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

Arquitetura de um Controlador de Elasticidade para Nuvens Federadas

Guilherme Fay Vergara

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

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

Brasília
2017

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Fay Vergara, Guilherme
FV494a Arquitetura de um Controlador de Elasticidade para Nuvens Federadas / Guilherme Fay Vergara; orientador Aletéia Patrícia Favacho de Araújo. -- Brasília, 2017.
89 p.

Tese (Doutorado - Mestrado em Informática) -- Universidade de Brasília, 2017.

1. Computação em Nuvem. 2. Controlador de Elasticidade. 3. BioNimbuZ. I. Patrícia Favacho de Araújo, Aletéia , orient. II. Título.

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

Arquitetura de um Controlador de Elasticidade para Nuvens Federadas

Guilherme Fay Vergara

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

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

Prof.ª Dr.ª Alba Cristina M. A. de Melo
CIC/UnB

Prof.ª Dr.ª Aline de Paula Nascimento
IC/UFF

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

Brasília, 23 de fevereiro de 2017

Dedicatória

Dedico este trabalho à minha família, por sua capacidade de acreditar e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado saúde e força para superar as dificuldades.

A minha orientadora Aletéia, pelo suporte no desenvolvimento deste trabalho, pelos incentivos, correções e puxões de orelha durante todo este processo, não somente por ter me ensinado, mas por ter me feito aprender, meus sinceros agradecimentos.

A minha mãe, que me deu apoio e incentivo nas horas difíceis, de desânimo e abatimento. Ao meu pai que apesar de todas as dificuldades sempre esteve ao meu lado me dando conselhos e sempre servindo de inspiração. Meus eternos agradecimentos.

Aos meus colegas do laboratório LABID, em especial ao Breno, que estiveram comigo durante toda essa batalha, e a todos que direta ou indiretamente fizeram parte desta caminhada, o meu muito obrigado.

Resumo

Com a constante evolução das aplicações, surgiu a necessidade de se ter um ambiente no qual fosse possível processar aplicações de forma dinâmica, escalável e sob demanda. Assim, emergiu a plataforma de computação em nuvem. Nesse ambiente, os recursos são alocados conforme a demanda de utilização. Para que seja possível um aumento/decréscimo escalar no número de recursos, a característica de elasticidade é fundamental. A elasticidade é definida como a capacidade da nuvem de se adaptar às alterações na quantidade de recursos ou de solicitar/liberar recursos de acordo com a necessidade. Nesse cenário, aplicações científicas, tais como as de Bioinformática, tem se beneficiado do conceito de nuvens pela sua característica de tratar grandes quantidades de dados, que demandam significativas quantidades de recursos computacionais. Neste cenário de execuções longas com grande número de recursos a elasticidade possibilita um uso mais adequado da nuvem, alocando os recursos somente nos momentos que são necessários. Assim sendo, este trabalho propõe uma arquitetura de um Controlador de Elasticidade para nuvens federadas, que seja capaz de provisionar/desprovisionar máquinas virtuais nas diversas nuvens da federação, além de também ser capaz de aumentar/diminuir a capacidade computacional de cada um dos recursos de maneira automática.

Palavras-chave: Computação em nuvem, Nuvem Federada e Elasticidade

Abstract

With the constant evolution of applications, there arose a need to have an environment which was able to process applications dynamically, scalabe and on demand. Thus emerged the cloud computing platform. In this environment, resources are allocated according to demand. To be able to scale on demand, the characteristic of the elasticity is fundamental. Elasticity is defined as the ability of the cloud to adapt to changes in the amount of resources or to request/release resources as needed. In this scenario, scientific applications, such as those of Bioinformatics, have benefited from the concept of cloud computing due, to their requirement of handling large amounts of data which demand significant amounts of computational resources. This work proposes an architecture for a Elasticity Controller for federated clouds, capable of provisioning/deprovisioning virtual machines across the various clouds of the federation, as well as being able to automatically increase/decrease the computational capacity of each of the resources.

Keywords: Cloud computing, Federated cloud and Elasticity

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O aumento na demanda computacional pela indústria, comércio e academia, nos quais a maior parte do processamento computacional é realizado por servidores locais e *datacenters* proprietários, é uma realidade. Entretanto, nesse modelo computacional, a capacidade de processamento adquirido, muitas vezes não é utilizado por completo. Por consequência, paga-se por energia, manutenção e tecnologias que são subutilizadas. Neste contexto, surgiu a ideia de que, em substituição ao pagamento por uma estrutura proprietária, de manutenção cara, que às vezes fica ociosa, fosse utilizada uma estrutura que disponibilizasse os seus recursos de acordo com a necessidade de cada usuário. Dessa forma, surgiu a computação em nuvem. Nesse paradigma de computação, os recursos são alocados conforme a demanda de utilização. A computação em nuvem busca reduzir o custo computacional, além de aumentar a confiabilidade e a flexibilidade, de tal forma que os recursos e as aplicações sejam adquiridos como serviços [8]. Para que o compartilhamento dos recursos da nuvem fosse garantido, foi necessário o uso de virtualização, permitindo que vários ambientes virtuais sejam executados sobre um hardware físico. Isso possibilita a otimização do uso de recursos e a economia de escala [53].

Em sistemas de computação em nuvem, diferentes características estão presentes, tais como a adaptação automática, a elasticidade, a interoperabilidade, o pagamento pelo uso do recurso e de acordo com o nível de serviço [84]. Dessa forma, uma das principais vantagens oferecidas pela nuvem é a sua capacidade de expansão dinâmica, permitindo executar serviços que variam suas necessidades por recursos no tempo.

Dessa forma, destaca-se como uma importante característica do ambiente de nuvem a chamada elasticidade. Segundo Mell *et al.* [53] é a característica que permite um rápido provisionamento e desprovisionamento de recursos, com capacidade de recursos virtuais praticamente infinita a qualquer momento. A elasticidade vem sendo oferecida pelos provedores de nuvem por meio de diferentes mecanismos, objetivando evitar o provisionamento excessivo ou insuficiente de nuvens [13].

Nesse cenário, aplicações científicas, tais como as de Bioinformática, tem se beneficiado do conceito de nuvem pela sua característica de tratar grandes quantidades de dados, que consomem e produzem enormes conjuntos de dados, e inerentemente demandam significativas quantidades de recursos computacionais [60]. Toda esta demanda computacional entregue pela computação em nuvem com tecnologias de alto desempenho, pode ser vital para *workflows* científicos alcançarem a automação, pois a execução de *workflows* científicos é computacionalmente intensiva e demanda alta disponibilidade de recursos computacionais. Um *workflow* científico pode ser definido como a caracterização formal de um processo científico que consiste em uma sequência ordenada de atividades [21].

Todavia, como a quantidade de dados a ser processada aumenta cada vez mais no mundo da computação, surge a necessidade de ter um poder computacional cada vez maior, para que se possa atender pedidos em um tempo hábil, de modo automático e dinâmico. Porém, uma nuvem possui recursos limitados, e por conta disso surgiu a ideia de se integrar nuvens. A federação de nuvens [72] é um conjunto de nuvens que possui todos os seus recursos gerenciados por meio de uma interface conectada a todas elas, de forma que se uma nuvem não tiver recursos para determinado processamento, uma outra nuvem, que esteja com recursos disponíveis no momento, possa realizar este processamento. O objetivo é aumentar os recursos disponíveis, pois se um recurso de uma nuvem se esgotar, uma outra nuvem pode emprestar os seus recursos, potencializando sua função.

Nesse contexto, Saldanha *et al.* [73] propôs uma plataforma de nuvem federada para executar diferentes aplicações na federação, por meio de um ambiente transparente, flexível, eficiente e tolerante à falhas, com acesso a grande poder de processamento e de armazenamento. Essa plataforma é chamada BioNimbuZ.

Dessa forma, o BioNimbuZ foi escolhido como estudo de caso para a implementação de um controlador de elasticidade, de forma que a arquitetura possa provisionar/desprovisionar as máquinas virtuais que deseja para cada tarefa, e também aumentar/diminuir a capacidade computacional de cada um dos recursos melhorando a capacidade de execução da federação.

Assim sendo, este trabalho propõe uma arquitetura de um controlador de elasticidade para nuvens federadas, que seja capaz de provisionar/desprovisionar máquinas virtuais nas diversas nuvens da federação, além de também ser capaz de aumentar/diminuir a capacidade computacional de cada um dos recursos de maneira automática e reativa,

1.1 Motivação

A federação de nuvens tem se mostrado um conceito capaz de integrar diferentes infraestruturas, proporcionando uma maior flexibilidade na escolha de provedores, e uma visão na qual a quantidade de armazenamento e de processamento seja vista como recursos ilimitados. Porém, não existe uma padronização para um controlador de elasticidade que possa se integrar com outras nuvens.

Este trabalho tem como motivação a necessidade de se ter um serviço de elasticidade inter-nuvens, que seja capaz de aumentar/diminuir os recursos de acordo com a demanda em qualquer nuvem integrante da federação.

1.2 Problema

A elasticidade de máquinas virtuais em nuvens federadas, por muitas vezes pode se tornar uma tarefa muito complexa e desafiadora. Assim sendo, o problema que este trabalho tenta resolver está em controlar a elasticidade em um ambiente de nuvens federadas, ou seja, controlar a criação/deleção ou aumentar/diminuir a capacidade de processamento/armazenamento das máquinas virtuais da federação para que elas se adequem a carga de trabalho atual.

1.3 Objetivos

Este trabalho tem como objetivo geral desenvolver para um ambiente de nuvens federadas um controlador de elasticidade, que suporte o provisionamento/desprovisionamento automático e sob demanda de máquinas virtuais.

Para cumprir o objetivo geral desde trabalho, faz-se necessário atingir os seguintes objetivos específicos:

- Definir as ações de elasticidade que ativam o serviço de elasticidade no ambiente de nuvens federadas;
- Propor um controlador de elasticidade horizontal para criar e deletar dinamicamente máquinas virtuais, de acordo com o que foi submetido para a federação;
- Propor um controlador de elasticidade vertical para que possa aumentar e diminuir a capacidade de processamento de uma máquina virtual da plataforma;
- Integrar o serviço de elasticidade ao ambiente de nuvens federadas BioNimbuZ;

1.4 Estrutura da Dissertação

Esta dissertação de mestrado está dividida em mais cinco capítulos. No Capítulo 2 são apresentados os principais conceitos de computação em nuvem, suas características, sua arquitetura e os modelos de implantação. O Capítulo 3 aborda os conceitos de elasticidade em computação em nuvem, e traz os principais trabalhos relacionados. O Capítulo 4 apresenta a plataforma BioNimbuZ, mostrando sua arquitetura e sua organização lógica. No Capítulo 5 é descrita a proposta de um controlador de elasticidade para o ambiente de federação de nuvem. E por fim, o Capítulo 6 apresenta as conclusões obtidas com a realização desta pesquisa e, alguns trabalhos futuros.

Capítulo 2

Referencial Teórico

Este capítulo faz uma revisão bibliográfica acerca dos principais temas desta dissertação. Na Seção 2.1 são apresentadas as principais características e as definições de computação em nuvem. Em seguida, na Seção 2.2 são abordados os conceitos de federação de nuvens. Na Seção 2.3 são apresentados os *workflows* de Bioinformática, e por fim na seção 2.4 as considerações finais deste Capítulo.

2.1 Computação em Nuvem

A definição de computação em nuvem segundo o NIST (*National Institute of Standards and Technology*) [53] é *"A computação em nuvem é um modelo para acesso conveniente, sob demanda, e de qualquer lugar, a uma rede compartilhada de recursos de computação (isto é, redes, servidores, armazenamento, aplicativos e serviços) que possam ser prontamente disponibilizados e liberados com um esforço mínimo de gestão ou de interação com o provedor de serviços."*

Já para Buyya *et al.* [8], a definição de computação em nuvem é: *"Nuvem é um sistema de computação paralela e distribuída que consiste em um conjunto de computadores interligados e virtualizados que são dinamicamente providos e apresentados como um ou mais recursos de computação unificada baseada em contratos de níveis de serviço estabelecidos através de negociação entre o prestador de serviço e os consumidores."*

Como pode ser observado, a definição de computação em nuvem na literatura não é unânime, mostrando que essa é uma área da computação que ainda está em desenvolvimento. Porém, Vaquero *et al.* [84], fizeram uma avaliação de mais de 20 definições e sugeriram uma proposta unificada, que pode ser traduzida como: *"Um grande conjunto de recursos virtualizados e compartilhados (hardware, plataformas de desenvolvimento ou software) que podem ser facilmente acessados e utilizados. Esses recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga variável de trabalho, permitindo*

uso otimizado dos mesmos. Este conjunto de recursos é tipicamente explorado por um modelo de pagamento por utilização chamado de pay-per-use em que as garantias são oferecidas pelo provedor de infraestrutura por meio de contratos de serviço personalizados (Service Level Agreement – SLA)".

Assim sendo, nota-se que as grandes vantagens da computação em nuvem são que o cliente poderá ter acesso a arquivos/aplicações em qualquer lugar, bastando ter uma conexão com a Internet. Os custos com hardware, por muitas vezes, são menores, pois o cliente não precisa mais comprar computadores com alto poder de processamento e de armazenamento para executar sua aplicação, pois essa responsabilidade passa a ser dos provedores de nuvens.

Segundo Badger *et al.* [2], a computação em nuvem possui algumas características importantes. As principais são:

1. **Auto-atendimento sob demanda (*On-Demand Self-Service*):** Este aspecto permite que a nuvem ofereça ao consumidor os seus serviços, tais como armazenamento e processamento, de forma direta e transparente, sem a participação de nenhum administrador dos provedores de nuvem, e de forma que atendas as suas necessidades.

2. **Amplo acesso à rede (*Ubiquitous Network Access*):** Todos os serviços da nuvem podem ser acessados de maneira padronizada pela grande maioria de aparelhos que possuem acesso à Internet. A nuvem se utiliza de mecanismos que promovem o uso de plataformas heterogêneas. Este tipo de acesso facilita o acesso para os usuários que já estão acostumados com certo padrão de interface, e facilita aos programadores, pois não precisam desenhar uma interface nova para cada tipo de dispositivo.

3. ***Pool de Recursos* (*Resource Pooling*):** Este modelo visa atender a múltiplos clientes, simplesmente atribuindo ou retribuindo os recursos virtuais dinamicamente conforme sua demanda através do modelo multi-inquilino (*multi-tenancy*) [53]. O serviço disponibilizado na nuvem, tem que ser capaz de gerenciar a adesão de novos clientes, enquanto por exemplo, garante a segurança de dados (para que um cliente não acesse os dados dos outros), ou a escalabilidade (para que a infraestrutura suporte o aumento de carga inerente a novas adesões).O cliente não possui um controle ou conhecimento sobre o local onde seus recursos estão sendo alocados, tendo somente uma informação mais ampla, como o país em que se encontra o *Datacenter*.

4. **Transparência:** O usuário pode não conhecer onde estão alocados seus recursos computacionais, sejam eles VMs, arquivos, programas, etc., tendo uma visão so-

mente da "ponta do *iceberg*", ou seja, a nuvem mascara toda a infraestrutura e oferece ao cliente somente uma interface para acesso, tornando assim tudo transparente.

5. **Elasticidade (*Elasticity*):** A elasticidade está associada à necessidade de se aumentar ou diminuir rapidamente os recursos alocados, visando um maior aproveitamento das capacidades disponíveis. Por vezes, esta característica dá a ilusão de recursos computacionais infinitos. Essa é uma das principais características que torna a computação em nuvem um serviço muito atrativo, e é o foco deste trabalho.

6. **Serviços Mensuráveis (*Measured Service*):** Todos os serviços podem ser monitorados e controlados, automaticamente, pela nuvem. Isso permite que o fornecedor da nuvem possa cobrar do cliente exatamente o que foi consumido de recurso por ele, e ao cliente, é dada a facilidade dele controlar seus gastos.

2.1.1 Arquitetura da Computação em Nuvem

Na literatura há diferentes propostas de arquitetura [47] [86], e a Figura 2.1 apresenta uma destas propostas. Nela é possível identificar três atores, que são os Prestadores de Serviço (*Service Providers*), os Prestadores de Infraestrutura (*Infrastructure Providers*) e os Usuários de Serviço (*Service Users*).

Os provedores de infraestrutura disponibilizam recursos para que os provedores de serviços hospedem suas aplicações sem precisarem se preocupar com as questões de estrutura física, a fim de obter escalabilidade e flexibilidade. Os usuários de serviço acessam as aplicações que foram colocadas a disposição através da Internet.Tanto usuários de serviço quanto provedores de serviço pagam apenas por aquilo que consumirem, o chamado *pay-per-use* [84].

Essa arquitetura de nuvem está dividida em três camadas distintas. A mais próxima do usuário é a que contém se encontram os serviços, ela possui uma interface para que os clientes tenham acesso às aplicações que foram disponibilizadas. A camada mais baixa é a infraestrutura de fato. Ali estão localizados os servidores, os *datacenters*, a rede e toda a parte física que compõe a nuvem. A camada do meio fornece facilidades para que o provedor de serviços consiga disponibilizar suas aplicações, sem ter que se preocupar com as individualidades do hardware [84].

Uma outra proposta de arquitetura foi feita por Foster *et al.* [29] e está representada na Figura 2.2. A camada de infraestrutura contém os recursos computacionais, como os recursos de armazenamento e os de processamento, ou seja é o hardware propriamente dito.

