77	0.52
8	Amazon	US	Amazon	EU	7.55	0.69	5.77	1.04
9	Amazon	US	Google	US	11.46	1.54	8.72	0.44
10	Amazon	US	Google	EU	14.26	2.19	12.53	1.26
11	Amazon	EU	Amazon	BR	6.63	0.38	4.01	1.98
12	Amazon	EU	Amazon	US	10.06	0.8	4.57	1.08
13	Amazon	EU	Amazon	EU	15.88	0.59	16.35	1
14	Amazon	EU	Google	US	24.86	4.43	15.59	0.78
15	**Amazon**	**EU**	**Google**	**EU**	**24.88**	**1.53**	**16.65**	**0.7**
16	**Google**	**US**	**Google**	**US**	**21.26**	**4.59**	**25.43**	**1.98**
17	Google	US	Google	EU	17.77	5.63	18.5	1
18	Google	US	Amazon	BR	7.84	1.14	5.46	0.28
19	Google	US	Amazon	US	17.61	2.01	14.86	1.64
20	Google	US	Amazon	EU	10.94	0.57	9.7	0.16
21	Google	EU	Google	US	12.2	3.13	18.31	2.68
22	Google	EU	Google	EU	12.34	3.49	19.03	3.25
23	**Google**	**EU**	**Amazon**	**BR**	**7.46**	**0.23**	**2.43**	**0.24**
24	Google	EU	Amazon	US	10.28	1.86	7.13	1.1
25	Google	EU	Amazon	EU	12.55	1.92	16.33	1.11
26	Azure	BR	Amazon	BR	6.6	0.64	21.13	2.98
27	Azure	BR	Amazon	US	4.65	0.29	9.45	2.57
28	**Azure**	**BR**	**Amazon**	**EU**	**2.44**	**0.24**	**9.08**	**0.11**
29	Azure	BR	Google	US	3.79	0.32	8.22	1.54
30	Azure	BR	Google	EU	5.29	1.28	8.76	1.09
31	Azure	US	Amazon	BR	6.76	0.21	11.1	1.18
32	Azure	US	Amazon	US	5.56	0.43	21.17	3.14
33	Azure	US	Amazon	EU	7.23	0.33	15.15	1.44
34	Azure	US	Google	US	15.06	8.41	12.93	0.79
35	Azure	US	Google	EU	9.82	1.93	16.18	1.21
36	Azure	EU	Amazon	BR	3.37	0.52	6.03	2.35
37	Azure	EU	Amazon	US	7.73	4.59	10.3	3.29
38	Azure	EU	Amazon	EU	16.5	4.82	21.46	3.35
39	Azure	EU	Google	US	7.13	0.76	15.86	0.8
40	Azure	EU	Google	EU	8.54	1.93	17.28	1.28

presente no contexto atual para cenários em que ocorrem transferências de dados através da Internet entre pontos muito distantes geograficamente. Além disso, estes valores ainda podem ser considerados aceitáveis, visto que arquivos 8 GB, por exemplo, levariam apenas cerca de 56 minutos para serem transmitidos numa velocidade de 2.43 MB/s.

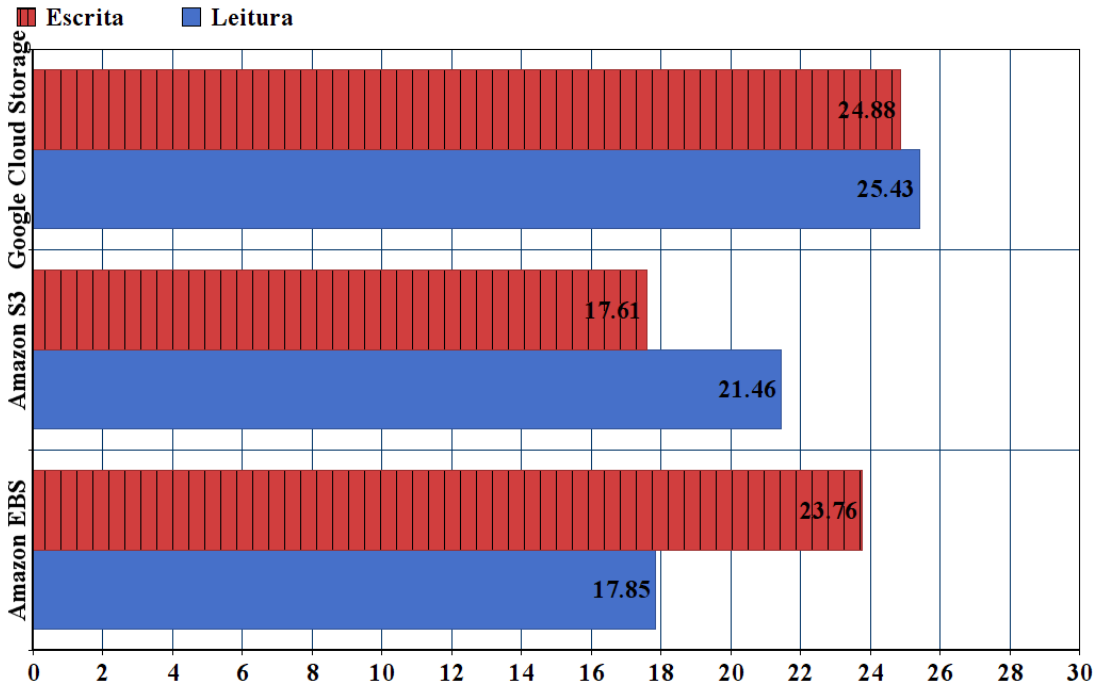

Figura 4.1: Comparativo dos Serviços de Armazenamento por Bloco e por Objeto (Valores em MB/s).