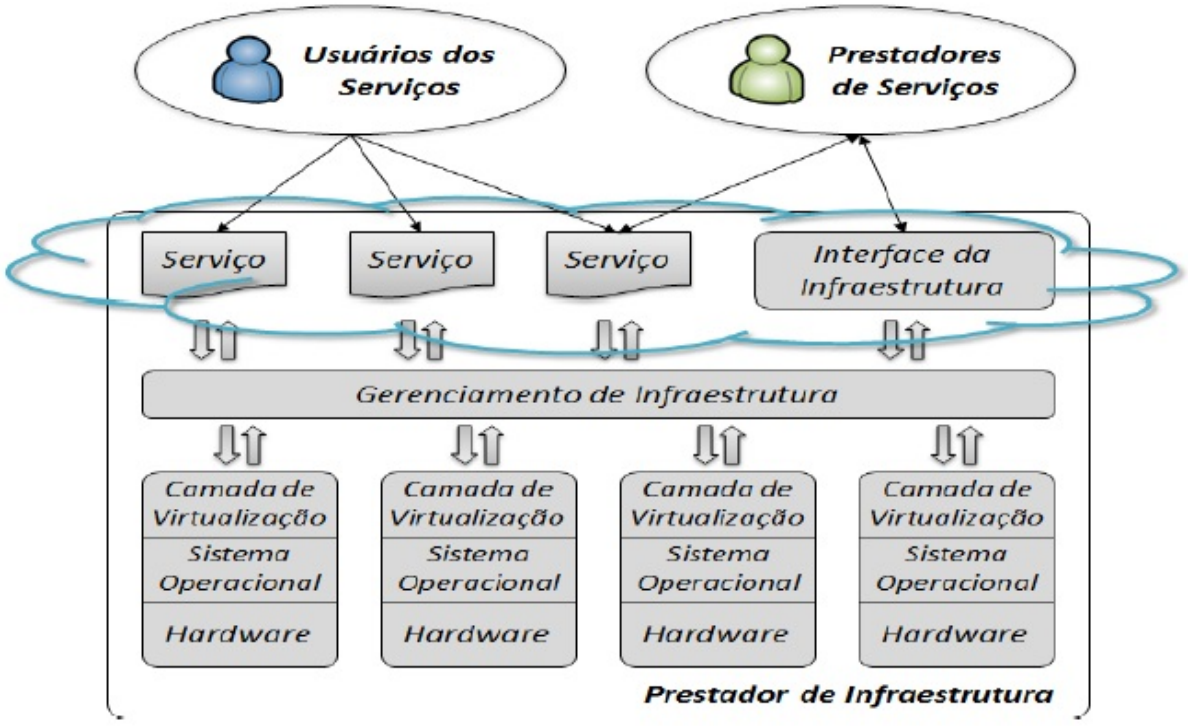

Figura 2.1: Arquitetura da Computação em Nuvem, adaptado de Vaquero *et al.* [84].

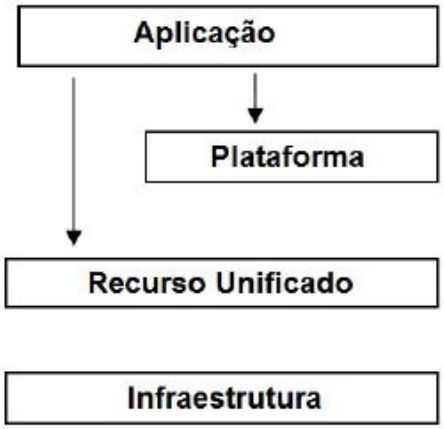

Figura 2.2: Arquitetura da Computação em Nuvem, adaptado de Foster *et al.* [29].

A camada de recursos unificados contém os recursos que foram encapsulados, geralmente, por meio da virtualização, e estão disponíveis tanto para os usuários finais quanto para a camada superior. Assim, *clusters* e computadores virtuais são exemplos de recursos desta camada. A camada de plataforma consiste em um conjunto de ferramentas especializadas que estão executando em cima da camada de recursos unificados, tais como

plataformas de desenvolvimento. E por fim, a camada de aplicação que contém as aplicações que executam na nuvem.

2.1.2 Modelos de Serviço

Os serviços oferecidos no paradigma de computação em nuvem podem ser divididos em categorias. Apesar de ser possível encontrar na literatura propostas com mais de três classes [70], o mais comum é dividir os serviços em *Infrastructure-as-a-Service - IaaS*, *Plataform-as-a-Service - PaaS* e *Software-as-a-Service - SaaS*, descritos a seguir:

- **Infraestrutura como Serviço (*Infrastructure-as-a-Service - IaaS*):**

  Conhecido também como *Hardware as a service* (HaaS) [53], esse modelo oferece a virtualização de recursos, tanto computacionais como de armazenamento e de comunicação. Esta camada permite prover serviços sob demanda, que podem ser executados por diferentes sistemas operacionais, e permite uma pilha de aplicativos customizáveis.

  Logo, os desenvolvedores podem se concentrar mais na produção e não na infraestrutura, permitindo um desenvolvimento mais rápido, e reduzindo bastante os gastos da empresa com tais recursos computacionais.

  O termo IaaS refere-se a um tipo de serviço baseado em técnicas de virtualização de recursos computacionais oferecidos através da Internet. Tais recursos podem ser escalados conforme a necessidade do cliente, e esta característica torna este serviço muito atrativo, pois os clientes pagam somente pelo que usarem, não pagando pelo tempo que seus servidores ficarem ociosos. Dessa forma, sua utilização é recomendada quando se tem uma demanda volátil, como por exemplo em lojas virtuais, que vendem muito no Natal, e no resto do ano há pouca procura. Também é aconselhado para empresas que tendem a crescer rapidamente, e não tem capital suficiente para uma infraestrutura própria. Contudo, não recomenda-se a utilização de IaaS quando as aplicações da empresa necessitam de hardware específico, ou os níveis de desempenho necessários para as aplicações tenham limite pelo provedor. Além disso, não é aconselhado para empresas em que suas legislações não permitam o armazenamento de dados fora da empresa.

- **Plataforma como Serviço (*Plataform-as-a-Service - PaaS*):**

  O PaaS fornece todos os recursos necessários para construir aplicações e serviços, sem ter que realizar a transferência ou a instalação de algum software. Plataforma como serviço fornece aplicações de projeto, teste, desenvolvimento entre outras. O usuário não administra ou controla a infraestrutura, ou seja, a camada de IaaS

torna-se invisível para o usuário da PaaS, não se preocupando assim com hardware, porém ele possui controle sobre as aplicações implantadas [53].

É aconselhável o seu uso quando há necessidade de trabalhos em equipe, integração e triagem de serviços e banco de dados. Aconselha-se também quando há a necessidade de um ambiente complexo para a aplicação. A grande desvantagem desse modelo é a falta de portabilidade entre provedores quando há utilização de linguagens proprietárias.

- **Software como Serviço (*Software-as-a-Service - SaaS*):**

  É o modelo de serviço no qual as aplicações são fornecidas como *host* para o cliente, que as acessa via Internet. A vantagem na perspectiva do cliente se dá em razão de que quando o software é fornecido através de um *host*, todo o suporte a esse software passa a ser de responsabilidade e obrigação do prestador da infraestrutura da nuvem, desobrigando o usuário. O provedor faz todas as atualizações necessárias, e mantém a infraestrutura do software em funcionamento [53].

  É importante destacar que o SaaS ajuda a reduzir os custos, visto que dispensa a aquisição de licenças de sistemas, garantindo ao usuário que ele somente pagará pelo que realmente usar. E mais, o fato de o SaaS ser disponibilizado através da Internet, possibilita que os softwares por ele oferecidos possam ser acessados através de qualquer dispositivo com a Internet, podendo ser usado por *tablets*, *smartfones*, etc, aumentando a acessibilidade e a robustez do SaaS.

  A sua utilização é recomendada quando há a necessidade de acesso às aplicações por meio de acesso remoto (dispositivos móveis), e também quando é necessária a utilização de algum software por um curto período de tempo.

2.1.3 Modelos de Implantação

Existem diferentes modos para implementar um serviço de computação em nuvem. O modelo de implantação irá depender da necessidade da aplicação cliente. Segundo Mell *et al.* [53], os modelos de implantação da computação em nuvem podem ser divididos em nuvens públicas, privadas, comunitárias e híbridas, as quais são descritas a seguir:

- **Modelo Privado:**

  O modelo de implantação privado é aquele em que é construído, operado e mantido pela operadora da nuvem, e compartilhado por toda a organização. Este modelo é ideal para organizações que querem entrar no mundo da computação em nuvem, mas não abrem mão do controle da mesma. Um dos principais motivos que tem levado

as organizações a escolherem o modelo privado é a segurança que este modelo provê, pois segundo Taurion [80], diretor de novas tecnologias aplicadas da IBM Brasil, o que a nuvem privada tem de diferente é o fato de restringir acesso, pois se encontra atrás do *firewall* da organização. Ela é uma forma de aderir à tecnologia mantendo controle do nível de serviço e aderência às regras de segurança. Instituições financeiras como bancos, grandes empresas multinacionais como Microsoft e IBM, fazem o uso do modelo privado. Segundo Chirigati [14], uma nuvem privada é, em geral, construída sobre um *datacenter* privado.

- **Modelo Comunitário:**

  Já o modelo comunitário é uma variação do modelo privado, normalmente usado por organizações que tem um objetivo de negócio em comum (por exemplo, para ser usada por um consórcio de compras), e compartilham do mesmo recurso (dados, software, hardware, etc). Este modelo, normalmente, acarreta em uma diminuição dos custos da implantação, pois os servidores não são exclusivos de uma organização, o que por outro lado diminui a segurança. Segundo Mell *et al.* [53], este modelo pode ser administrado por organizações ou por um terceiro, e pode existir localmente ou remotamente.

- **Modelo Público:**

  O modelo público é provido e designado para serviços de propósito geral. Ele utiliza-se do modelo de comercialização *pay-per-use*, ou seja, pague o quanto usar. Pelo motivo dos clientes pagarem somente pelo o que usarem isso normalmente diminui os custos drasticamente, pois o cliente não precisa gastar com um *datacenter* exclusivo. Este modelo, normalmente, é usado por pequenas empresas, *startups* ou setores de TI dentro de empresas maiores de outro ramo, justamente, pelo baixo custo e pela fácil aquisição de seus serviços.

- **Modelo Híbrido:**

  O modelo híbrido por sua vez, como o próprio nome diz, é uma composição entre os outros dois ou mais modelos. É um modelo, por exemplo, que amplia os recursos de um modelo privado, podendo utilizar os recursos de um modelo público também. Segundo Chirigati [14], o termo "computação em ondas" é, em geral, utilizado quando se refere às nuvens híbridas.

2.2 Federação de Nuvens

A computação em nuvem tem buscado atingir níveis cada vez melhores de eficiência na disponibilização de serviços, e com o passar dos anos mais e mais necessidades foram surgindo. Assim, a utilização de nuvens de forma isolada passou a não ser mais suficiente para algumas aplicações.

Além das grandes nuvens públicas, mantidas por grandes organizações, centenas de outras nuvens menores, privadas ou híbridas, vêm sendo implantadas de maneira heterogênea e independente. Com isso, surge o cenário em que a federação de nuvens computacionais interoperáveis se torna uma alternativa interessante para otimizar o uso dos recursos oferecidos por essas diversas instituições.

Por definição, uma federação de nuvens é um sistema cooperativo de duas ou mais provedoras de serviço de computação em nuvem [11]. Esta cooperação, em nível de cliente de nuvens, é interessante por quebrar a dependência por um único provedor, melhorando assim a disponibilidade de serviço da federação, reduzindo custos, e também aumentando o número de combinações de instâncias.

Bittman [6] dividiu a evolução do paradigma de computação em nuvem em três fases. A primeira fase é chamada de monolítica e se caracteriza pelas ilhas proprietárias, com serviços fornecidos por empresas de grande porte, como Google [36], Amazon [50] e Microsoft [54].

Na segunda fase, chamada de cadeia vertical de fornecimento, ainda se tem o foco nos ambientes proprietários, mas as empresas começam a utilizar alguns serviços de outras nuvens, sendo vista como o começo da integração. E por último, tem-se a terceira fase, a federação horizontal, na qual pequenos provedores se juntam para aumentar sua escalabilidade e eficiência na utilização dos recursos, e começam a ser discutidos padrões de interoperabilidade. Atualmente, tem-se vivido a terceira fase, alcançando a etapa em que os pequenos provedores se aliam horizontalmente.

Por não existir um padrão para a federação de nuvens computacionais, uma nova arquitetura foi proposta, chamada de BioNimbuZ [49], que tem como objetivo garantir de forma dinâmica, transparente e escalável a execução de aplicações em nuvens federadas.

Assim sendo, o BioNimbuZ [49] é uma proposta de arquitetura para a realização de uma federação horizontal, que visa facilitar a integração entre diversos provedores de nuvem. O intuito é garantir a máxima eficiência na utilização dos recursos, e aumentar a escalabilidade dos provedores. Esta proposta é detalhada no Capítulo 4.

Contudo, há várias outras plataformas de federação de nuvens na literatura [39], [22], [64]

O Bionimbus [39] é um sistema *cloudbased* de código aberto para gerenciamento, análise e compartilhamento de dados genômicos que foi desenvolvido pelo Instituto de Ge-

nómica e Biologia de Sistemas (IGSB) na Universidade de Chicago. O Bionimbus foi desenvolvido para promover a tecnologia de código aberto para gerenciamento, análise, transporte e compartilhamento de grandes conjuntos de dados genômicos de forma segura e compatível. A comunidade Bionimbus contém uma variedade de conjuntos de dados biológicos públicos, incluindo um conjunto de dados de mais de 1.000 genomas. Os usuários podem criar uma imagem de uma VM que pode ser instanciado. Essas imagens executam o sistema operacional Linux com diferentes núcleos de computação (1 a 4) e memória (3 a 15GB). As capacidades de armazenamento de dados são 10GB.

O CometCloud [22] é um *framework* autônomo projetado para habilitar plataformas de dados e computação altamente heterogêneas e dinamicamente federadas, que podem suportar fluxos de trabalho de aplicativos com requisitos diversos e em constante mudança. Isso ocorre por meio de uma federação autônoma e sob demanda de recursos de dados e computação distribuídos geograficamente, usando abstrações de nuvens elásticas e plataformas de ciência como serviço. Consequentemente, o CometCloud pode criar um ambiente ágil e programável que evolui de forma autônoma ao longo do tempo, adaptando-se a alterações nos requisitos de infraestrutura e aplicação.

O projeto mOSAIC [64] construiu um *framework* para gerenciar diferentes provedores de nuvem, através de uma API, para formar uma federação. O projeto usa uma representação comum de recursos e um mecanismo semântico para gerenciar aplicativos em vários provedores. Buyya *et al.* investigam o uso de modelos de utilidade baseados no mercado para negociar o uso de recursos em vários provedores.

Por se tratar de uma ferramenta *open-source*, de alta grau de reuso e possibilidade de mudança nos seus módulos, além de trabalhar muito bem com *workflows* de Bioinformática, para esta dissertação foi escolhido como estudo de caso a plataforma BioNimbuZ, que é uma plataforma que oferece interfaces web de fácil utilização e eficiência na execução de ferramentas que extensivamente usam recursos de memória e de armazenamento.

2.3 *Workflows* Científicos

O termo *workflow* tem sua origen na década de 1970, associadas aos processos de automação de escritórios. O objetivo era oferecer soluções para diminuir a geração e a distribuição de documentos em papel de uma organização. Neste contexto, um *workflow* pode ser entendido como a automação total ou parcial de um processo, na qual informações ou tarefas são passadas de uma entidade para outra, de acordo com um conjunto de regras [81].

Mais recentemente, o conceito de *workflow* vem sendo aplicado às ciências na automação de experimentos computacionais que necessitam de grande poder de processamento

e manipulam grande quantidade de dados, possivelmente, distribuídos. Nesse contexto, um *workflow* pode ser definido como um grafo direcionado e acíclico (DAG, do inglês *Directed Acyclic Graph*), no qual os vértices e as arestas representam as atividades e suas dependências, respectivamente [87].

Assim sendo, *workflows* de Bioinformática estão relacionados ao processamento de dados biológicos, podendo ser usados, por exemplo, em processos de sequenciamento de DNA (Ácido Desoxirribonucleico) ou RNA (Ácido Ribonucleico). Estes processos se relacionam na descoberta de qual é a sequência de bases que forma cada fragmento de DNA ou RNA de um determinado organismo que está sendo investigado. A partir desses fragmentos, tanto de DNA quanto de RNA, vários processos computacionais podem ser executados de acordo com os objetivos do projeto.

De forma geral, projetos de Bioinformática possuem suporte computacional, nos quais são projetados *workflows* que transformam fragmentos de entrada, de forma a extrair informações como funções biológicas e localização dentro da célula. O exemplo na Figura 2.3 mostra um *workflow* de três fases: filtragem, mapeamento/montagem e análise, descritas em detalhes a seguir.

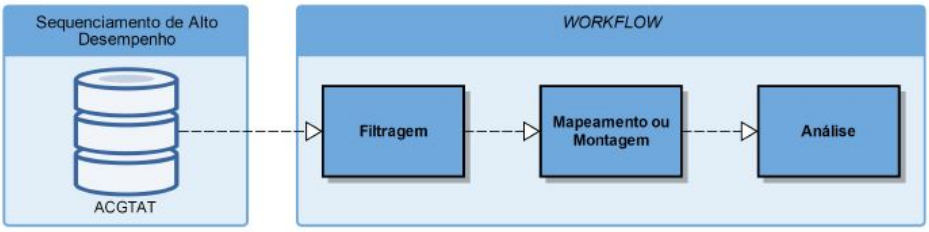

Figura 2.3: Exemplo de *Workflow* de Bioinformática, adaptado de [63].

Em um *workflow* para montagem de DNA, os biólogos inicialmente realizam processos laboratoriais de coleta e replicação do material biológico. Em seguida, é realizado o sequenciamento de pequenas porções de DNA ou de RNA (de acordo com o projeto), que são denominadas *reads*.

As *reads* coletadas passam para o processo de filtragem, que elimina as *reads* de qualidade inferior ou parte delas, com um certo grau de confiabilidade. Na fase de mapeamento (ou montagem), o objetivo é localizar em um genoma de referência o alinhamento para o maior número de *reads* filtradas. O término da fase de sequenciamento é realizado na fase de análise, na qual tem-se grandes porções do DNA ou RNA sequenciados, que são analisados por diferentes processos que dependem do objetivo do experimento [63].

Todo esse processo de execução de *workflows* de Bioinformática demanda expressivos recursos computacionais. Para isso, pode-se utilizar o poder computacional disponível na computação em nuvem.

2.4 Considerações Finais

Neste capítulo foram abordados os principais conceitos de computação em nuvem, dando ênfase às mais relevantes características, sua arquitetura e seus modelos de implantação. Outros conceitos abordados foram as principais características e classificações da computação em nuvem. Por fim, foram apresentadas as definições de federação de nuvens e de *workflows* científicos. No próximo capítulo serão apresentadas as principais características de elasticidade em nuvem.

Capítulo 3

Elasticidade em Nuvem

Este capítulo tem como objetivo fazer uma revisão bibliográfica acerca da elasticidade em computação em nuvem. Na Seção 3.1 são apresentadas as classificações de elasticidade. Em seguida, na Seção 3.2 é feita uma revisão sobre os principais trabalhos relacionados ao tema de elasticidade. Para finalizar, a seção 3.3 apresenta as principais considerações finais deste capítulo.

3.1 Elasticidade em Nuvem

Nos últimos anos a computação em nuvem tem atraído a atenção da indústria e do mundo acadêmico, tornando-se cada vez mais comum encontrar na literatura casos de adoção da nuvem por parte das empresas e instituições de pesquisa. Um dos principais motivos é a possibilidade de aquisição de recursos de uma forma dinâmica e elástica. De fato, a elasticidade é um grande diferencial e é, atualmente, vista como indispensável para o modelo de computação em nuvem [62].

O termo elasticidade pode ser definido como a capacidade de se adaptar às alterações na quantidade de recursos ou de solicitar/liberar recursos de acordo com a sua necessidade [40], além de um rápido provisionamento e desprovisionamento, com capacidade de recursos virtuais praticamente infinita a qualquer momento [53].

A elasticidade na computação em nuvem possui diversas características, e essas características vem sendo aperfeiçoadas em diversos trabalhos [17] [33] [58], que serão melhor descritos a seguir:

Coutinho *et al.* [17] propuseram uma classificação de elasticidade em dois grandes grupos, os métodos e os modelos. Os **métodos** são as ações que são tomadas para elasticidade e podem ser divididas em elasticidade horizontal, que consistem em adicionar/remover instâncias, e elasticidade vertical, que consiste em adicionar/remover recursos (processamento, memoria e disco) de uma máquina virtual, e por último a migração, que consiste

na transferência de uma máquina virtual que está rodando em um servidor físico para outro. Por outro lado, os modelos são divididos em, reativos, que reagem a alguma carga de trabalho atual; E proativos/preditivos que utilizam técnicas para prever a utilização futura e disparar a elasticidade antes da capacidade ser excedida, esta classificação pode ser observada na Figura 3.1.

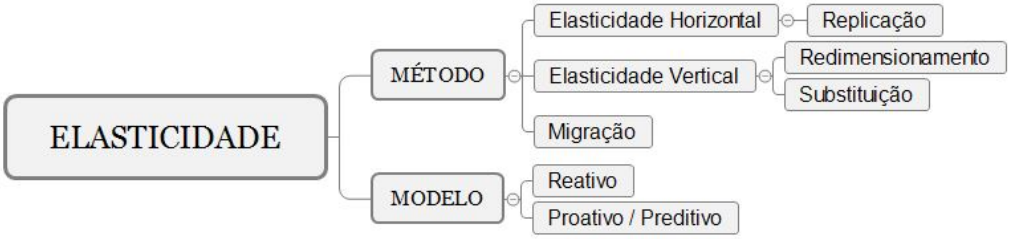

Figura 3.1: Classificação de Elasticidade, proposta em [17].

Já para Galante *et al.* a classificação é dividida em quatro características, como pode ser visto na Figura 3.2. A primeira delas, o **escopo**, define onde a elasticidade é controlada, se é pelos provedores de nuvem, ou pela própria aplicação do usuário.

A segunda característica é **Política**, que está relacionada à interação necessária para a execução da elasticidade, podendo ser dividida em manual, que é de responsabilidade do próprio usuário monitorar o ambiente e realizar todas as ações de elasticidade; e automático, na qual o controle e as ações de elasticidade são feitas automaticamente pelo controlador de acordo com regras e parâmetros especificados pelo SLA, podendo ser aplicadas pelo método reativo, o qual é baseado em regras pre definidas; ou pelo método preditivo, que utiliza heurísticas para antecipar a carga de trabalho.

A **Proposta** da elasticidade vem sendo usualmente utilizada para prevenir um provisionamento inadequado dos recursos, e consequentemente degradação da performance do sistema, porém outras propostas também tem sido estudadas, tais como, custo e energia.

Por ultimo, classifica-se a elasticidade quanto ao **Método** que é subdividido em 3 soluções. A Replicação consiste em adicionar/remover máquinas virtuais do ambiente, já o Redimensionamento consiste em adicionar/remover CPU e memória de uma máquina virtual em execução. E por último a Migração que transfere uma máquina virtual em execução de um *host* para outro.

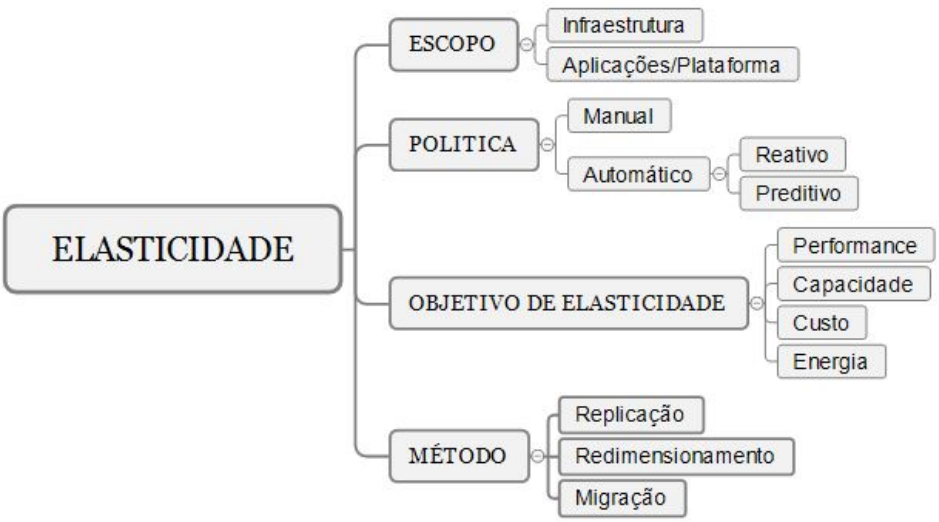

Figura 3.2: Classificações de Elasticidade, proposta por [33].

Athanasius *et al.* propuseram uma classificação em seis categorias, como pode ser visto na Figura 3.3. O primeiro deles, o **Escopo**, define o local no qual a elasticidade é feita, e esta dividido em outras duas classificações, que são: provedor, que indica se a técnica elástica tem que ser aplicada diretamente pelo provedor de infraestrutura, ou Aplicação, que indica que a elasticidade deve ser feita pela própria aplicação é de um tipo particular de aplicação em nuvem, por exemplo banco de dados.

Já a categoria **Proposta**, classifica qual a proposta da elasticidade, que podem ser de Performance, Disponibilidade, Custo e Energia. A categoria **Tomada de Decisão**, refere-se ao mecanismo de decisão adotado para disparar a elasticidade, e está dividido em quatro diferentes categorias, Gatilho, o qual indica se a elasticidade vai ser disparada de maneira reativa ou proativa; Mecanismo, refere-se a alguma metodologia de decisão; Modelo de Predição, utiliza algum modelo para predizer futuras variações de carga; e Modelo de Sistema, que refere-se a utilização de um modelo para representar o comportamento da elasticidade no sistema.

A categoria **Ação Elástica** define a ações de elasticidade, que podem ser aplicadas de diferentes formas. elasticidade horizontal que é a adição do número de máquinas virtuais, Migração em tempo real, é a migração de uma máquina virtual de um *host* para outro, ou elasticidade vertical que é a alocação de mais memória ou CPU para um máquina virtual.

As ações de elasticidade podem ainda ter outros dois tipos, a Reconfiguração da aplicação, que é quando a ferramenta de elasticidade é capaz de manipular aspectos de uma

ferramenta especifica, como tamanho da cache do banco de dados; e migração em tempo real de aplicação, quando apenas alguns componentes de uma aplicação são migrados, ao invés de uma máquina virtual inteira.

A elasticidade pode categorizar-se também pelo **Provedor**. Essa classificação está relacionada ao número de provedores na infraestrutura, podendo ser um Único, que denota que somente um provedor de nuvem é suportado; Único*, quando mais de um provedor é suportado, porém não ao mesmo tempo; e múltiplos, na qual o controle de elasticidade é feito através de todos os provedores de nuvem ao mesmo tempo.

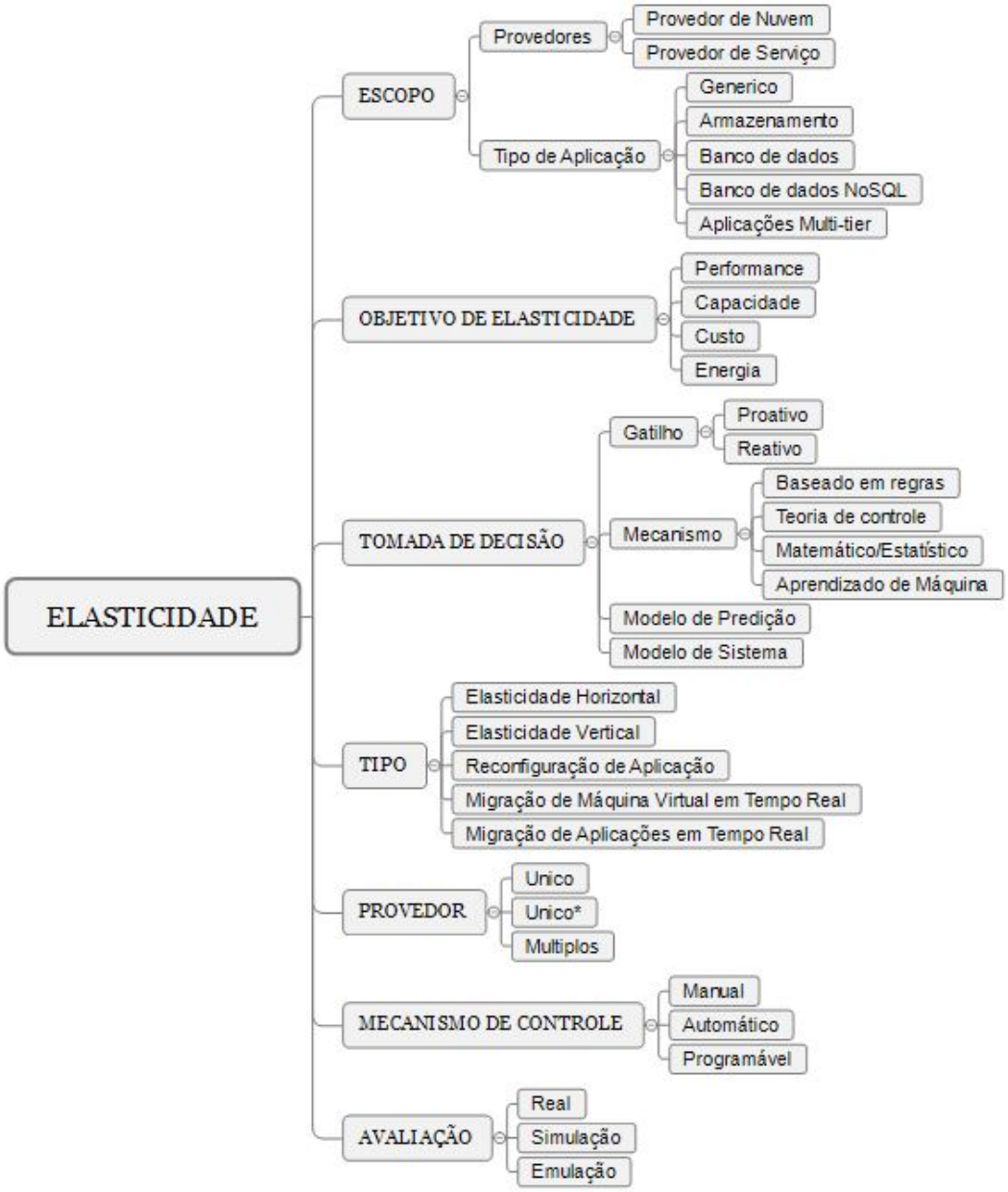

Figura 3.3: Classificações de Elasticidade, proposto por [58].

Finalmente a **Avaliação**, que refere-se a qual tipo de avaliação da elasticidade é feita, podendo ser, Simulação, na qual os resultados são obtidos baseados em simulações; Emulação na qual os resultados são obtidos através de rum ambiente artificial que se comporta de acordo com o mundo real; e Real, na qual os testes do mecanismo elástico são feitos em um ambiente real.

Porém, nenhum desses trabalhos abrange todas as características definidas ao longo dos anos. Assim, este trabalho propõe uma integração desses conceitos de elasticidade em nuvem, conforme pode ser visto na Figura 3.4. Na integração proposta, a elasticidade foi dividida em nove grupos, os quais são descritos nas próximas seções.

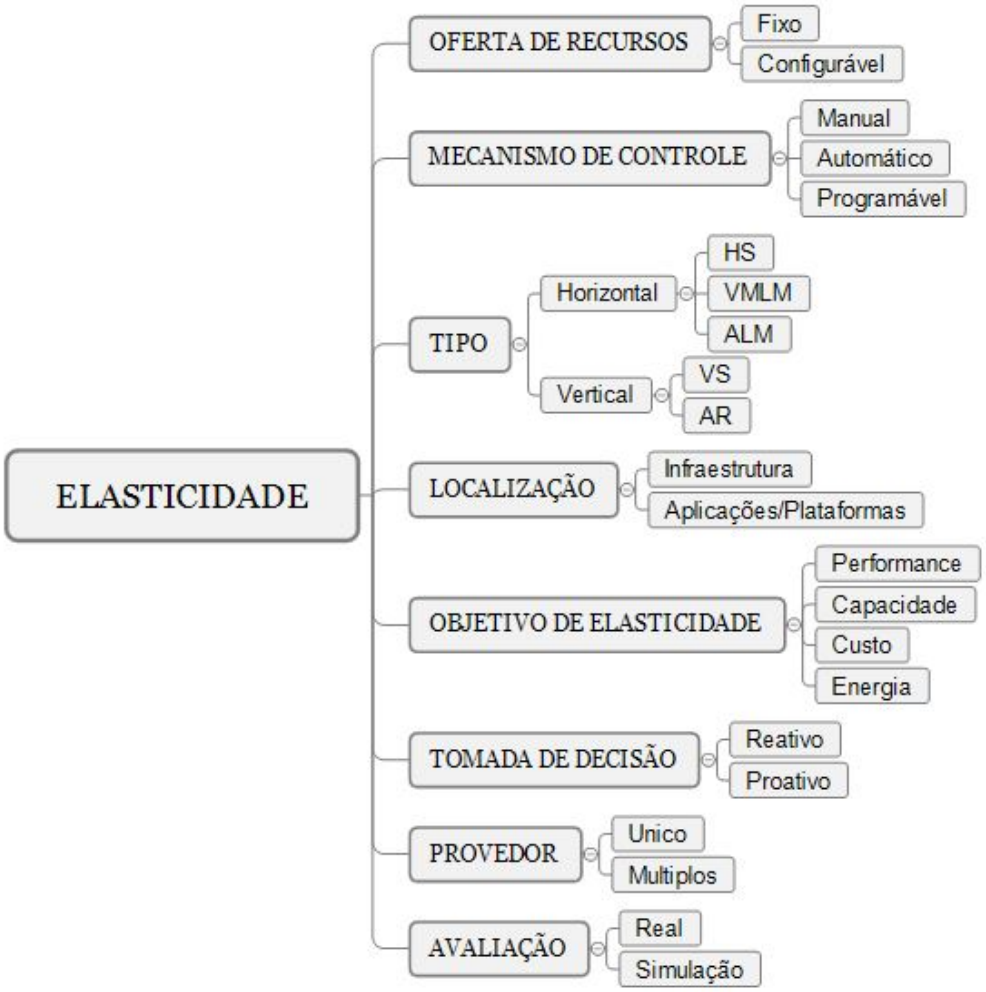

Figura 3.4: Integração entre as Classificações de Elasticidade Propostas na Literatura.

3.1.1 Elasticidade Quanto a Oferta de Recursos

Com relação à oferta de recursos, as nuvens de IaaS podem oferecer recursos de modo fixo ou configurável. No modo fixo, as VMs são oferecidas como um conjunto pré-definido de CPU, memória e E/S, chamados, por exemplo, pela Amazon [50] de tipos de instância, e pela GoGrid [34] e Rackspace [67] de tamanho de servidor nos provedores. A oferta de conjuntos fixos pode ser um problema em casos em que o usuário possui uma necessidade específica que não pode ser mapeada em um dos conjuntos disponibilizados pelo provedor.

Por outro lado, no modo configurável, o usuário pode escolher os recursos de forma personalizada, de acordo com a sua necessidade. Embora seja mais adequado ao conceito de nuvem, o modo configurável está disponível em poucas nuvens de IaaS, tais como *Profitbricks* [66] e *CloudSigma* [15].