Para critérios de comparação, realizou-se o mesmo teste de transferência com escritas e leituras para o Amazon EBS, que é um serviço de Armazenamento por Blocos. A Figura 4.1 mostra um gráfico comparativo entre os melhores valores obtidos em cada um dos serviços de Armazenamento por Objetos e no Amazon EBS. Com o gráfico apresentado na Figura 4.1 é possível ver que as taxas de transferência obtidas nos três serviços são muito próximas, com uma pequena vantagem para o serviço de Armazenamento por Objetos da Google. Com isso, pode-se chegar a conclusão de que, além das vantagens já citadas, os serviços de Armazenamento por Objetos analisados apresentam, em média, taxas de leitura e de escrita iguais ou superiores àquelas obtidas no serviço de Armazenamento por Blocos analisado, embasando ainda mais a escolha dos serviços a serem utilizados para compor a base do módulo do novo serviço de armazenamento do BioNimbuZ

4.2 Funcionamento do Serviço de Armazenamento Proposto

Conforme visto na Seção 3.4, o funcionamento do serviço de armazenamento atual do BioNimbuZ apresenta algumas limitações, provenientes da utilização dos discos das instâncias de máquinas virtuais da federação para a realização do armazenamento na mesma. Isto faz com que seja necessário realizar a replicação de arquivos, a fim de se obter um grau de disponibilidade aceitável para os arquivos, dificultando cálculos necessários para os SLAs a serem oferecidos para os usuários. Além disso, gera o acoplamento de nós de processamento e de armazenamento em um mesmo ambiente de nuvens, gerando desperdícios de poder computacional.

Celesti [2] e Cachin [6] propõem que, ao combinar serviços de armazenamento já consolidados e de qualidade oferecidos por provedores de nuvens, são obtidos resultados evidentemente superiores, se comparados à tentativa de implementar as funcionalidades destes serviços de maneira interna ao seu serviço, como o que ocorre no cenário atual do BioNimbuZ.

Neste contexto, e tendo em mente a maneira como estes serviços de armazenamento disponíveis funcionam, o novo serviço de armazenamento irá combinar os serviços oferecidos pelo Amazon S3 e pelo Google *Cloud Storage* para implementar um serviço de armazenameto que não mais necessite de instâncias de máquina virtual para realizar o armazenamento de arquivos na federação.

Desta forma, com o novo serviço de armazenamento implementado neste trabalho, serão desacopladas as responsabilidades de armazenamento e de processamento em nós distintos na federação. Além disso, será eliminada a necessidade de tarefas de replicação de arquivos para que se obtenha um grau de disponibilidade aceitável, visto que esta funcionalidade já é oferecida pelos próprios serviços de armazenamento dos provedores.

Assim sendo, o serviço de armazenamento proposto continua funcionando como um módulo interno ao núcleo do BioNimbuZ, e tem como responsabilidade a gestão de todas as transferências de arquivos realizadas entre as instâncias de máquinas virtuais da federação e os *buckets* da federação. Para realizar este acesso aos arquivos armazenados nos *buckets* da federação, o serviço de armazenamento utiliza, além de métodos oferecidos pelas APIs (do inglês *Application Programming Interface*) específicas fornecidas pelos provedores dos serviços, adaptadores de código aberto para Linux FUSE (*Filesystem in Userspace*). Estes adaptadores permitem que se realize a montagem dos *buckets* no sistema de arquivos Unix, através de diretórios virtuais, permitindo que se acesse os objetos contidos nestes *buckets* de maneira trivial, mesmo que os acessos ainda estejam sendo feitos por meio da Internet.

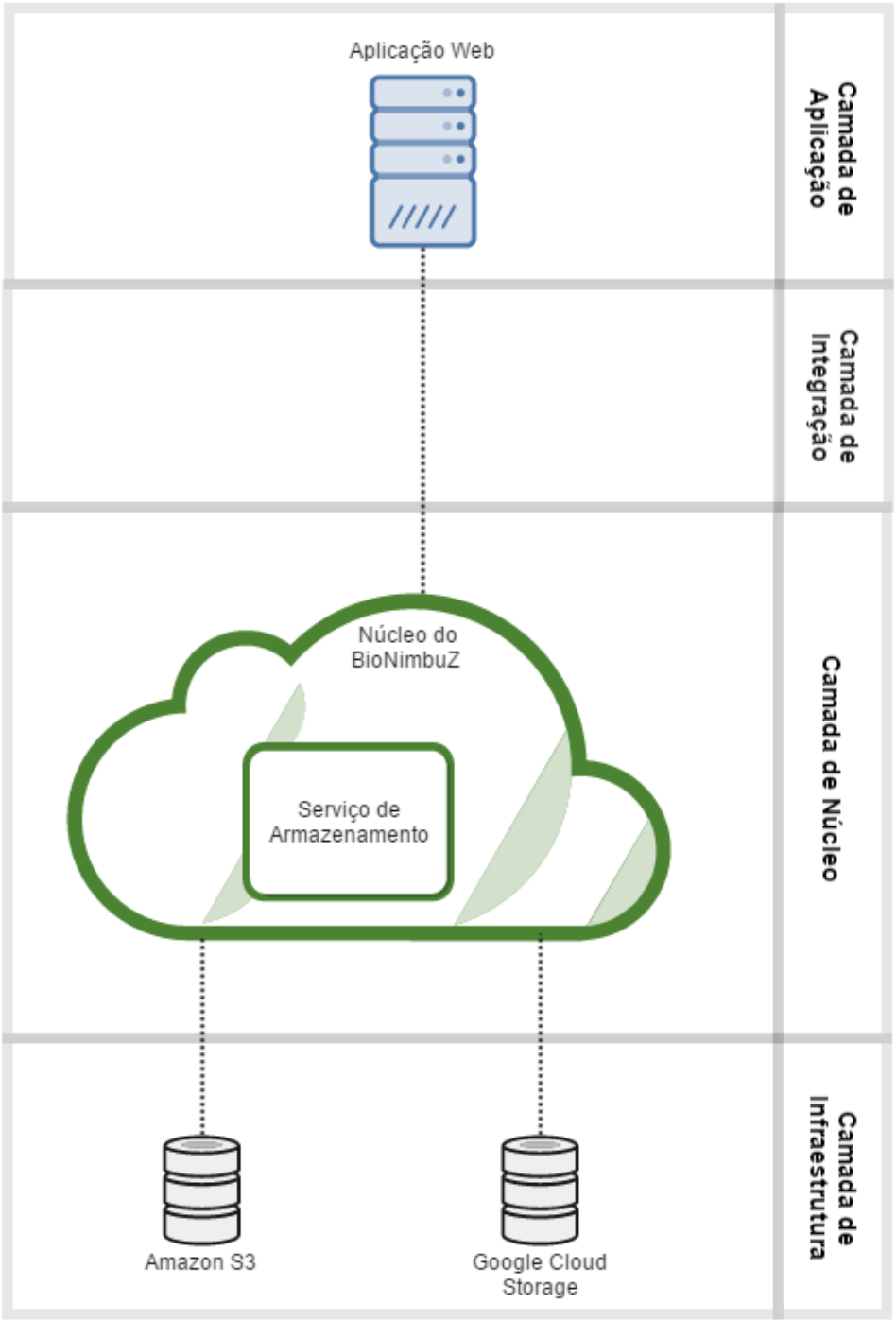

Figura 4.2: Diagrama Simplificado de Comunicação do Serviço de Armazenamento Proposto.

A Figura 4.2 mostra um diagrama simplificado da arquitetura do BioNimbuZ, conforme visto anteriormente na Figura 3.4, mostrando a comunicação do novo serviço de armazenamento e os serviços de armazenamento externos utilizados na federação.

No cenário atual do BioNimbuZ, um usuário que deseja realizar a execução de um *workflow* deve realizar o *login* através da aplicação web, que irá autenticar suas credenciais com o núcleo do BioNimbuZ. Em seguida, o usuário deve efetuar o *upload* dos arquivos de entrada necessários para a execução de seu *workflow*. Este arquivo será enviado para algum dos *buckets* da federação, passando primeiramente pelo núcleo do BioNimbuZ. Por fim, o usuário deve realizar o *design* do *workflow* através da interface existente na aplicação web. As tarefas referentes a este *workflow* são escalonadas entre os nós de processamento da federação do BioNimbuZ, que realizam a requisição dos arquivos de entrada, quando necessários, para em seguido realizar o *download* destes, diretamente dos *buckets* que os contém.

A Seção 4.2.1 a seguir irá descrever com um maior detalhamento o funcionamento e as responsabilidades desse novo serviço de armazenamento implementado no BioNimbuZ.

4.2.1 Funcionamento no Novo Serviço de Armazenamento

O núcleo do BioNimbuZ, conforme descrito na Seção 3.2.3, é dividido em vários módulos, cada um sendo responsável por determinados serviços na federação. Assim sendo, o serviço de armazenamento proposto por este trabalho foi implementado como um novo módulo no núcleo do BioNimbuZ, cujas responsabilidades são:

- Manter uma lista com todos os *buckets* que serão utilizados pelo BioNimbuZ, contendo suas informações, como nome, estado (se está montado ou não), caminho no qual está montado, além de seus valores de latência e banda;
- Autenticar credenciais do BioNimbuZ para acesso aos *buckets* que serão utilizados com seus respectivos provedores;
- Montar todos os *buckets* que serão utilizados no sistema de arquivos de cada instância de máquina virtual executando o BioNimbuZ;
- Manter atualizada a lista de arquivos presentes em cada um dos *buckets* da federação no Zookeeper;
- Realizar o *upload* dos novos arquivos gerados durante ou ao fim de execuções de *workflows* para o *bucket* da federação que possua o maior valor de banda medido por aquela instância contendo o arquivo;
- Atender a requisições de arquivos disparadas pelo serviço de escalonamento, caso estes arquivos estejam em algum dos *buckets* da federação.

Para atender as requisições solicitadas pelo serviço de escalonamento, foram implementados dois comportamentos para o serviço de armazenamento, como pode ser visto nas Figuras 4.3 e 4.4. Estes comportamentos são descritos a seguir:

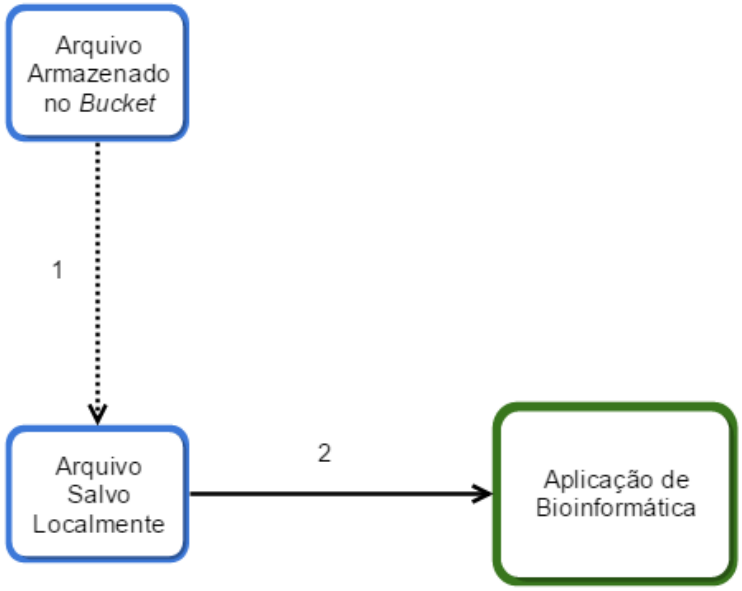

Figura 4.3: Diagrama do Comportamento 1 para Execução de Tarefas com o Novo Serviço de Armazenamento.

- **Comportamento 1:** Neste comportamento, apresentado na Figura 4.3, durante a execução de um *workflow*, quando realizada uma requisição de arquivo por algum dos nós de processamento da federação, é realizado o *download* prévio do arquivo (1), para somente então se iniciar a execução da etapa do *workflow* em questão (2).