Para que se possa tirar o máximo proveito da elasticidade fornecida pela nuvem, não basta que os recursos virtualizados sejam elásticos. Também é necessário que as aplicações tenham capacidade de se adaptar ou serem adaptadas dinamicamente de acordo com alterações em seus recursos. Tais mecanismos podem ser classificados de acordo com o mecanismo de controle, quanto a localização e quanto ao tipo de mecanismo de elasticidade suportada [32].

3.1.2 Elasticidade Quanto ao Mecanismo de Controle

O mecanismo de controle pode ser feito de três maneiras: o controle do tipo manual, do tipo programável e o do tipo automático. O controle refere-se ao modo de interação necessário para a execução de ações de elasticidade. Se o sistema possui controle manual, significa que o usuário é o responsável por monitorar o seu ambiente virtual e suas aplicações, bem como executar todas as ações de elasticidade pertinentes. Neste caso, a interação usuário-nuvem é feita com o uso de uma interface.

No controle do tipo programável as ações de elasticidade são feitas por meio de chamadas a APIs disponibilizadas pelo provedor da nuvem. Geralmente, estas APIs estão disponíveis para linguagens voltadas à aplicações web, tais como Java [41], PHP [65], Ruby [71], entre outras.

No controle automático de elasticidade, o controle e as ações são tomadas por um controlador de elasticidade, de acordo com as regras e as configurações feitas pelo usuário, ou definidas pelo contrato de serviço (SLA), como ilustra a Figura 3.5. O controlador de elasticidade vale-se de informações sobre a carga de trabalho, uso de CPU e de memória, tráfego de rede, entre outros, para tomar decisões de quando e quanto escalar os recursos. Estas informações podem ser coletadas por um sistema de monitoramento ou pela própria aplicação.

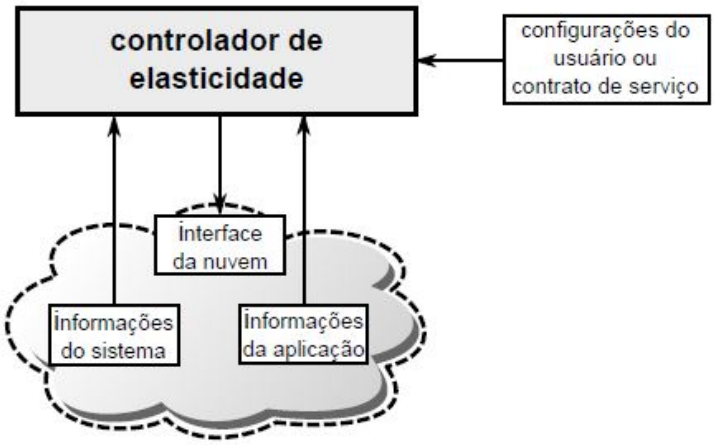

Figura 3.5: Controlador de Elasticidade, adaptado de Galante *et al.* [32].

3.1.3 Elasticidade Quanto ao Tipo Suportado

É possível analisar os mecanismos quanto ao suporte à elasticidade horizontal e vertical [83]. Se o mecanismo suporta apenas elasticidade horizontal, os recursos são alocados ou desalocados em termos de VMs completas. Neste caso, a aplicação deve ser implementada de modo a assimilar diferentes quantidades de VMs em seu ambiente de execução.

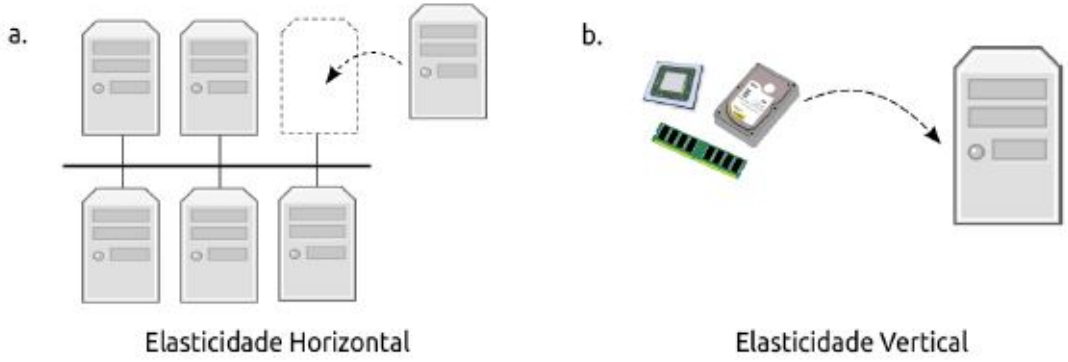

Figura 3.6: Elasticidade Horizontal(a) e Vertical(b).

Uma nuvem com elasticidade horizontal (veja Figura 3.6(a)) possibilita apenas a adição ou a remoção dinâmica de VMs da plataforma de computação alocada pelo usuário. Por outro lado, a elasticidade vertical (veja Figura 3.6(b)) é caracterizada pela possibilidade de se alterar a capacidade de VMs em execução. Tipicamente, a elasticidade vertical é implementada através da adição ou da remoção de CPUs e de memória, mas também pode ser utilizada no contexto de redes e de armazenamento. A implementação desses dois métodos pode ser feita da seguinte forma:

- **Escalabilidade Horizontal -** *Horizontal Scaling (HS):*

Também chamada de replicação, este método consiste em adicionar/remover instâncias do ambiente virtual de um usuário. Essas instâncias podem ser aplicativos, contêineres ou máquinas virtuais. A replicação é atualmente o método mais utilizado para fornecer elasticidade.

- **Escalabilidade Vertical -** *Vertical Scaling (VS):*

  Consiste em adicionar/remover recursos (por exemplo, unidade de processamento, memória e disco) de uma máquina virtual. Essa abordagem pode ser subdividida em outras duas.

  - *Redimensionamento*: Consiste em alterar, em tempo de execução, os recursos atribuídos para uma máquina virtual (por exemplo, atribuir mais CPUs físicas e memória a uma máquina virtual em execução). Essa solução é comum em sistemas operacionais baseados em Unix, uma vez que eles suportam mudanças *on-the-fly* (sem reinicialização) nas CPUs ou na memória disponíveis.
  - *Substituição*: Consiste em adicionar servidores mais poderosos para substituir os menos poderosos. Esta abordagem, geralmente, inicia uma réplica mais poderosa e para a menos poderosa. Ela é comum em provedores de nuvem pública.

- **Migração em Tempo Real de Máquinas Virtuais -** *VM Live Migration (VMLM):*

  Consiste em transferir uma máquina virtual que está sendo executado em um servidor físico para outro, em tempo de execução. As máquinas virtuais podem ser migradas de um servidor para outro, ou outra zona, dentro de uma mesma nuvem, para fins de consolidação ou balanceamento de carga.

- **Reconfiguração da Aplicação -** *Application Reconfiguration (AR):*

  Acontece quando a ferramenta elástica é capaz de lidar com aspectos específicos de aplicações, por exemplo, tamanho de cache de banco de dados;

- **Migração em Tempo Real de Aplicações -** *Application Live Migration (ALM):*

  Bem como na migração em tempo real de máquinas virtuais, este cenário consiste em transferir o que está sendo executado em um servidor físico para outro em tempo de execução, porém neste caso são migrados somente alguns componentes específicos, tal como instâncias do banco de dados.

3.1.4 Elasticidade Quanto a Localização

Esta classificação considera o local na qual as ações de elasticidade são tomadas, o qual podem ser nas infraestruturas dos provedores de nuvem, ou interno às aplicações ou plataformas.

Também chamados de mecanismos externos [32], os controles de elasticidade nas infraestruturas, geralmente, são implementados pelos provedores, o qual são responsáveis por converter os requisitos dos usuários em ações disponíveis pelas provedoras. O controlador monitora os dados das aplicações e toma decisões em quando os recursos devem ou não ser escalados [83]. Por exemplo, o trabalho apresentado em [38] depende de uma ferramenta que é instalada nas infraestruturas IaaS. Outras ferramentas necessitam de privilégios especiais aos recursos (por exemplo, o trabalho [59] depende de um módulo modificado do KVM, e a proposta [5] é integrado dentro do OpenStack), ou acessa a informação que apenas um provedor de nuvem é capaz de fornecer.

Por outro lado, a elasticidade pode ser controlada pelas aplicações em si, ou pelas suas plataformas, também chamados de mecanismos internos [32]. Nesse caso, o controlador de elasticidade é incorporado à aplicação, ou ao seu ambiente de execução. Neste modelo o controlador interage com a nuvem para poder requisitar ou liberar recursos, ou seja, são desenvolvidos especificamente para coletar informações particulares de uma aplicação, fornecendo uma solução sob-medida.

3.1.5 Elasticidade Quanto ao Objetivo

A elasticidade é essencial para o conceito de computação em nuvem, e vem sendo utilizada efetivamente para vários fins [23]. A elasticidade pode ter como objetivo aumentar a performance e/ou a capacidade, e para diminuir o custo financeiro ou energético.

O aumento de performance refere-se a manutenção, evitando a insuficiência de recursos e, consequentemente, a degradação do desempenho do sistema comumente causados por um provisionamento inadequado. Além disso, para garantir também tanto o SLA quanto o que foi especificado pelo usuário. Contudo, alguns estudos têm descrito o uso da elasticidade para outros fins, tais como, redução nos custos [76] e poupança de energia [77].

3.1.6 Elasticidade Quanto a Tomada de Decisão

Em relação à tomada de decisão a elasticidade pode ser dividida em reativa e proativa. A primeira refere-se as soluções reativas, esse tipo de solução emprega mecanismos de Regra-Condição-Ação. Nesse caso, as políticas de elasticidade são definidas por um conjunto de regras, e quando uma condição é satisfeita, uma ação é disparada [83]. Uma representação

desse modelo é mostrada na Figura 3.7. O uso de técnicas reativas é bastante comum, e é encontrado na maioria das soluções comerciais, tais como [48], [50] e [69].

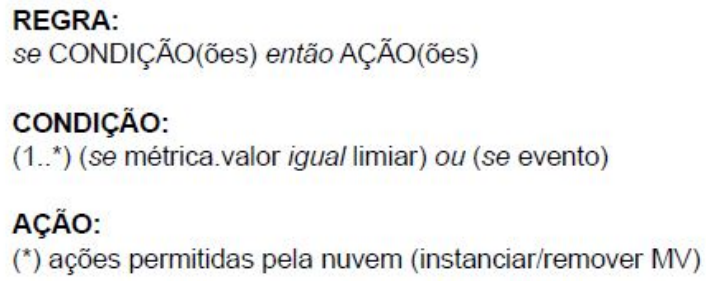

Figura 3.7: Mecanismo Regra-Condição-Ação, adaptado de Galante *et al.* [32].

Por outro lado, abordagens proativas usam heurísticas e técnicas matemáticas para antecipar o comportamento da carga do sistema, e com base nesses resultados, decide quando e como escalar os recursos. Segundo Moore *et al.* [56], a abordagem proativa é mais apropriada para os casos em que a carga de trabalho apresenta padrões bem definidos com periodicidade uniforme, facilitando a previsão de cargas futuras.

3.1.7 Elasticidade Quanto ao Provedor

Esta categoria de classificação refere-se ao número de provedores que a elasticidade suporta simultaneamente. É possível que seja um único provedor, quando somente um único provedor de nuvem é configurado para a elasticidade.

Por outro lado, pode ser realizado com múltiplos provedores, quando a infraestrutura provê elasticidade para mais de um provedor, porém, não simultaneamente. Ou também múltiplos provedores de nuvem ou uma federação de nuvens, simultaneamente, quando a infraestrutura consegue controlar a elasticidade em mais de um provedor de nuvem ao mesmo tempo.

3.1.8 Elasticidade Quanto a Avaliação

O último aspecto refere-se ao tipo de avaliação da elasticidade. As possíveis classificações são simulação e real. No primeiro os resultados obtidos são a partir de ambientes de computação artificial, ou seja, simuladores (por exemplo, CloudSim [9]). E real quando a ferramenta elástica é aplicada em uma infraestrutura de nuvem real.

3.2 Trabalhos Relacionados

Na literatura há vários trabalhos relacionados ao tema desta dissertação. Assim, algumas soluções são preditivas, propondo diversas estratégias para estimar com antecedência o comportamento das cargas de trabalho, outras reativas, que reagem as mudanças da carga de trabalho.

Como soluções reativas e horizontais é possível citar, Marshall *et al.* [52], Alessandro *et al.* [46], o Elastack [5] e o projeto RESERVOIR[12]. O trabalho desenvolvido por Marshall *et al.* [52] propôs um modelo elástico reativo capaz de responder às mudanças na demanda de recursos, buscando o melhor aproveitamento dos mesmos sobre um ambiente de nuvem. A solução foi desenvolvida para a plataforma Nimbus [42], utilizando o *Xen* [3] como virtualizador. Neste trabalho foram criadas três políticas de alocação de recursos, sendo que a elasticidade implementada por meio da replicação de máquinas virtuais.

Leite *et al.* [46] apresenta um método baseado em linha de produto de software para lidar com as variabilidades dos serviços oferecidos por nuvens de infraestrutura (IaaS). O método utiliza modelo de *feature* estendido com atributos para descrever os recursos e para selecioná-los com base nos objetivos dos usuários. A escolha das máquinas virtuais é feita utilizando um algoritmo de frente de pareto [46].

O Elastack [5] é um sistema genérico e pode ser aplicado a qualquer estrutura IaaS existente. Ele é destinado a permitir a elasticidade horizontal. Esta abordagem oferece a qualquer infraestrutura da nuvem os mecanismos para implementar monitoramento e adaptação automatizados, bem como, flexibilidade. O Elastack foi integrado e testado com o OpenStack [75] mostrando como adicionar esses recursos importantes com um mínimo de modificações na instalação padrão.

O projeto RESERVOIR [12], também implementa um mecanismo reativo. A elasticidade é especificada explicitamente adicionando regras de elasticidade ao *Service Manifest*, que é um descritor do serviço baseado no padrão *Open Virtualization Format (OVF)*. As regras especificam condições, que com base em eventos de monitoramento na camada de aplicação ou de outra forma, o que leva a executar as ações especificadas disponíveis na plataforma OpenNebula [55].

Calheiros *et al.* [10] propuseram uma técnica de provisionamento de máquina virtual para se adaptar dinamicamente às mudanças na carga de trabalho, e fornecer garantias de qualidade de serviço (do inglês, *Quality of Service* - QoS) para os usuários finais. Sua solução é o modelo do sistema de filas e informações de carga de trabalho para direcionar as decisões de replicação do mecanismo provisionado. Os autores apresentaram vários experimentos baseados em simulação de carga de trabalho de produção, e os resultados indicaram que a técnica de provisionamento proposto pode detectar mudanças na inten-

sidade da carga de trabalho que ocorrem ao longo do tempo, e pode instanciar réplicas adequadamente para alcançar as metas de QoS de aplicação.

Dawoud *et al.* [20] apresentaram uma arquitetura elástica para o gerenciamento dinâmico de recursos e otimização de aplicações em ambiente virtualizado, utilizando o Xen [3]. O trabalho implementou três atuadores de CPU, Memória e Aplicação. Esses atuadores executados em paralelo garantem a eficiente alocação de recursos e a otimização do desempenho da aplicação, reorganizando as máquinas virtuais dinamicamente sobre os recursos disponíveis.

A questão de custo é abordada por Sharma *et al.* [76]. O objetivo do sistema é tentar minimizar o custo de implantação das máquinas na nuvem, ao mesmo tempo que faz elasticidade para combater às mudanças na carga de trabalho. Esse trabalho leva em conta o custo de cada instância de máquina virtual, as possibilidades de migrar ou replicar a máquina virtual, bem como o tempo de transição de uma configuração para outra.

Konstanteli *et al.* [43] apresentaram uma abordagem probabilística para o problema de alocação ótima de serviços em recursos físicos virtualizados, quando a elasticidade horizontal é necessária. O modelo de otimização formulado constitui um teste probabilístico. Com base no conhecimento estatístico, esta abordagem utiliza modelos de previsão capazes de prever os usos de recursos com base em dados históricos de monitoramento.

Tabela 3.1: Trabalhos Relacionados ao tema Elasticidade.

Paper	**Tipo de elasticidade**	**Localização**	**Mecanismo de Controle**	**Objetivo de Elasticidade**	**Avaliação**	**Provedor**	**Ação de Elasticidade**	**Tomada de Decisão**	**Oferta de Recursos**
Este Trabalho	Horizontal e Vertical	Plataforma	Automático e Manual	Performance e custo	Real	Federação	Elasticidade Horizontal e Elasticidade Vertical	Reativo	Fixo e Configurável
RESERVOIR [12]	Horizontal	Infraestrutura	Automático	Performance	Simulação	Único	Elasticidade Horizontal	Reativo	Fixo
Leite *et al.* [46]	Horizontal	Plataforma	Automático	Performance e Custo	Real	Múltiplos	Elasticidade Horizontal	Reativo	Fixo
Marshall *et al.* [52]	Horizontal	Infraestrutura	Automático	Performance	Simulação	Único	Elasticidade Horizontal	Reativo	Fixo
Elastack [5]	Horizontal	Infraestrutura	Automático	Performance	Real	Único	Elasticidade Horizontal	Reativo	Fixo
Calheiros *et al.* [10]	Horizontal	Infraestrutura	Automático	Performance	Simulação	Único	Elasticidade Horizontal	Proativo	Fixo
Dawoud *et al.* [20]	Vertical	Infraestrutura	Automático	Performance	Real	Único	Elasticidade Vertical	Reativo	Fixo
Sharma *et al* [76].	Vertical	Plataforma	Automático	Custo	Real	Único	Elasticidade Vertical	Reativo	Fixo
Konstanteli *et al.* [43]	Vertical	Infraestrutura	Automático	Performance	Real	Único	Elasticidade Vertical	Proativo	Fixo

Assim, nota-se na tabela 3.1 que existem trabalhos na literatura que tratam do gerenciamento da elasticidade, porém, nenhum com a abordagem deste trabalho. O mecanismo proposto se diferencia de todos os outros, por oferecer um controlador de elasticidade tanto horizontal quanto vertical, em um ambiente de nuvem federada. Logo, o controlador de elasticidade proposto neste trabalho é apresentado em detalhes no Capítulo 5.