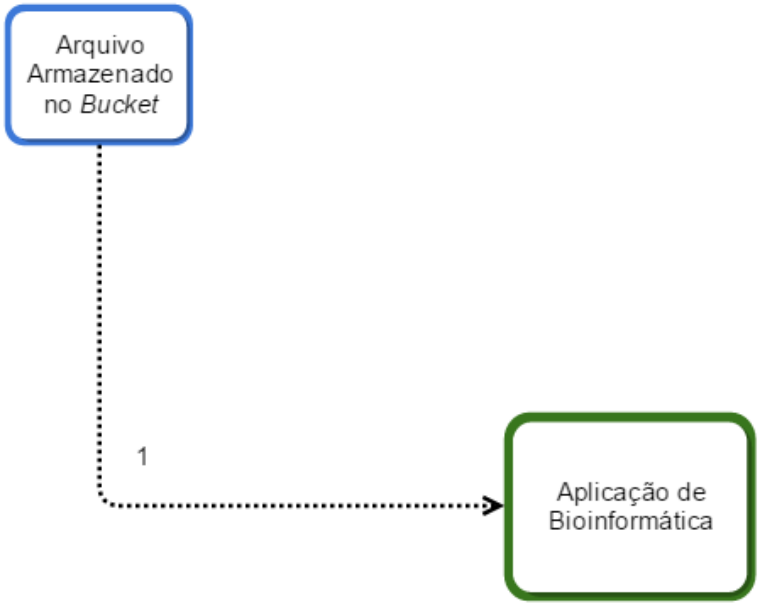

Figura 4.4: Diagrama do Comportamento 2 para Execução de Tarefas com o Novo Serviço de Armazenamento.

- **Comportamento 2:** Neste comportamento, visto na Figura 4.4, durante uma execução de *workflow*, ao ser realizada uma requisição de arquivo por algum dos nós de processamento da federação, é passado como parâmetro de entrada da etapa do *workflow* a referência daquele arquivo armazenado no *bucket*, através de seu caminho dentro do diretório virtual montado localmente no sistema de arquivos da instância. Com a utilização desta referência, o *download* é realizado simultaneamente à execução da tarefa (1).

Com essa implementação do serviço proposto, é possível atingir na federação um grau de disponibilidade para os arquivos da federação, oferecendo um serviço mais robusto e confiável ao seu usuário final, garantindo também uma federação das nuvens para os serviços de armazenamento.

4.3 Considerações Finais

Este capítulo abordou as etapas de desenvolvimento do novo serviço de armazenamento proposto para o ambiente de nuvens federadas do BioNimbuZ. Para isto, foi inicialmente detalhado o funcionamento e as limitações dos tipos de serviço de armazenamento disponíveis atualmente pelos grandes provedores de serviço em nuvem, e explicitados os motivos

para a decisão do tipo de serviço escolhido. Em seguida, foi apresentado o funcionamento deste serviço de armazenamento proposto e detalhadas suas principais vantagens.

No próximo capítulo serão apresentados os testes realizados para a verificação do funcionamento do novo serviço de armazenamento.

Capítulo 5

Resultados Obtidos

Neste capítulo serão apresentados os testes realizados para verificar o funcionamento do serviço de armazenamento proposto neste trabalho, assim como testes de desempenho para comparação com o serviço de armazenamento anterior. Para isso, a Seção 5.1 descreve os objetivos esperados na realização destes testes. A Seção 5.2 mostra o ambiente no qual foram realizados os testes. A Seção 5.3 apresenta os procedimentos e os resultados obtidos nos testes funcionais, para testar o funcionamento do novo sistema implementado. Por fim, na Seção 5.4, estão os procedimentos e os resultados obtidos nos testes de desempenho realizados.

5.1 Objetivos dos Testes

O objetivo dos testes funcionais, apresentados na Seção 5.3, é verificar o funcionamento do novo serviço de armazenamento em um cenário real de nuvem federada, com a integração completa entre as alterações realizadas no núcleo do BioNimbuZ. Para isso, foram realizados testes para as principais funcionalidades do novo serviço, através da execução completa de um *workflow* real de Bioinformática, utilizando os dois comportamentos de execução de tarefas, vistos nas Figuras 4.3 e 4.4.

Em seguida, na Seção 5.4, foram realizados testes de desempenho, cujo objetivo é verificar a *performance* do novo serviço para averiguar o grau de usabilidade do mesmo se comparado com o serviço de armazenamento antigo. Com isso, será possível analisar se houve algum ganho de *performance* com a nova proposta, além dos ganhos já citados de praticidade e acessibilidade de arquivos, visto que esses estarão armazenados em múltiplos provedores de nuvem e poderão ser acessados de maneira distribuída e sob demanda, de qualquer instância de máquina virtual da federação.

5.2 Ambiente de Testes e *Workflow* Utilizado

Nesta seção serão apresentados os detalhes referentes a infraestrutura utilizada para a execução dos testes presentes neste capítulo, assim como detalhes acerca do *workflow* utilizado nos mesmos.

5.2.1 Configuração do Ambiente

Para a realização de todos os testes apresentados neste capítulo foram utilizadas instâncias de máquinas virtuais criadas na federação do BioNimbuZ, todas do mesmo provedor, o Google *Cloud Platform*, e contendo as mesmas configurações. Estas instâncias de máquinas virtuais representam os nós de processamento do BioNimbuZ, e cada uma executa o mesmo código do núcleo do BioNimbuZ. Além disso, em todos os testes realizados, foi utilizada uma mesma máquina local para executar a aplicação web do BioNimbuZ. A Tabela 5.1 apresenta as configurações presentes nesta máquina e nas instâncias da federação. Por fim, foram utilizados os *buckets* da federação, que podem ser vistos na Tabela 5.2.

Tabela 5.1: Configuração dos Computadores Utilizados nos Testes.

Infraestrutura	**Item de Configuração**	**Configuração**
Aplicação Web	Processador	Intel Core i5 3.1 GHz
	Memória	8 GB de Memória RAM
	Armazenamento	256 GB SSD
	Sistema Operacional	Linux Ubuntu 14.04
Instâncias do BioNimbuZ	Processador	4 vCPU
	Memória	15 GB de Memória RAM
	Armazenamento	10 GB HD
	Sistema Operacional	Linux Ubuntu 14.04
	Provedor	Google *Cloud Platform*

Tabela 5.2: *Buckets* Criados na Federação do BioNimbuZ.

Nome	**Provedor**	**Região**
bionimbuz-a-br	Amazon	Brasil (BR)
bionimbuz-a-us	Amazon	Estados Unidos (US)
bionimbuz-a-eu	Amazon	Europa (EU)
bionimbuz-g-us	Google	Estados Unidos (US)
bionimbuz-g-eu	Google	Europa (EU)

5.2.2 *Workflow* Utilizado

Com o objetivo de testar a integração e medir a *performance* do serviço de armazenamento proposto, foram executados testes utilizando uma parte de um *workflow* real de Bioinformática, cujo objetivo é a identificação de genes diferencialmente expressos em células humanas cancerosas do rim e do fígado [13, 27], com fragmentos gerados pelo sequenciador Illumina [1]. O diagrama das duas etapas do *workflow* utilizado pode ser visto na Figura 5.1.

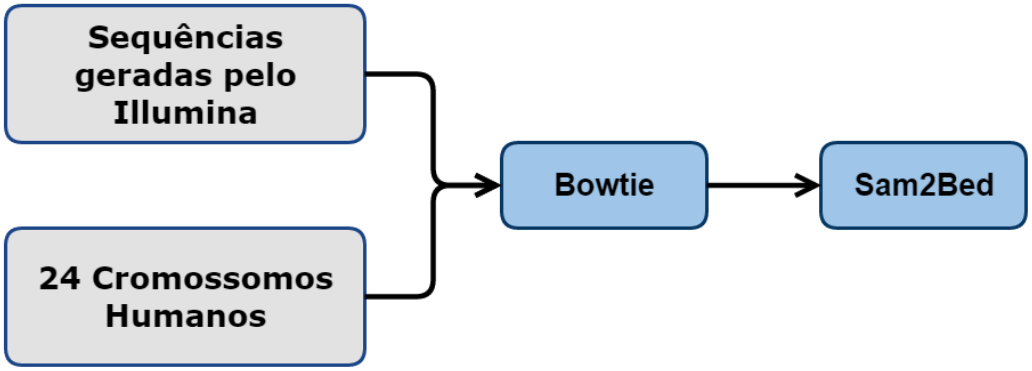

Figura 5.1: Etapas do *Workflow* Utilizado nos Testes.

Na primeira etapa do *workflow* é utilizada a ferramenta Bowtie [2], a qual mapeia os fragmentos nos 24 cromossomos humanos de referência, obtidos no banco de dados NCBI (*National Center for Biotechnology Information*) [3]. O Bowtie identifica a região do genoma de referência onde cada fragmento de entrada está localizado, dessa forma, ao mapear um conjunto de fragmentos em uma mesma região, é possível se inferir que esses vieram de uma mesma região no genoma de referência.

Na segunda etapa é utilizado um *script* de conversão, chamado Sam2Bed [4], que converte o arquivo de saída gerado pelo Bowtie, cuja extenção é .SAM para o formato .BED, que poderá ser utilizado por outras aplicações de Bioinformática.

5.3 Testes Funcionais

O objetivo dos testes apresentados nesta seção é testar as funcionalidades implementadas no serviço de armazenamento proposto por este trabalho. Estes testes foram realizados

[1] http://www.illumina.com/
[2] http://bowtie-bio.sourceforge.net/index.shtml
[3] http://www.ncbi.nlm.nih.gov/
[4] http://bedops.readthedocs.org/en/latest/content/reference/file-management/conversion/sam2bed.html

por meio de execuções do *workflow* descrito na seção anterior, utilizando os dois comportamentos para execuções oferecidos pelo novo serviço de armazenamento. Estas execuções englobam todas as principais funcionalidades implementadas, conforme pode ser visto a seguir:

- **Comportamento 1:** Execução de um *workflow* realizando o *download* prévio do arquivo de entrada, testando as seguintes funcionalidades:
  - *Upload* do arquivo de entrada feito através da aplicação web, e enviado posteriormente através do núcleo para um dos *buckets* da federação;
  - *Download* prévio do arquivo na instância que irá realizar a execução;
  - *Upload* automático do arquivo de saída do *workflow*;
  - *Download* do arquivo de saída através da aplicação web.
- **Comportamento 2:** Execução de um *workflow* utilizando a referência do arquivo armazenado no *bucket* como entrada, testando as seguintes funcionalidades:
  - *Upload* do arquivo de entrada feito através da aplicação web, e enviado posteriormente através do núcleo para um dos *buckets* da federação;
  - Execução e *download* da tarefa de maneira simultânea, através da referência do arquivo armazenado no *bucket* presente no sistema de arquivos da máquina;
  - *Upload* automático do arquivo de saída do *workflow*;
  - *Download* do arquivo de saída através da aplicação web.

Além disso, foi necessário que se realizassem testes acerca da implementação do novo serviço de armazenamento utilizando cada um dos *buckets* da federação, apresentados na Tabela 5.2, a fim de se comprovar o funcionamento do serviço de armazenamento proposto utilizando os serviços dos dois provedores, o Amazon S3 e o Google *Cloud Storage*, e em todas as regiões geográficas utilizadas. Portanto, cada teste de execução foi realizado utilizando um dos *buckets* e um dos dois comportamentos, totalizando dez execuções.

Todas as execuções foram realizadas na mesma instância da federação, criada na região dos Estados Unidos, cuja configuração pode ser vista na Tabela 5.1, com a aplicação web sendo executada sempre em uma máquina local, também especificada na Tabela 5.1. Os resultados esperados das execuções utilizando o **Comportamento 1** eram:

- Após a realização do *upload*, feito através da aplicação web, o arquivo deveria estar armazenado no *bucket* utilizado, e persistido no banco de dados do núcleo;
- Ao receber a tarefa, o serviço de armazenamento deveria realizar o *download* completo do arquivo de entrada, para somente então ser iniciada a execução desta tarefa;

- Ao final da execução do *workflow*, o serviço de armazenamento deveria identificar a presença do arquivo de saída, e realizar o seu *upload* para algum dos *buckets* da federação;
- Em seguida, deveria ser possível realizar o *download* deste arquivo de saída pelo *browser* do usuário, através da aplicação web.

De maneira semelhante, os resultados esperados das execuções utilizando o **Comportamento 2** eram:

- Após a realização do *upload*, feito através da aplicação web, o arquivo deveria estar armazenado no *bucket* utilizado, e persistido no banco de dados do núcleo;
- Ao receber a tarefa, a execução deveria ser iniciada imediatamente, utilizando como parâmetro de entrada a referência armazenada no *bucket* do arquivo de entrada do *workflow*;
- Ao final da execução do *workflow*, o serviço de armazenamento deveria identificar a presença do arquivo de saída e realizar o seu *upload* para algum dos *buckets* da federação;
- Em seguida, deveria ser possível realizar o *download* deste arquivo de saída pelo *browser* do usuário, por meio da aplicação web.