3.3 Considerações Finais

Neste capítulo foram apresentadas as definições de elasticidade, a classificação proposta e o estado da arte. O próximo capítulo tem como objetivo apresentar a plataforma de federação de nuvens computacionais BioNimbuZ, apresentando sua arquitetura e seus serviços.

Capítulo 4

BioNimbuZ

Este capítulo tem como objetivo apresentar a plataforma de federação de nuvens computacionais BioNimbuZ, que foi a plataforma escolhida para testar o controlador de elasticidade proposto neste trabalho. Assim, na Seção 4.1 é apresentada uma visão geral desta plataforma. Na Seção 4.2 são apresentadas em detalhes sua arquitetura e os seus serviços. E na Seção 4.3 as considerações finais deste Capítulo.

4.1 Visão Geral

O BioNimbuZ é uma plataforma para federação de nuvens que foi proposta originalmente por Saldanha [73], e que tem sido aprimorada constantemente em outros trabalhos [1] [4] [16] [49] [57] [68] [74]. O BioNimbuZ foi desenvolvido para suprir a demanda de plataformas de nuvens federadas, visto que a utilização de nuvens de forma isolada já não atende, em muito casos, às necessidades de processamento, de armazenamento, entre outros, na execução das aplicações de Bioinformática.

O BioNimbuZ permite a integração entre nuvens de diversos tipos, tanto privadas quanto públicas, deixando que cada provedor mantenha suas características e políticas internas, e oferece ao usuário transparência e ilusão de infinidade de recursos. Desta forma, o usuário pode usufruir de diversos serviços sem se preocupar com qual provedor está sendo de fato utilizado.

Outra característica desta plataforma é a flexibilidade na inclusão de novos provedores, pois são utilizados *plugins* de integração que se encarregam de mapear as requisições vindas da arquitetura para as requisições de cada provedor especificamente. Isso é fundamental para que alguns objetivos possam ser alcançados, principalmente, na questão da escalabilidade e da flexibilidade.

Originalmente, toda a comunicação existente no BioNimbuZ era realizada por meio de uma rede *Peer-to-Peer* (P2P) [79]. Porém, para alcançar os objetivos desejados de

escalabilidade e de flexibilidade, percebeu-se a necessidade de alterar a forma de comunicação entre os componentes da arquitetura do BioNimbuZ, pois a utilização de uma rede de comunicação *Peer-to-Peer* (P2P) não estava mais suprindo as necessidades nestes dois quesitos.

É importante ressaltar que os outros objetivos inicialmente propostos por Saldanha [73], tais como obter uma arquitetura tolerante a falhas, com grande poder de processamento e de armazenamento, e que suportasse diversos provedores de infraestrutura, foram mantidos e melhorados.

Assim sendo, no trabalho [49] foi proposta a utilização de Chamada de Procedimento Remoto (RPC) [45], para realizar a comunicação de forma transparente, pois ela permite a chamada de procedimentos que estão localizados em outras máquinas, sem que o usuário perceba [82].

Para auxiliar na organização e na coordenação do BioNimbuZ, foi utilizado um serviço voltado à sistemas distribuídos chamado Apache ZooKeeper [31], e o protocolo SFTP, para transferências dos arquivos. Além disso, foi implementada uma política de armazenamento que considera a latência e o local em que o serviço será executado.

Após essa evolução, Azevedo e Freitas [1], melhoraram a política de armazenamento, à qual passou a considerar a quebra de arquivos e o cálculo da largura de banda entre o cliente/servidor em sua decisão de compactação de arquivos e de transferência de arquivos.

Entretanto, a aplicação ainda não possuía uma interface gráfica amigável, com o intuito de melhorar a interação cliente/servidor, Ramos [68] implementou uma interface gráfica utilizando *primefaces* e um controlador de *jobs*, responsável por fazer a ligação entre a Camada de Interface com o usuário e o Núcleo do BioNimbuZ.

Com o objetivo de melhor distribuir as tarefas da plataforma, Barreiros Júnior [4] implementou um novo escalonador, chamado de C99, que se baseia no algoritmo de busca combinacional *beam search*, levando em consideração o custo por hora dos recursos a serem alocados para o BioNimbuZ.

Com a necessidade de melhorar o armazenamento dos arquivos e aproveitar o aumento na variedade e na qualidade dos serviços oferecidos pelos provedores de nuvem, Santos [74] modificou o serviço de armazenamento, possibilitando o serviço de armazenamento em nuvem ofertados pelos provedores de nuvem.

Com a implementação da interface gráfica, o novo escalonador e a melhoria no serviço de armazenamento, a plataforma sofreu algumas mudanças, resultando na arquitetura, que será apresentada na Seção 4.2.

4.2 Arquitetura do BioNimbuZ

O BioNimbuZ faz uso de uma arquitetura hierárquica distribuída, conforme apresentada na Figura 4.1 que foi proposta em [68]. Ela representa a interação entre as quatro camadas principais: Aplicação, Comunicação, Núcleo e Infraestrutura.

A primeira camada, a Camada de Aplicação, permite a integração entre a aplicação do usuário e os serviços do núcleo da plataforma. A interface com o usuário pode ocorrer tanto através de linha de comando ou por uma interface gráfica, e é nela que devem ser inseridos os *workflows* que serão executados nas diversas nuvens.

A Camada de Aplicação comunica-se com a Camada de Núcleo através da Camada de Integração, que é a responsável pela troca de mensagens entre a Camada de Núcleo e a Aplicação *web* utilizando *webservices*, que disparam requisições da aplicação *web* para o Núcleo, e recebe respostas do Núcleo para a Aplicação.

A Camada de Núcleo é responsável por realizar toda a gerência da plataforma e seus serviços, bem como se comunicar com os *plugins* de cada provedor. E por último, tem-se a Camada de Infraestrutura, que é composta pelos *plugins* utilizados pelo BioNimbuZ e que compõem sua federação de nuvens. Nas próximas Seções serão descritos o funcionamento e as características de cada uma destas camadas.

4.2.1 Camada de Aplicação

A Camada de Aplicação é a responsável por fazer toda a interação com o usuário. É nesta camada em que os usuários podem fazer *log-in* no sistema BioNimbuZ, por meio de uma interface gráfica acessível que pode ser acessada via Internet. Após o *log-in* na aplicação o usuário acessa a tela inicial do BioNimbuZ, como pode ser vista na Figura 4.2.

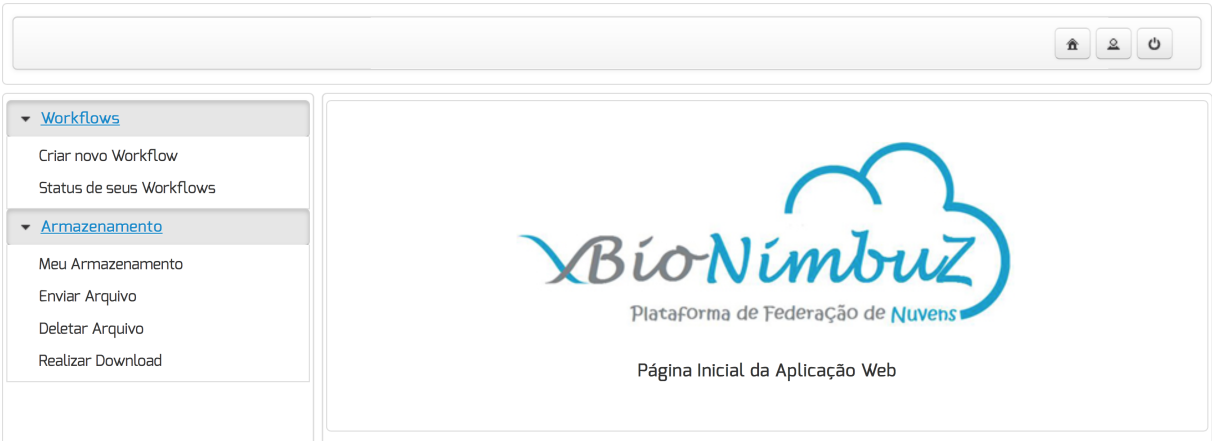

Figura 4.2: Tela Inicial da Aplicação *Web* do BioNimbuZ [68].

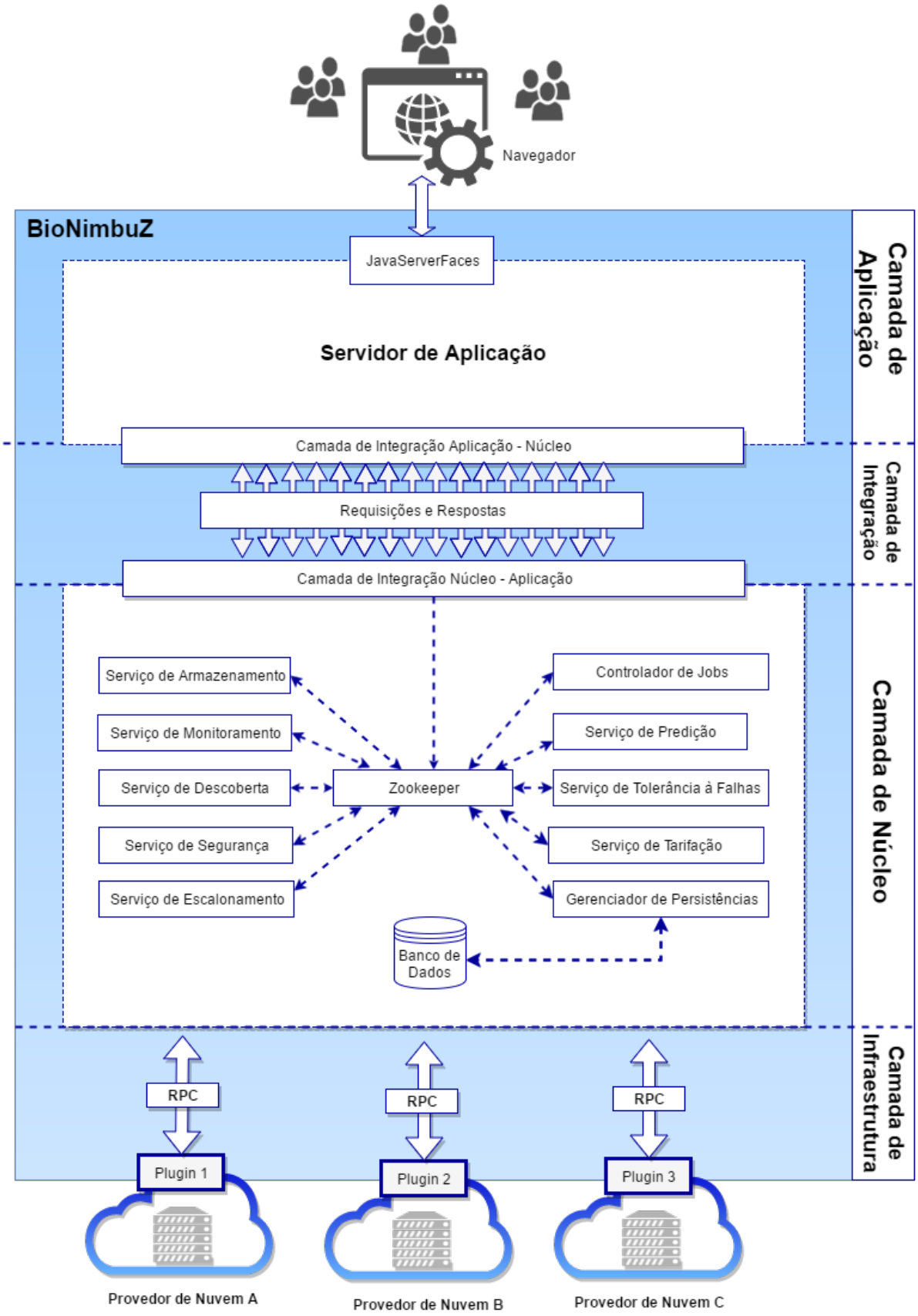

Figura 4.1: Arquitetura da Plataforma BioNimbuZ, adaptado de [68].

Pela interface do BioNimbuZ, o usuário é capaz de criar e projetar seus *workflows* de maneira gráfica, ligando passos, indicando dependências, incluindo argumentos e indicando quais arquivos de entrada serão utilizados, como pode ser visto na Figura 4.3.

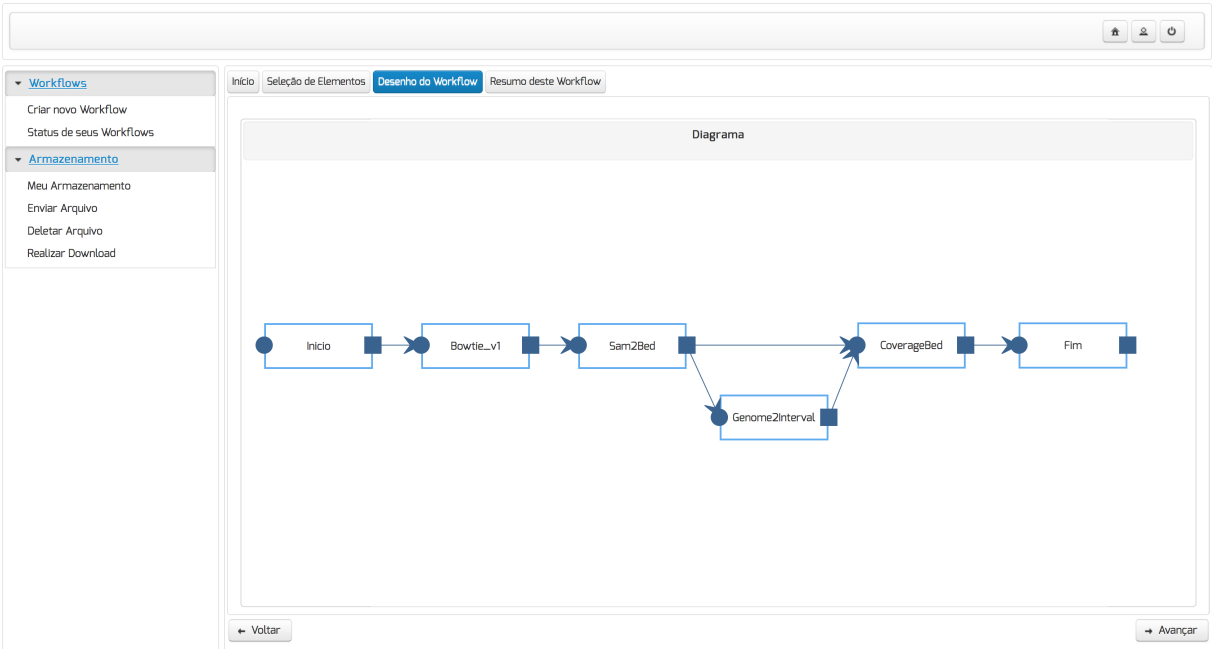

Figura 4.3: Tela de Montagem de Fluxo do *Workflow* da Aplicação [68].

Assim que o *workflow* for executado, é função dessa camada também apresentar o *feedback* da execução, mostrando para o usuário se o *workflow* foi completado com sucesso ou não, como pode ser visto na Figura 4.4.

E por último, outra funcionalidade disponível nesta camada é o envio de arquivos, que serão utilizados como entrada dos *workflows* criados pelo usuário, bem como a exclusão desses arquivos, como pode ser visto na Figura 4.5.

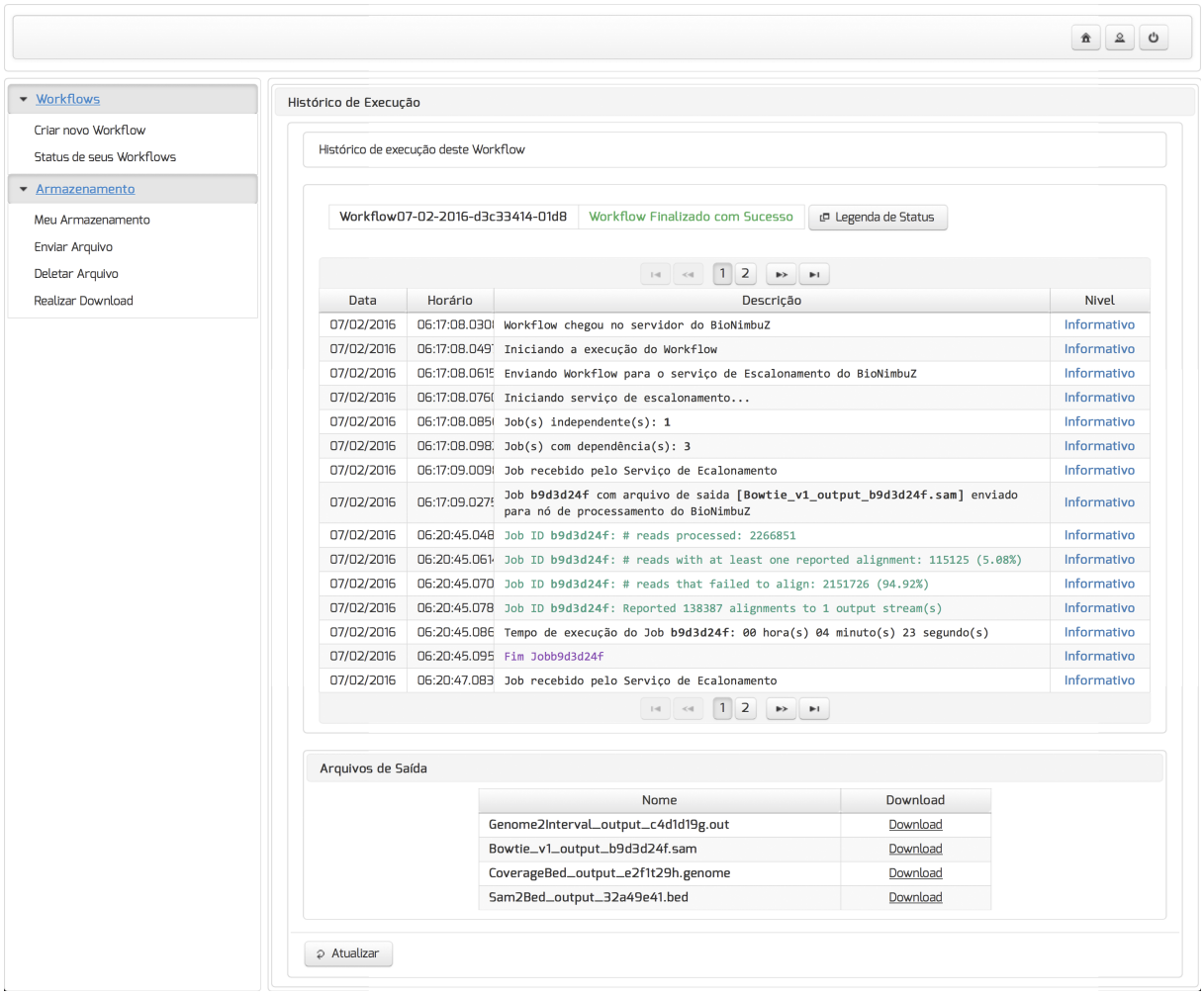

Figura 4.4: Tela de Monitoramento da Execução do *Workflow* [68].