Todos os testes de execução apresentaram os resultados esperados. Isto comprovou que todas as funcionalidades do serviço de armazenamento proposto funcionam de maneira correta, assim como a interação entre este serviço no núcleo do BioNimbuZ e na aplicação web.

5.4 Teste de Desempenho

Após a confirmação de que o novo serviço de armazenamento do BioNimbuZ funcionava de maneira correta, obtida através dos testes funcionais apresentados na seção anterior, os testes de desempenho realizados nesta seção têm como objetivo a verificação da *performance* apresentada pelo novo serviço. Estes testes consistem em execuções do mesmo *workflow* utilizado nos testes funcionais, porém com o foco no tempo total de execução deste *workflow* em diferentes cenários.

Dessa forma, os primeiros testes de desempenho tinham o intuito de comparar a *performance* dos dois comportamentos para execução de tarefas oferecidos pelo serviço de armazenamento. Para isso, foi utilizada, novamente, uma instância de máquina virtual da federação, criada na região dos Estados Unidos, para execução dos *workflows*, com a

aplicação web sendo executada em uma máquina local. As especificações de ambas as máquinas podem ser vistas na Tabela 5.1.

Em seguida, foram escolhidos os *buckets* com base na sua localização geográfica e usando como base os valores de leitura expostos na Tabela 4.1. O uso dos valores de leitura foi escolhido como base pois é este tipo de operação que acontece quando um nó de processamento do BioNimbuZ realiza um *download* de um arquivo, independente de qual dos comportamentos esteja sendo utilizado, para executar uma tarefa. Portanto, o objetivo destes testes é observar a diferença de desempenho entre o **Comportamento 1** e o **Comportamento 2** para diferentes velocidades de acesso na transferência desse arquivo, conforme visto nos cenários a seguir:

1. **Cenário 1:** acesso a um *bucket* da Google na região dos Estados Unidos (US), mesma região na qual foi criada a instância que executa a tarefa, oferecendo assim um cenário com uma alta velocidade de leitura, (Teste número 16 na Tabela 4.1);
2. **Cenário 2:** acesso a um *bucket* da Amazon na região da Europa (EU), gerando um cenário com uma velocidade de leitura média (Teste número 20 na Tabela 4.1), menor do que a velocidade apresentada no **Cenário 1**, mas maior do que a velocidade apresentada no **Cenário 2**;
3. **Cenário 3:** acesso a um *bucket* Amazon na região do Brasil (BR), oferecendo um cenário com uma velocidade de leitura baixa (Teste número 18 na Tabela 4.1).

Assim sendo, foram realizados seis testes de execução do *workflow* escolhido, sendo uma execução para cada comportamento e em cada um dos cenários apresentados. Os tempos de execução destes testes podem ser vistos na Figura 5.2, na qual observa-se que houveram pequenos ganhos de *performance* em todos os cenários utilizando o **Comportamento 2**, com uma redução de até 5.7% do tempo total de execução nos cenários com uma pior banda de leitura.

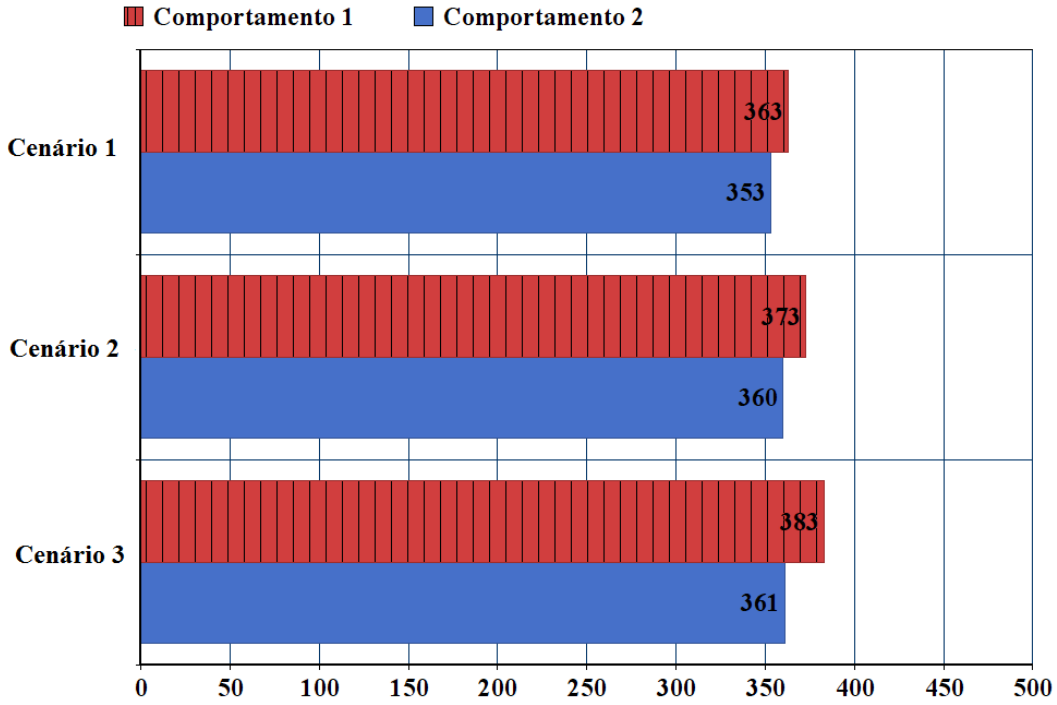

Figura 5.2: Tempos de Execução para os Dois Comportamentos do Novo Serviço de Armazenamento (tempos em segundos).

Em seguida, realizou-se testes com a execução do mesmo *workflow* utilizando o serviço de armazenamento antigo do BioNimbuZ, com o intuito de comparar os tempos de execução obtidos nestas execuções, aos tempos obtidos nas execuções utilizando o novo serviço de armazenamento.