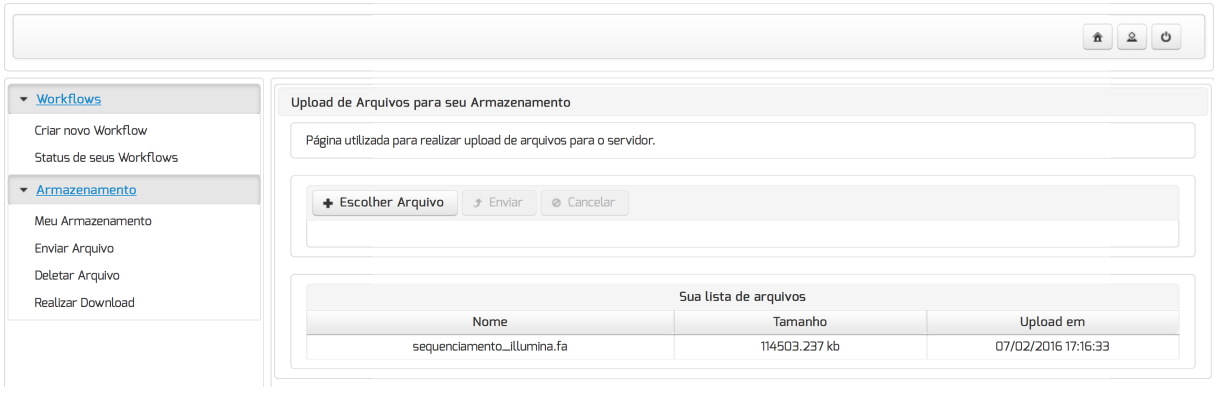

Figura 4.5: Tela de *Upload* de Arquivos [68].

4.2.2 Camada de Comunicação

A Camada de Comunicação é a responsável por toda a troca de mensagens entre a aplicação *web* e o Núcleo. Essa camada utiliza uma comunicação via *webservices REST* (*REpresentational State Transfer*) [28]. O *REST* é voltado para sistemas baseados na Internet e tem sido amplamente utilizado na integração de sistemas, pois utiliza operações definidas no protocolo *HTTP*, tais como *PUT*, *GET* e *DELETE*.

Dessa forma, para possibilitar a troca de mensagens entre a Camada de Aplicação e a Camada de Núcleo do BioNimbuZ, existem três entidades principais: Requisições (*requests*) que contém todos os dados necessários para a execução das ações requisitadas pela aplicação *web*; Respostas (*responses*) que definem todas as mensagens devolvidas pela Camada de Núcleo do BioNimbuZ para uma dada ação; E, por último, as Ações (*actions*) que definem o comando à ser executado pelo núcleo, enviando-lhe uma requisição, a fim de se obter uma resposta com os dados requeridos.

4.2.3 Camada de Núcleo

O Núcleo do BioNimbuZ (ou Camada de Núcleo) é o responsável por toda a gerência da federação, e possui oito serviços principais: serviço de descoberta, serviço de monitoramento, serviço de tolerância a falhas, serviço de escalonamento, serviço de segurança, serviço de armazenamento, serviço de predição e serviço de tarifação. Além desses, há dois controladores, que são: controladores de *Jobs* e de SLA. Todavia, novos serviços podem ser incorporados à medida em que forem demandados.

A fim de exercerem suas funcionalidades, os serviços interagem entre si por meio do gerenciamento de troca de mensagens provido pelo ZooKeeper. A seguir são descritas as principais funções de cada elemento da Camada Núcleo:

- **Controlador de *Jobs***: É o que recebe do usuário a requisição da Camada de Aplicação e faz a ligação com o núcleo do BioNimbuZ, verificando as credenciais dos usuários com o serviço de segurança, para permitir ou não a execução das tarefas. É responsável por gerenciar os vários pedidos e garantir que os resultados sejam entregues de forma correta para cada uma das requisições, e mantê-los para que possam ser consultados posteriormente;
- **Controlador de SLA:** Para toda tarefa submetida à 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, que o usuário deseja durante sua execução na plataforma da federação. Isso tudo ocorre para que haja um Acordo de Nível de Serviço (SLA) estabelecido previamente entre o usuário e a federação. Caso esse acordo seja violado, haverá 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.

O controlador de SLA é responsável por implementar o ciclo de vida de SLA, o qual compreende seis atividades: descoberta de provedores de serviço, definição de SLA, estabelecimento do acordo de serviço, monitoramento de violação do acordo, término de acordo e aplicação de penalidades por violação, como pode ser visto na Figura 4.6;

Figura 4.6: Ciclo de Vida do Controlador de SLA.

- **Serviço de Monitoramento:** Responsável por realizar o acompanhamento dos recursos da federação, junto ao Zookeeper, com a utilização de *watchers* e tratando os alertas enviados pelos mesmos. Uma outra responsabilidade deste serviço é permitir a recuperação dos dados principais utilizados pelos módulos de cada servidor BioNimbuZ, armazenados na estrutura do Zookeeper, para possíveis reconstruções ou garantias de execuções de serviços solicitados;
- **Serviço de Descobrimento:** É o serviço que identifica e mantém informações a respeito dos provedores de nuvem que estão na federação, tais como capacidade de armazenamento, processamento e latência de rede. Além disso, também mantém detalhes sobre parâmetros de execução e arquivos de entrada e de saída. Para obter

estas informações a respeito dos provedores, o Zookeeper é consultado, pois todos os provedores presentes na federação possuem *znodes* que armazenam seus dados. Assim, sempre que uma modificação é realizada, como a saída ou a entrada de algum provedor, os *watchers* alertam os outros serviços, mantendo assim todos os participantes da federação atualizados;

- **Serviço de Escalonamento:** É o serviço que recebe o pedido para a execução dos *jobs*, dividindo-os em unidades menores, chamadas de *tasks*, e distribui essas *tasks* entre os provedores selecionados. O serviço de escalonamento também é responsável por acompanhar toda a execução do *job*, e manter um registro das execuções já escalonadas.

  Para realizar a distribuição de tarefas na federação, algumas métricas são levadas em consideração, tais como latência, balanceamento de carga, tempo de espera e capacidade de processamento, entre outras, visando atender o que foi determinado no acordo de SLA. Atualmente, o BioNimbuZ tem implementado três políticas de escalonamento, cada uma focando em um objetivo diferente [4] [7] [61]. O mais recente deles, e que vem sendo utilizado atualmente, é o algoritmo C99 [4]. Esse algoritmo tem por objetivo ser *any-time* e incremental, com uma rápida convergência para boas soluções. Por incremental e *any-time*, entende-se que o algoritmo convergirá para um melhor conjunto de soluções com o decorrer de sua execução, e que o algoritmo poderá ser parado em qualquer momento da sua execução, retornando ainda um conjunto de boas soluções;

- **Serviço de Armazenamento:** Responsável pela estratégia de armazenamento dos arquivos que são utilizados ou mantidos pelas aplicações. O armazenamento deve ocorrer de forma eficiente para que as aplicações possam utilizar os arquivos com o menor custo possível. Este custo é calculado utilizando-se algumas métricas, tais como latência de rede, distância entre os provedores e capacidade de armazenamento. O Serviço de Armazenamento foi, inicialmente, introduzido com o trabalho de Bacelar e Moura [57], e logo após aprimorada por Azevedo e Freitas [1].

  Em seguida com a necessidade de melhorar o modo de armazenar e de dispor os arquivos, Santos [74] adicionou ao Serviço de Armazenamento um módulo que permite dispor e armazenar os arquivos da aplicação nos Serviços de Armazenamento em nuvem, possibilitando que esse serviço tenha dois modos de armazenamento. O primeiro modo é o armazenamento nas próprias máquinas virtuais que estão executando o *workflow*, já o segundo modo utiliza-se da tecnologia dos serviços de armazenamento por objeto [24], oferecidos pelos provedores externos utilizados na federação, mais especificamente o Amazon S3 [51] e o Google *Cloud Storage* [35]. O

armazenamento dos arquivos é realizado em *buckets*, que são volumes nos quais os arquivos são armazenados. Desta forma, requisições internas de arquivo são atendidas através da transferência do arquivo solicitado entre um dos *bucket* contendo o arquivo, e a máquina virtual realizando a requisição;

- **Serviço de Tolerância à Falhas:** O Serviço de Tolerância à Falhas tem como objetivo principal garantir que todos os serviços do BioNimbuZ estejam sempre disponíveis e, em caso de falhas, inicie alguma ação de recuperação. O serviço de tolerância a falhas deve atuar de forma distribuída na plataforma, e estar presente em outros serviços, monitorando seu estado;

- **Serviço de Segurança:** Segurança em nuvem federada é uma área de estudo em constante evolução, e o serviço de segurança deve trabalhar em diversos pontos para fornecer um serviço efetivamente seguro. O primeiro passo é a autenticação de usuários, ou seja, é preciso saber se o usuário que está tentando acessar algum recurso na federação é quem ele realmente diz ser. Depois da autenticação, vem a autorização, que consiste em verificar se o usuário pode realizar as ações que deseja. Muitos outros aspectos podem ser usados por este serviço, como a criptografia de mensagens, para garantir a confidencialidade na troca de informações entre provedores, e também a verificação de integridade de arquivos, de modo que seja possível garantir que um arquivo não seja alterado por fatores externos à federação. O trabalho do Moraes [16] implementou alguns níveis de segurança para o BioNimbuZ;

- **Serviço de Tarifação:** Tem a responsabilidade de calcular o quanto os usuários devem pagar pela utilização dos serviços oferecidos na plataforma BioNimbuZ. Para que isso seja possível, este serviço se mantém em constante contato com o Serviço de Monitoramento para obter informações, tais como tempo de execução e quantidade de máquinas virtuais alocadas para as tarefas que foram e que estão sendo executadas pelo usuário. Em um ambiente de nuvem federada, a complexidade de tal serviço é maior do que em um ambiente de nuvens computacionais, uma vez que a infraestrutura de uma plataforma de nuvens federadas é garantida pela integração de recursos de diversas nuvens autônomas.

  Assim, é função do serviço de tarifação, quando se trata de nuvens federadas, garantir o cumprimento das métricas de tarifação das nuvens independentes, definidas nos contratos assinados entre as nuvens provedoras e a plataforma, e realizar a cobrança baseada na demanda do cliente, de maneira que não haja percepção por parte do cliente e de que a plataforma obtenha recursos de diversas nuvens de forma autônoma. Diante do exposto, Sluzala[78] definiiu uma arquitetura para o modelo de tarifação do BioNimbuZ, mostrada na Figura 4.7.

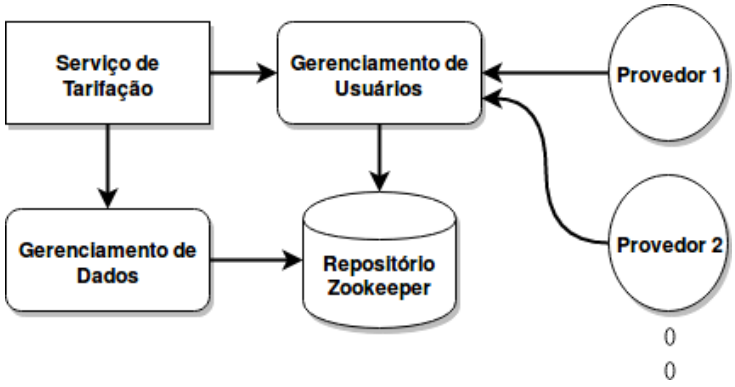

Figura 4.7: Arquitetura do Serviço de Tarifação

Na arquitetura apresentada na Figura 4.7, o componente Gerenciamento de Dados é o responsável pela obtenção das métricas de tarifação das nuvens que compõem a Camada de Infraestrutura do BioNimbuZ. O componente Gerenciamento de Usuário é o componente responsável por cumprir as métricas de tarifação definidas nos contratos firmados com as nuvens fornecedoras de recurso, e realizar a cobrança dos clientes baseada na demanda de suas aplicações. Por fim, o Zookeeper é utilizado como mecanismo de persistência, para que os dados adquiridos pelos componentes Gerenciamento de Dados e Gerenciamento de Usuário, estejam disponíveis para que os outros serviços possam usá-lo.

- Serviço de Predição: Qualquer *workflow* submetido à federação possui custos computacionais e financeiros. Desta forma, o BioNimBuZ possui o Serviço de Predição de Custos e Recursos Computacionais - sPCR, com o objetivo de auxiliar o usuário na escolha de um ambiente computacional que execute seus *workflows* científicos de forma transparente, com estimativas de tempo, custo financeiro e recursos a serem utilizados. O usuário preenche um *template* com as limitações de custo financeiro e de tempo de execução, para que o serviço estime o custo, o tempo e os melhores recursos computacionais que se adequem as variáveis informadas. A Figura 4.8 define a estrutura e o funcionamento do serviço de predição, que é realizado em quatro fases.

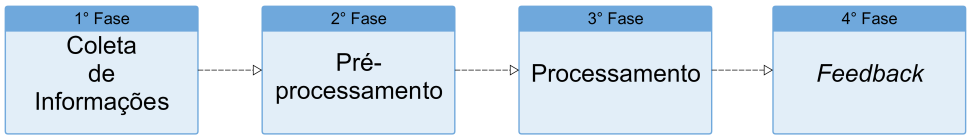

Figura 4.8: Fases de Execução do sPCR.

A primeira fase consiste na coleta das informações do *workflow* a ser executado, estas informações são fornecidas por meio da interface com o usuário, no qual o mesmo insere a estrutura do *workflow* a ser executado, juntamente com as restrições de custo financeiro, tempo máximo de duração da execução, definição entre uma execução de baixo custo ou de alto desempenho, além de definir parâmetros de recursos computacionais, tais como capacidade de memória RAM, disco e poder de processamento. Estas informações irão especificar a lista de recursos compatíveis com as restrições impostas.

A segunda fase é composta pela base de dados histórica e pela comunicação com o serviço de tarifação do ambiente de nuvem. A base de dados histórica irá auxiliar na acurácia da escolha dos recursos, na estimativa do tempo de execução e do custo financeiro. A base é composta por informações de execuções passadas, que são armazenados detalhes do *workflow*, tais como softwares utilizados, arquivos envolvidos, dimensão dos dados, tempo de execução de cada fase, consumo computacional de cada fase, entre outros.

Logo, é possível utilizar algoritmos que avaliem a base histórica a procura de execuções similares, para aprimorar a escolha do recurso, e nas estimativas de custo e tempo. A comunicação com o serviço de tarifação é realizada de forma em que o serviço informa a lista de recursos disponíveis, indicando a capacidade computacional de cada máquina virtual, custos financeiros e detalhes do provedor. Com isso, esta fase tem o objetivo de coletar informações descritas pelo usuário e do ambiente computacional, com o intuito de reportar as informações necessárias para a terceira fase do processo de predição.

A terceira fase é composta por uma meta-heurística baseada na abordagem GRASP [26], que consome as informações da fase anterior, com o objetivo de minimizar custos e tempo de execução, levando em consideração as definições do usuário. O objetivo desta fase é que por meio da meta-heurística seja possível disponibilizar boas soluções, em tempo viável.

Após a construção da solução, realizada na terceira fase, são enviados para a interface do usuário, os *feedbacks* das soluções encontradas, e as estimativas de tempo e de custo da execução, completando assim a quarta e última fase da predição. Além destas informações, a fase de *Feedback* alimenta a base de dados histórica, a fim de registrar os dados da predição e, assim, continuamente aperfeiçoar as métricas da predição.

4.2.4 Camada de Infraestrutura

A Camada de Infraestrutura consiste em todos os recursos que os provedores de nuvens colocam à disposição da federação, e os seus respectivos *plugins* de integração. O principal objetivo desta camada é prover uma interface de comunicação entre o BioNimbuZ e os provedores de nuvens, utilizando para tal os *plugins* que mapeiam as requisições provenientes da plataforma, para os comandos específicos de cada provedor, individualmente.

Assim sendo, faz-se necessário desenvolver um *plugin* para cada plataforma de nuvem, tais como o *Elastic Compute Cloud* (EC2) [50], o *Simple Storage Service* (S3) [51], e outros.

Para realizar a comunicação de forma transparente, permitindo a execução de chamadas de procedimento que estão localizadas em outras máquinas, sem que o usuário perceba, e para auxiliar na organização e na coordenação do BioNimbuZ, utiliza-se de serviços voltados à sistemas distribuídos, chamados Apache Avro [30] e Apache ZooKeeper[31], serviços esses que são apresentados nas próximas seções.