Para os testes realizados com o serviço de armazenamento antigo, foram utilizadas três instâncias de máquinas virtuais da federação, todas com a mesma especificação, as quais foram apresentadas na Tabela 5.1. Foram realizadas duas execuções para estes testes: na primeira, utilizou-se uma máquina virtual criada na região dos Estados Unidos (US), requisitando um arquivo presente em uma outra máquina virtual, criada também na região dos Estados Unidos (US), replicando o **Cenário 1** dos testes realizados anteriormente. Na segunda execução utilizou-se novamente uma máquina virtual criada na região dos Estados Unidos (US), mas realizando uma requisição de arquivo para uma máquina virtual criada na Europa (EU), replicando o **Cenário 2** dos testes realizados acima. Os resultados destes testes podem ser vistos na Figura 5.3.

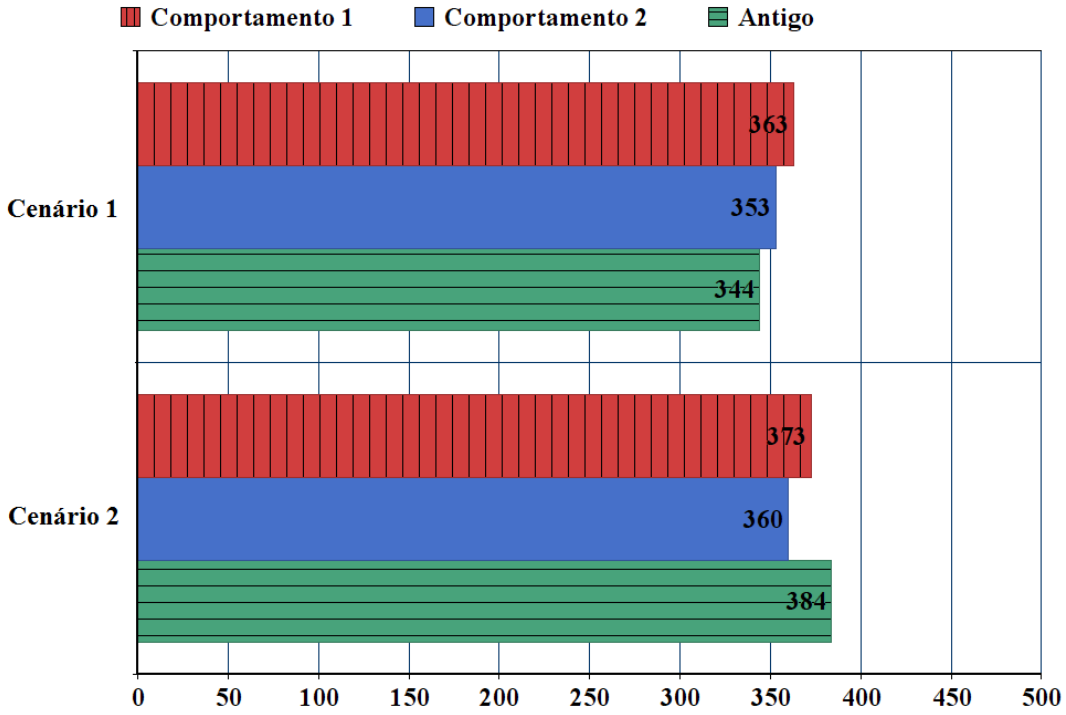

Figura 5.3: Comparação entre os Tempos de Execução dos Serviços de Armazenamento Novo e Antigo (tempos em segundos).

Conforme pode ser observado na Figura 5.3, no **Cenário 1**, em que a transferência do arquivo é feita entre o nó de armazenamento e o nó de processamento criados em uma mesma região geográfica, no caso, os Estados Unidos (US), o novo serviço de armazenamento apresentou uma pequena perda de *performance*. Entretanto, para o **Cenário 2**, no qual a transferência do arquivo foi feita entre o nó de armazenamento e o nó de processamento criados em regiões geográficas distintas, Estados Unidos (US) e Europa (EU), houve um pequeno ganho de *performance*.

Esse ganho de *performance* ocorre pois no **Cenário 2**, onde a velocidade de transferência é inferior àquela obtida no **Cenário 1**, o tempo necessário para que se realize o *download* por completo do arquivo antes da execução, como observado no **Comportamento 1**, é superior ao tempo extra que este mesmo *download* leva para ser realizado de maneira simultânea à execução da tarefa, como no **Comportamento 2**.

É importante ressaltar o fato de que a proposta deste novo serviço de armazenamento não era uma melhora especificamente na *performance*, mas sim na praticidade do acesso, visto que estes arquivos armazenados na federação podem ser acessados de maneira distribuída e sob demanda de qualquer máquina virtual instada no BioNimbuZ. Além disso, passa-se a teruma maior taxa de confiabilidade e de disponibilidade dos arquivos presentes

na federação, pois estes estão armazenados em serviços oferecidos por múltiplos provedores de nuvem, que garantem através de seus SLAs níveis satisfatórios de disponibilidade para estes arquivos. Como essas melhorias propostas foram alcançadas, pode-se dizer que estas pequenas discrepâncias de *performance* são pouco relevantes para o real uso do novo serviço, que atingiu o seu real objetivo.

5.5 Considerações Finais

Neste capítulo foram apresentados os testes realizados para comprovar o funcionamento do novo serviço de armazenamento e medir sua *performance* comparando-a ao serviço de armazenamento antigo.

Assim, com os resultados dos testes presentes neste capítulo, comprovou-se que os objetivos propostos foram alcançados, sendo estes uma melhoria na praticidade de acesso, uma obtenção maior de confiabilidade no serviço e uma maior disponibilidade dos arquivos armazenados da federação do BioNimbuZ.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um novo serviço de armazenamento para a plataforma de nuvens federadas BioNimbuZ, buscando uma melhoria no cenário atual da plataforma, no qual os discos das instâncias de máquina virtual da federação eram utilizados para realizar o armazenamento de arquivos na federação. A finalidade do novo serviço de armazenamento foi integrar serviços de armazenamento oferecidos pelos grandes provedores de nuvem a fim de oferecer um serviço mais robusto e confiável ao seu usuário final, graças aos serviços externos utilizados que já oferecem SLAs consolidados e satisfatórios. Com isso, a plataforma do BioNimbuZ garante tanto uma federação das nuvens para os serviços de processamento quanto para os serviços de armazenamento.