4.2.5 Apache Avro

O Apache Avro [30] é um sistema de RPC e de serialização de dados desenvolvido pela Fundação Apache. Algumas vantagens deste sistema são o fato de ser livre, de utilizar de mais de um protocolo de transporte de dados em rede, e o suporte a mais de um formato de serialização de dados. Quanto ao formato dos dados, existe o suporte aos dados binários e aos dados no formato JSON [18]. Em relação aos protocolos pode-se utilizar tanto o protocolo HTTP [27] quanto um protocolo próprio do Avro [30].

O Avro foi criado para ser utilizado com um grande volume de dados, e possui algumas características definidas pela própria Fundação Apache, como uma rica estrutura de dados com tipos primitivos, um formato de dados compacto, rápido e binário e a integração de forma simples com diversas linguagens de programação [30]. Assim, o Avro foi escolhido como *middleware* da camada de comunicação da plataforma BioNimbuZ.

4.2.6 Apache Zookeeper

O Apache Zookeeper [31] é um serviço de coordenação de sistemas distribuídos, criado pela Fundação Apache, para ser de fácil manuseio. Ele utiliza um modelo de dados que simula uma estrutura de diretórios, e tem como finalidade facilitar a criação e a gestão de sistemas distribuídos, que podem ser de alta complexidade e de difícil coordenação e manutenção.

No Zookeeper são utilizados espaços de nomes chamados de *znodes*, que são organizados de forma hierárquica, assim como ocorre nos sistemas de arquivos. Cada *znode* tem a

capacidade de armazenar no máximo 1 Megabyte (MB) de informação, e são identificados pelo seu caminho na estrutura. Neles podem ser armazenadas informações que facilitem o controle do sistema distribuído, tais como metadados, caminhos, dados de conFiguração e endereços [49].

Existem dois tipos de *znodes*, os persistentes e os efêmeros. O primeiro é aquele que continua a existir mesmo depois da queda de um provedor, sendo útil para armazenar informações a respeito dos *jobs* que foram executados, e outros tipos de dados que não se deseja perder. Os efêmeros, por outro lado, podem ser criados para cada novo participante da federação, pois assim que este novo participante ficar indisponível, o *znode* efêmero referente a ele será eliminado e todos os outros componentes do sistema distribuído saberão que ele não se encontra disponível. Na Figura 4.9 pode ser visto um exemplo da estrutura hierárquica dos *znodes* implementados no BioNimbuZ, nos quais os metadados da aplicação são armazenados de maneira em que seja eficiente a forma de recuperação de informação da federação de nuvem.

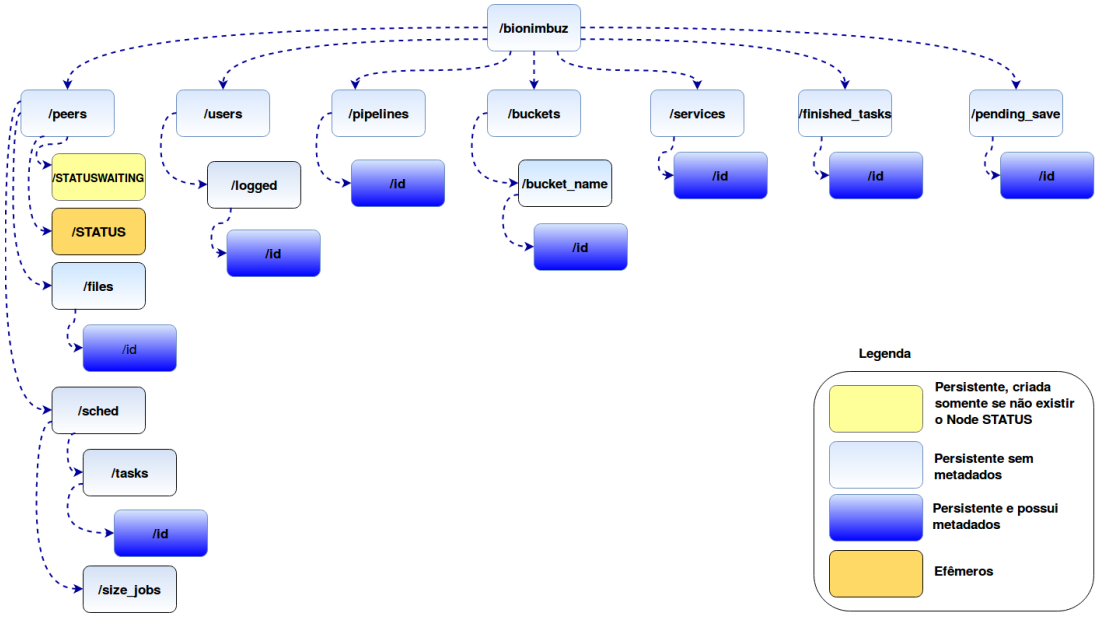

Figura 4.9: Estrutura Hierárquica dos *Znodes* no BioNimbuZ, adaptado de [68].

Como pode ser observado na Figura 4.9, o BioNimbuZ cria uma estrutura lógica da aplicação, de forma a receber as informações necessárias para a execução das tarefas de maneira dinâmica e transparente. Nesta estrutura existem *znodes* persistentes, para armazenar as informações importantes, que não podem ser apagadas caso alguma máquina fique indisponível, e um nó efêmero, que ativa uma *trigger*, chamada de *watcher*, para que os serviços recuperem as tarefas e os arquivos que estavam na máquina que ficou indisponível.

4.3 Considerações Finais

Este capítulo abordou os detalhes da plataforma de federação de nuvens BioNimbuZ, apresentando os serviços e as controladoras que o compõe. Além disso, foram mostrados detalhes sobre as tecnologias Apache Avro e Apache ZooKeeper, ambas utilizadas no BioNimbuZ. O Capítulo 5 apresentará a proposta do controlador de elasticidade para nuvens federadas, bem como um estudo de caso na plataforma BioNimbuZ.

Capítulo 5

Controlador de Elasticidade para Nuvens Federadas

A elasticidade consiste da capacidade da nuvem de aumentar ou diminuir os recursos, e para isso é necessário que o controlador seja capaz de criar e remover dinamicamente máquinas virtuais nas diversas nuvens da federação. Este trabalho propõe um controlador de elasticidade o qual é detalhado neste capítulo. Assim, na Seção 5.1 é proposto um controlador de elasticidade para nuvens federadas. Já na Seção 5.2 é feito um estudo de caso do controlador proposto utilizando a plataforma BioNimbuZ. Na Seção 5.3 são descritas algumas mudanças feitas na plataforma BioNimbuZ para integrar o controlador de elasticidade proposto. Por fim, na Seção 5.4 são mostrados os resultados dos testes da controlador proposto executados na plataforma BioNimbuZ.

5.1 Controlador de Elasticidade

O presente trabalho tem por objetivo propor um controlador de elasticidade para um ambiente de nuvens federadas, que suporte o provisionamento/desprovisionamento automático e sob demanda de máquinas virtuais, bem como a elasticidade horizontal e vertical dos recursos alocados.

Para isso, ele foi desenvolvido com três módulos principais, os quais são o *Manager*, o Controlador de Elasticidade Horizontal e o Controlador de Elasticidade Vertical. A Figura 5.1 apresenta a arquitetura deste controlador.

O *Manager* é o responsável por receber os eventos de disparo da elasticidade para as diversas nuvens, por decidir entre elasticidade horizontal ou vertical e por atualizar as informações das máquinas virtuais. O segundo módulo é o Controlador de Elasticidade Horizontal, o qual é responsável por instanciar as máquinas virtuais requisitadas, e além disso, ele também é responsável por remover máquinas virtuais que estão inutilizadas. Por

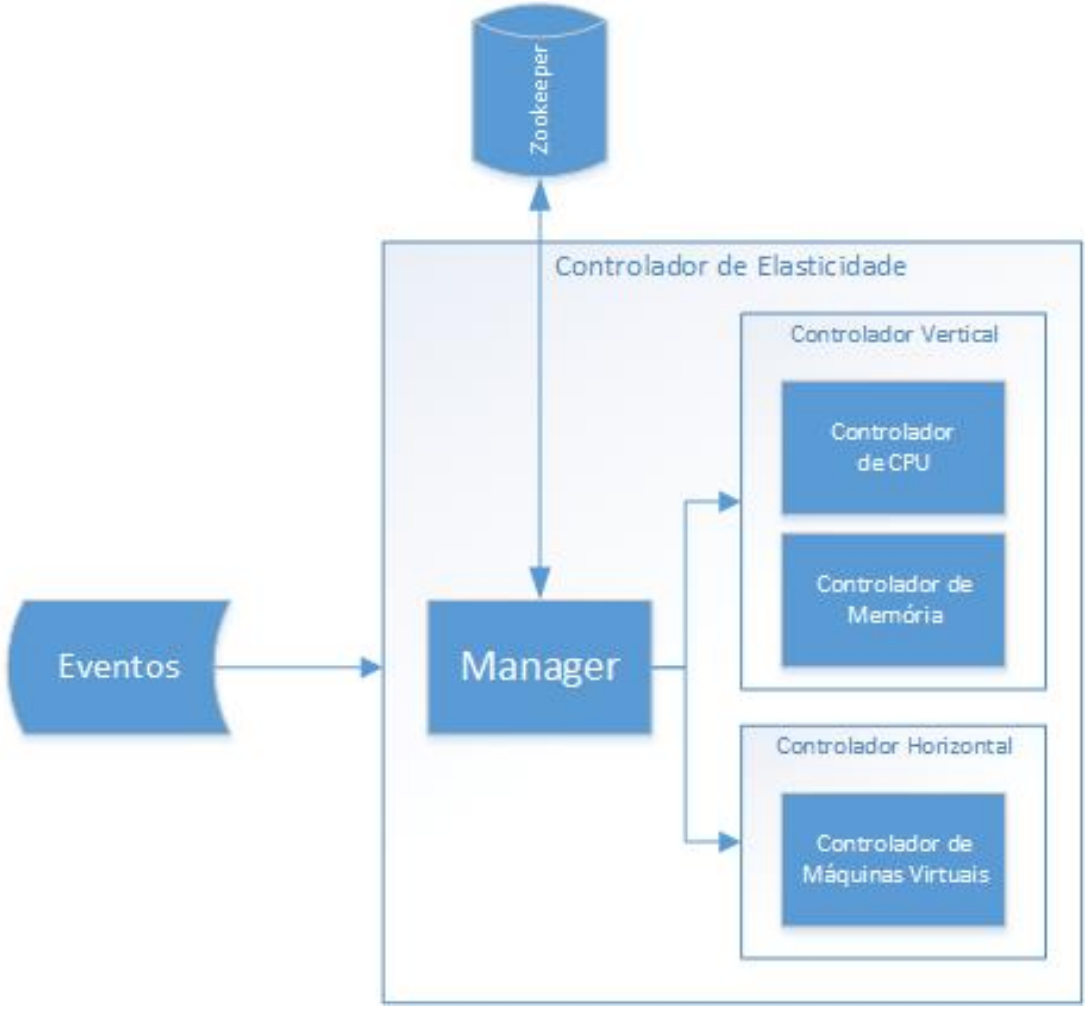

Figura 5.1: Controlador de Elasticidade Proposto Neste Trabalho.

último, tem-se o Controlador de Elasticidade Vertical, responsável por aumentar/diminuir o número de CPUs e de memória RAM das máquinas virtuais. Cada um desses elementos será descrito melhor nas próximas seções.

5.1.1 Projeto do *Manager*

O *Manager* é responsável por gerenciar periodicamente o sistema de monitoramento da federação, buscando o componente *status* de todas as máquinas virtuais, e coletando informações sobre a utilização dos recursos, ou seja, verificar a utilização de CPU e de memória RAM de cada VM ativa. É o *Manager* também que recebe os *requests* para a elasticidade, e decide se a elasticidade a ser disparada deve ser do tipo horizontal ou vertical. Se o *request* foi para a criação de uma nova máquina virtual, ele passa a tarefa para o Controlador Horizontal; Caso seja para aumentar uma máquina virtual já criada, ele passa então para o Controlador Vertical.

Outra tarefa importante do *Manager* é manter as informações (por exemplo, IP's, *status*, etc.) de todas as máquinas virtuais da federação, em algum serviço de coordenação de sistema distribuídos, por exemplo no Zookeeper.

Os principais eventos que fazem o módulo do *Manager* disparar a elasticidade são: criar máquina virtual, remover máquina virtual e executar elasticidade por falta de recursos. Esses eventos serão melhor descritos nos itens a seguir:

1. **Criar Máquina Virtual:** A criação automática de máquinas virtuais é um aspecto muito importante desta arquitetura de nuvem, pois permite que toda a infraestrutura seja transparente ao usuário, dando a ele uma visão única do sistema através do provisionamento dinâmico de recursos sob demanda, com o mínimo de esforço. Esta criação pode ser feita de duas maneira diferentes.

   Na primeira o usuário explicitamente requisita uma nova máquina virtual através de uma interface gráfica. Dessa forma, esse *request* vai para o *Manager* que passa para o Controlador Horizontal o *type* da máquina virtual que deve ser criada, e então cria a máquina virtual selecionada.

   Na segunda forma, mais transparente ao usuário, a escolha da máquina virtual se dá através de algum mecanismo de escolha automática. Este mecanismo deve levar em conta alguns parâmetros informados pelo usuário, tais como tempo máximo de duração da execução, escolha entre baixo custo ou de alto desempenho, entre outros. Da mesma forma que na anterior, um *request* vai para o *Manager* do controlador, que passa para o Controlador Horizontal o *type* da máquina virtual que deve ser criada.

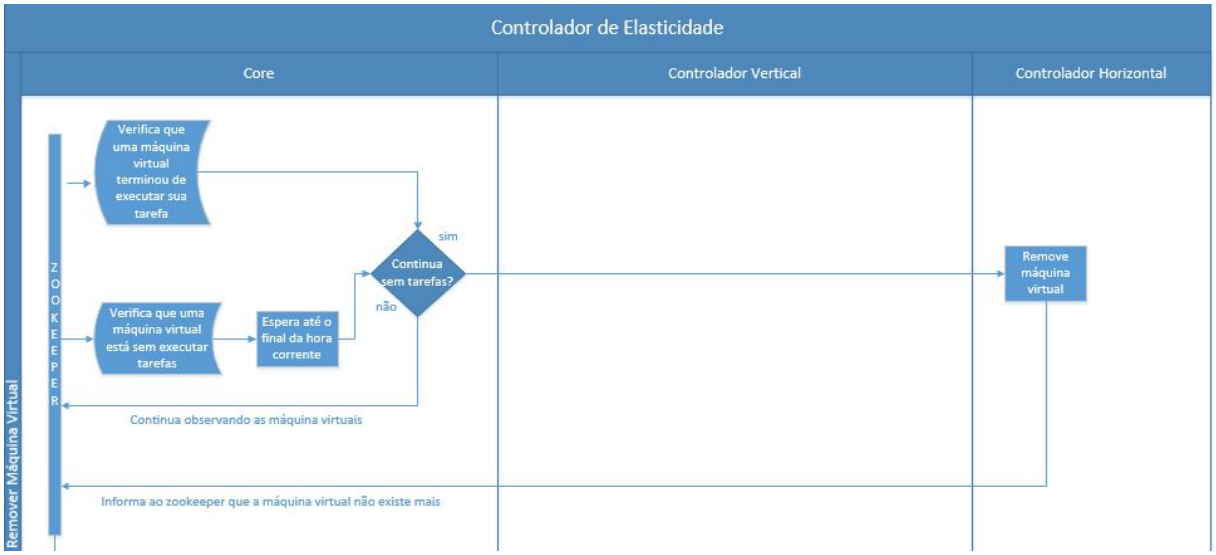

Figura 5.2: Criação de Máquina Virtual.

2. **Remover Máquina Virtual:** A remoção de máquinas virtuais é uma importante tarefa, e é utilizada para remover todas as máquinas que não executam mais tarefas. Isto é importante para diminuir o poder de processamento que não está sendo mais usado na federação e, consequentemente, diminuir os custos.

   Há duas formas de se fazer a remoção automática de máquinas virtuais. A primeira, mais geral, refere-se a máquinas que compartilham recurso. Neste caso, sempre que a máquina virtual não estiver executando nenhuma tarefa em algum momento, espera-se até o final da hora corrente, pois o método de pagamento é feito de hora em hora, e se ela continuar sem executar nenhuma tarefa, ela pode ser terminada e passar do estado de rodando para terminada.

   Já o segundo método refere-se ao caso em que uma única execução é feita na máquina virtual, não havendo compartilhamento de recursos. Neste caso, as máquinas virtuais executam a tarefa que lhes foi solicitada, e ao final da execução elas podem ser deletadas imediatamente. Após a exclusão da máquina virtual é preciso informar ao sistema que gerencia a federação de que a máquina não existe mais na federação.

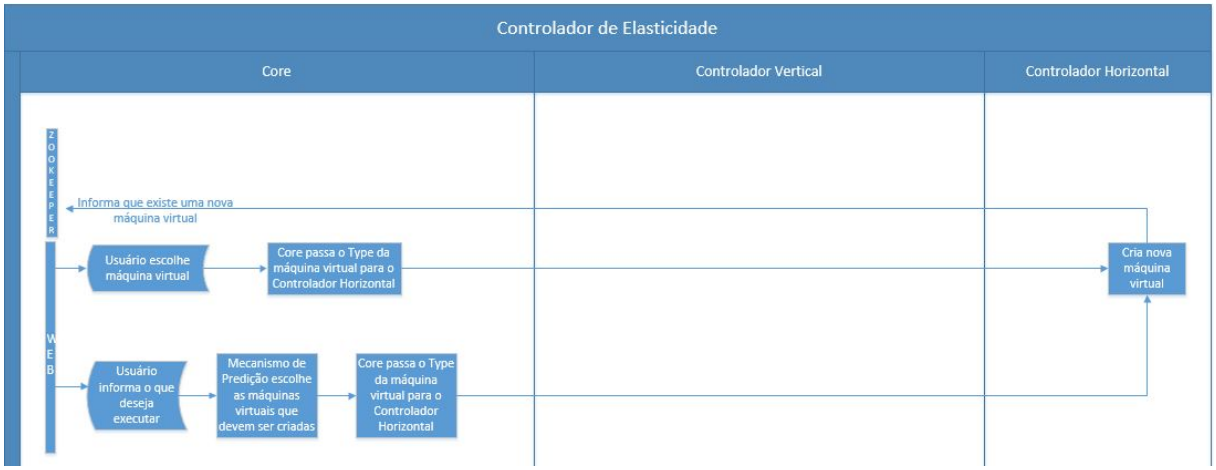

Figura 5.3: Remoção de Máquina Virtual.