Para isso, foi realizado um estudo detalhado acerca dos serviços de armazenamento disponibilizados pelos grandes provedores de nuvem, sendo feita uma análise de seus pontos negativos e positivos, para que fossem escolhidos os serviços de armazenamento utilizados para integrar o novo serviço de armazenamento do BioNimbuZ.

Além disso, foram realizados testes com o novo serviço de armazenamento, nos quais foi verificado com sucesso a integração e o funcionamento do mesmo em um cenário de execução de um *workflow* real de Bioinformática, garantindo um serviço mais robusto e uma maior disponibilidade e praticidade no acesso aos arquivos presentes na federação, já que agora os arquivos podem ser acessados de maneira distribuída e sob demanda de qualquer instância de máquina virtual da federação. Assim, foram realizados testes para medir o desempenho deste novo serviço de armazenamento, que apresentaram resultados indicando que a *performance* do novo serviço é muito semelhante à do serviço de armazenamento anteriormente presente no BioNimbuZ.

Como trabalhos futuros, sugere-se que sejam aplicadas métricas relacionadas ao armazenamento de arquivos, como o tempo necessário que leva para um arquivo ser transferido para uma máquina, na política de escalonamento do BioNimbuZ, que atualmente só considera métricas relacionadas a poder de processamento na hora do cálculo de custo e de

tempo em seu serviço.

Por fim, sugere-se que seja implementado na plataforma do BioNimbuZ um banco de dados distribuído, como por exemplo o Cassandra [1], para tornar a persistência dos dados da federação mais consistente e confiável, aumentando também a tolerância a falhas do sistema.

[1] http://cassandra.apache.org/

Referências

[1] A. Celesti, F. Tusa, M. Villari and A. Puliafito. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 337–345. IEEE, 2010. x, 1, 20, 22, 24, 26

[2] A. Celesti, M. Fazio, M. Villari and A. Puliafito. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. *Journal of Network and Computer Applications*, 59:208–218, 2016. 2, 38, 46

[3] A. Vajda and P. Stenström. Multi-core processors. `https://www.google.com/patents/US20140033217`, January 30 2014. US Patent App. 14/110,140. 7

[4] D. R. Azevedo and T. B. F. Júnior. BioCirrus: Uma Nova Política de Armazenamento para a Plataforma BioNimbuZ de Nuvem Federada. Monografia de graduação, Departamento de Ciência da Computação, Universidade de Brasília, 2015. 36, 38

[5] B. P. Rimal, E. Choi and I. Lumb. A taxonomy and survey of cloud computing systems. *In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference*, pages 54–51, 2009. 1, 17

[6] C. Cachin, R. Haas and M. Vukolić. Dependable storage in the intercloud. Technical report, Research Report RZ, 3783, 2010. 2, 38, 46

[7] H. H. P. M. Costa. Controle de Acesso na Plataforma de Nuvem Federada BioNimbuZ. Monografia de graduação, Departamento de Ciência da Computação, Universidade de Brasília, 2015. x, 25, 37

[8] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design.* Addison-Wesley, 2005. 7

[9] G. F. Pfister. *In search of clusters.* Prentice-Hall, 1998. 10

[10] R. F. Gallon. Política de Armazenamento de Dados em Nuvens Federadas para Dados Biológicos. Dissertação de mestrado, Departamento de Ciência da Computação, Universidade de Brasília, 2014. 25, 36

[11] I. Foster, C. Kesselman and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001. x, 12, 13

[12] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop*, pages 1–10, 2008. x, 13, 14, 15, 17

[13] S. M. Mane M. Stephens J. C. Marioni, C. E. Mason and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517, 2008. 54

[14] J. D. Sloan. *High performance Linux clusters with OSCAR, Rocks and MPI*. Rodopi, 2009. x, 9, 10, 11, 12

[15] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008. x, 14, 15, 16, 17

[16] M. Factor, K. Meth, D. Naor, O. Rodeh and J. Satran. Object storage: The future building block for storage systems. In *2005 IEEE International Symposium on Mass Storage Systems and Technology*, pages 119–123. IEEE, 2005. 41, 42

[17] M. Pitanga. *Construindo Supercomputadores com Linux*. Brasport, 2004. x, 7

[18] M. V. Steen and A. Tanenbaum. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2 edition, 2006. x, xii, 7, 9, 26

[19] C. Mann. The end of Moore's law. *Technology Review*, 103(3):42–48, 2000. 7

[20] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. *IEEE Solid-State Circuits Newsletter*, 3(20):33–35, 2006. 6

[21] B. R. Moura and D. L. Bacelar. Política para Armazenamento de Arquivos no ZooNimbus. Monografia de graduação, Departamento de Ciência da Computação, Universidade de Brasília, 2013. x, 2, 25, 26, 27, 36, 38

[22] G. S. S. Oliveira. Acosched - um Escalonador para o Ambiente de Nuvem Federada ZooNimbus. Monografia de graduação, Departamento de Ciência da Computação, Universidade de Brasília, 2013. 26

[23] P. Mell and T. Grace. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009. 14, 17

[24] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, 2009. 13

[25] R. Buyya, R. Rajkumar and R. N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010. x, 21, 22, 23

[26] V. A. Ramos. Um Sistema Gerenciador de Workflows Científicos Para a Plataforma de Nuvens Federadas BioNimbuZ. Monografia de graduação, Departamento de Ciência da Computação, Universidade de Brasília, 2016. x, 25, 31, 32

[27] M. D. Robinson and A. Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(3):1, 2010. 54

[28] S. Ramgovind, M. M. Eloff and E. Smith. The management of security in cloud computing. In *Information Security for South Africa (ISSA), 2010*, pages 1–7. IEEE, 2010. 2, 18

[29] H. V. Saldanha. BioNimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática. Dissertação de mestrado, Departamento de Ciência da Computação, Universidade de Brasília, 2012. x, 2, 17, 20, 25

[30] T. Bittman. The evolution of the cloud computing market. gartner blog network. `http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market/`, 2008. Acessado em 18 de setembro de 2015. x, 20