3. **Expansão de Recursos:** A elasticidade de expansão de recursos é acionada quando há falta de recursos, e é utilizada para aumentar o poder de processamento da federação, após as máquinas já terem sido provisionadas. Neste caso, deve-se constantemente analisar as máquinas da federação e coletar os dados de utilização, analisando em qual provedor de nuvem está localizada a máquina virtual sobrecarregada que pode estar em alguma nuvem pública, ou em alguma nuvem privada. Essa análise é de fundamental importância por que somente a nuvem privada é capaz de aumentar o número de *cores* das máquinas virtuais em tempo de execução. Dessa forma,

o *Manager* escolhe entre a Elasticidade Vertical ou Horizontal, conforme pode ser visto na Figura 5.4.

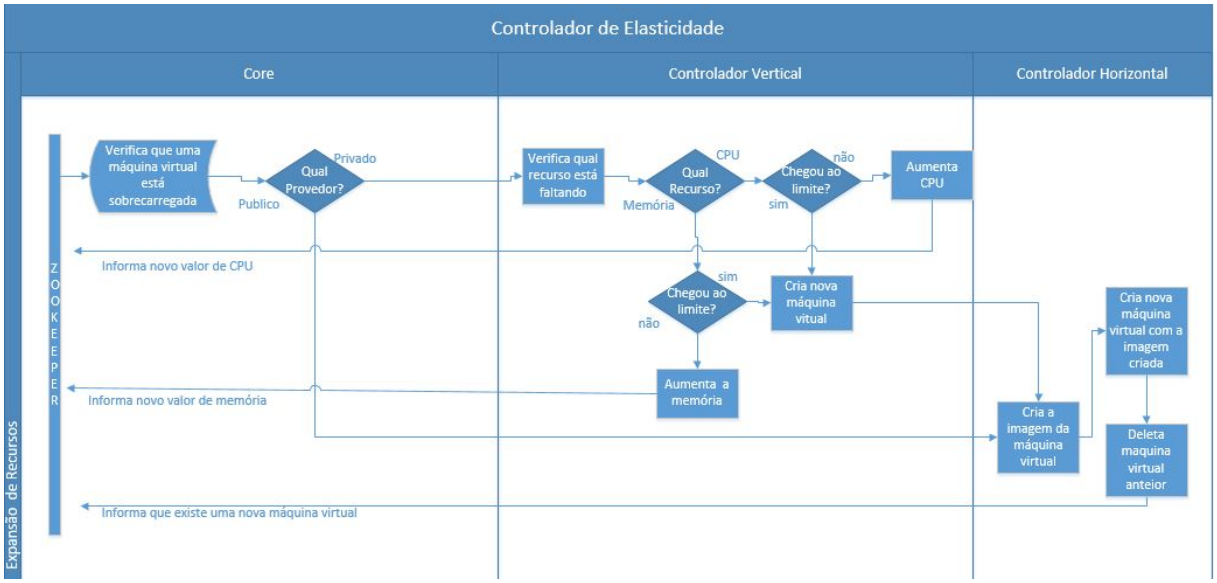

Figura 5.4: Expanção de Recursos.

O primeiro caso é quando a máquina virtual sobrecarregada está em algum provedor privado. Nesse caso, é adequado ativar a elasticidade vertical, pois é possível configurar nos provedores o valor máximo que uma máquina virtual pode aumentar em tempo de execução. A tarefa de elasticidade é passada então para o módulo Controlador de Elasticidade Vertical que decide se vai aumentar a CPU, a memória RAM ou ambos, com base nos valores extraídos das analises das máquinas virtuais. Caso a elasticidade vertical chegue ao seu limite, é criada então uma nova máquina virtual através do Controlador de Elasticidade Horizontal.

O segundo caso é quando a máquina virtual sobrecarregada está em um provedor público (Amazon, Google, etc), visto que este tipo de infraestrutura não permite uma elasticidade vertical. Assim, faz-se necessário realizar uma elasticidade horizontal. Neste caso, a tarefa de elasticidade é passada para o módulo Controlador de Elasticidade Horizontal, que cria então uma nova máquina virtual na federação.

5.1.2 Controlador de Elasticidade Horizontal

O Controlador de Elasticidade Horizontal é o responsável por criar novas máquinas virtuais. Essas máquinas podem ser criadas para executar alguma tarefa nova, que foi requi-

sitada pelo cliente da federação, ou para aumentar capacidade de execução das máquinas já provisionadas.

Este módulo deve fazer o uso das API's das nuvem públicas disponíveis. Assim sendo, quando chegar um *request* de criação de uma nova máquina virtual, deve-se descobrir de qual nuvem é a máquina a ser criada, e fazer a criação através da API. Essas API's têm a capacidade de executar as seguintes funções:

- Método *Create*: Responsável por informar para a API da nuvem relacionada para CRIAR uma nova máquina virtual;
- Método *Start*: Responsável por informar para a API da nuvem relacionada para RELIGAR uma máquina virtual PARADA;
- Método *Reboot*: Responsável por informar para a API da nuvem relacionada para REINICIAR uma máquina virtual;
- Método *Terminate*: Responsável por informar para a API da nuvem relacionada para APAGAR uma máquina virtual;
- Método *CreateAMI*: Responsável por informar para a API da nuvem relacionada para CRIAR uma nova IMAGEM de uma máquina virtual;

Por outro lado no caso em que é analisado constantemente se alguma máquina virtual está sobrecarregada, deve-se configurar os limites para a elasticidade. Esses limites podem ser configurados pelo administrador da federação, por meio de um arquivo de configuração, chamado *Template* de Elasticidade, como pode ser visto na Figura 5.5.

```
"ElasticityTrigger" : {
"Provider" : "Amazon",
"Properties" : {
"InstanceID" : "i-0948f0a6d7c1b31d1",
"MetricName" : "CPUUtilization",
"Statistic"  : "Average",
"Period"     : "3600000",
"UpperThreshold" : "90",
"LowerThreshold"   : "40"
}
},
```

Figura 5.5: Configuração do Template de Elasticidade

Na Figura 5.5 é possível observar que o administrador pode configurar informações importantes, tais como a métrica (que pode ser a utilização de CPU ou de memória RAM)

a estatística (que pode ser a média, o valor máximo ou o valor mínimo), o período (que é o tempo em que é coletado todos os valores para que a estatística possa fazer a sua informação) e por último, os *threshold's* (que são os valores de mínimo e de máximo da utilização da métrica selecionada) Por exemplo na Figura 5.5 tem-se uma máquina na Amazon com a métrica de utilização de CPU, na qual é coletada as informações durante uma hora, e calculada a média com um *threshold* de utilização máxima de 90%, e mínima de 40%. Comummente é utilizado como *threshold* máximo um valor de 90% como pode ser visto nos trabalhos de [19], [20] e [85].

5.1.3 Controlador de Elasticidade Vertical

O controlador de Elasticidade Vertical pode executar a expansão de uma máquina virtual de duas maneiras, por redimensionamento ou por substituição. A primeira, o redimensionamento, é feita através de configurações nos provedores de nuvem privada, tais como o CloudStack [44]. Nestes provedores é possível configurar o valor máximo que uma máquina virtual pode aumentar em tempo de execução. E, posteriormente, é configurado para que de maneira gradual aumente a CPU ou a memória RAM, até atingir o limite físico definido.

O segundo método, a substituição, é feito parando-se a máquina virtual que necessita aumentar a CPU ou a memória RAM, e então a mesma é reconfigurada. Essa reconfiguração é feita, por exemplo, mudando-se o tipo da máquina virtual criada.

5.2 Implementação no BioNimbuZ

Para a prova de conceito do controlador de elasticidade proposto, foi escolhida a plataforma de nuvens federadas BioNimbuZ [73]. Nessa plataforma a criação de máquinas virtuais é realizada de maneira totalmente manual, ou seja, o administrador do BioNimbuZ deve selecionar o provedor da federação que deseja criar uma máquina virtual, e em seguida criar a máquina e configurar todos os arquivos para a realização da federação de maneira manual. Além disso, no BioNimbuZ não há nenhum mecanismo que continuamente verifique a carga das máquinas criadas. Isso torna o BioNimbuZ um bom estudo de caso para o controlador de elasticidade proposto, visto que este controlador propõe justamente a criação e a elasticidade automática de máquinas virtuais no ambiente de federação de nuvem.

O estudo de caso tem como objetivo testar o controlador de elasticidade proposto, e busca tornar o BioNimbuZ uma plataforma que faça a criação/deleção automática das máquinas virtuais utilizadas nos *workflows*, bem como a elasticidade horizontal e vertical das máquinas virtuais provisionadas nas diversas nuvens da federação.

Para isso, fez-se necessário alterar o código das camadas de Aplicação e de Núcleo do BioNimbuZ, conforme são apresentados nas próximas seções.

5.2.1 Provisionamento Automático

A primeira característica implementada foi a de elasticidade horizontal, por meio do provisionamento automático de máquinas virtuais. Dessa forma, toda a criação de máquinas virtuais na plataforma BioNimbuZ passou a ser feita de maneira automática. Esta funcionalidade é de muita importância para o BioNimbuZ, pois permite ao usuário não se preocupar com a criação das máquinas virtuais a serem usadas na execução do seu *workflow*. Para isso, o fluxo implementado pelo Controlador na plataforma BioNimbuZ seguiu as etapas apresentadas na Figura 5.6, e descritas a seguir:

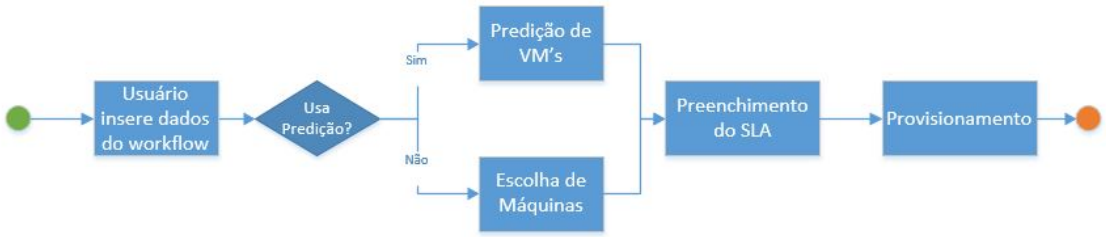

Figura 5.6: Controlador de Elasticidade Proposto neste Trabalho.

- **Etapa 1: Usuário insere os dados do *workflow***

  Ao clicar na opção, "Criar novo *Workflow*" do *menu*, o usuário é direcionado para uma tela na qual deve ser preenchida a descrição e os serviços que serão executados no *workflow*. Após a escolha dos serviços, o usuário é redirecionado para a tela mostrada na Figura 5.7, passando para a fase de *design* do *workflow*. Nela, são apresentados os elementos escolhidos pelo usuário que monta o *workflow* conforme sua necessidade, inserindo os arquivos de entrada desejados, para cada uma das fases do *workflow*. Estes dados (programas e arquivos) são utilizados pelo Serviço de Predição do BioNimbuZ para indicar quais são as máquinas virtuais mais indicadas para executar o *workflow* selecionado.

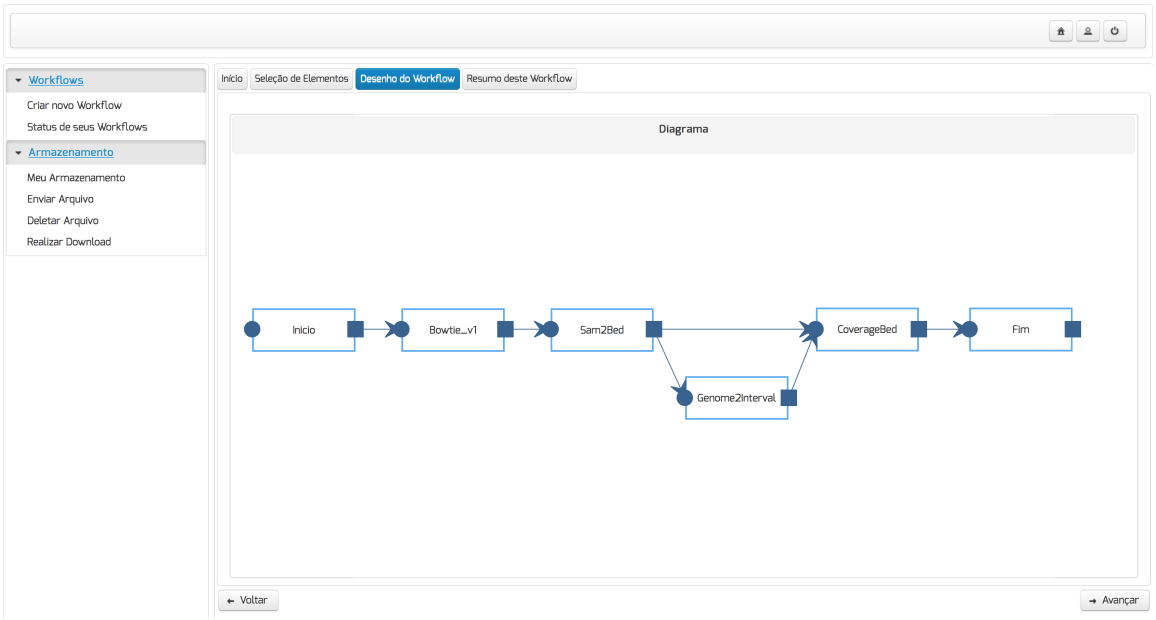

Figura 5.7: Tela de Montagem de Fluxo do *Workflow* da Aplicação.

- **Etapa 2: Predição das Máquinas Virtuais**

  Nesta fase o Serviço de Predição calcula e informa quais são as máquinas virtuais que devem ser criadas. Conforme apresentado no capítulo anterior, o Serviço de Predição utiliza uma abordagem baseada em GRASP. Essa abordagem utiliza-se das informações do *workflow* a ser executado, bem como tempo máximo de duração da execução e uma definição entre uma execução de baixo custo ou de alto desempenho. Estas informações restringem a lista de recursos compatíveis com as restrições impostas.

  A partir das definições do usuário, o Serviço de Predição através de uma metaheurística, informa ao controlador de elasticidade quais as máquinas virtuais que devem ser provisionadas. Esta etapa pode ser observada na Figura 5.8.

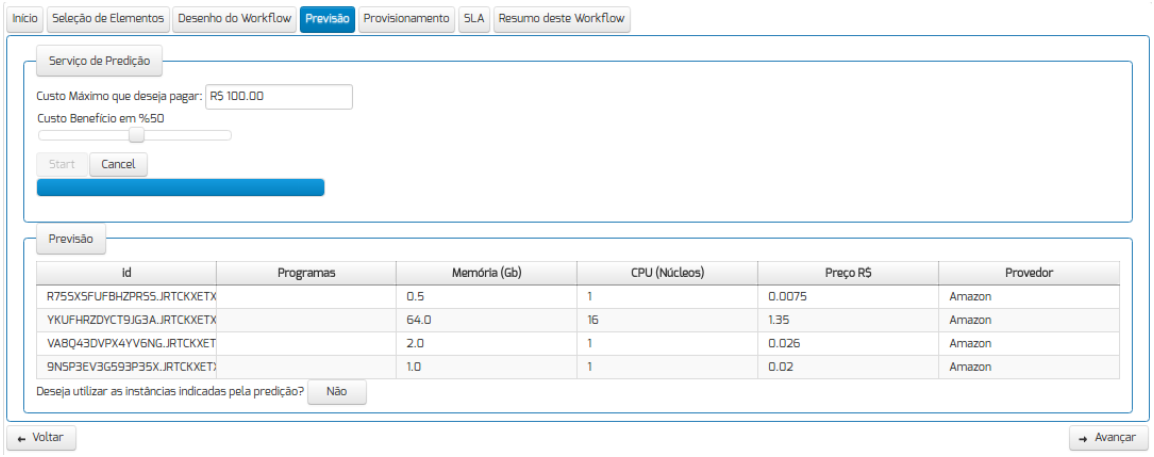

Figura 5.8: Tela de Predição.

Ao final desta etapa o usuário tem um *feedback* gráfico das máquinas virtuais que foram selecionadas, e que são inseridas em uma lista chamada *InstancesToCreate* com todas as máquinas escolhidas pela predição. O usuário, então, pode optar por usar esta lista ou não. Se decidir por utilizar as máquinas escolhidas, a lista então é passada para a etapa 4, onde o usuário irá preencher o SLA das máquinas selecionada. Se o usuário optar por não utilizar o que foi indicado, ele pode escolher novas máquinas na próxima etapa.

- **Etapa 3: Escolha das Máquinas**

  Caso o usuário não desejar utilizar as máquinas escolhidas na etapa de Predição, o usuário então é redirecionado para a etapa de provisionamento. Nesta etapa é mostrado para o usuário todas as máquinas virtuais disponíveis para execução de seus *workflows*. Essa lista de máquinas é constantemente atualizada por meio de um arquivo *json* que é gravado a partir dos provedores de nuvem, onde contém todas as informações das máquinas virtuais disponíveis, como pode ser visto na Figura 5.9.

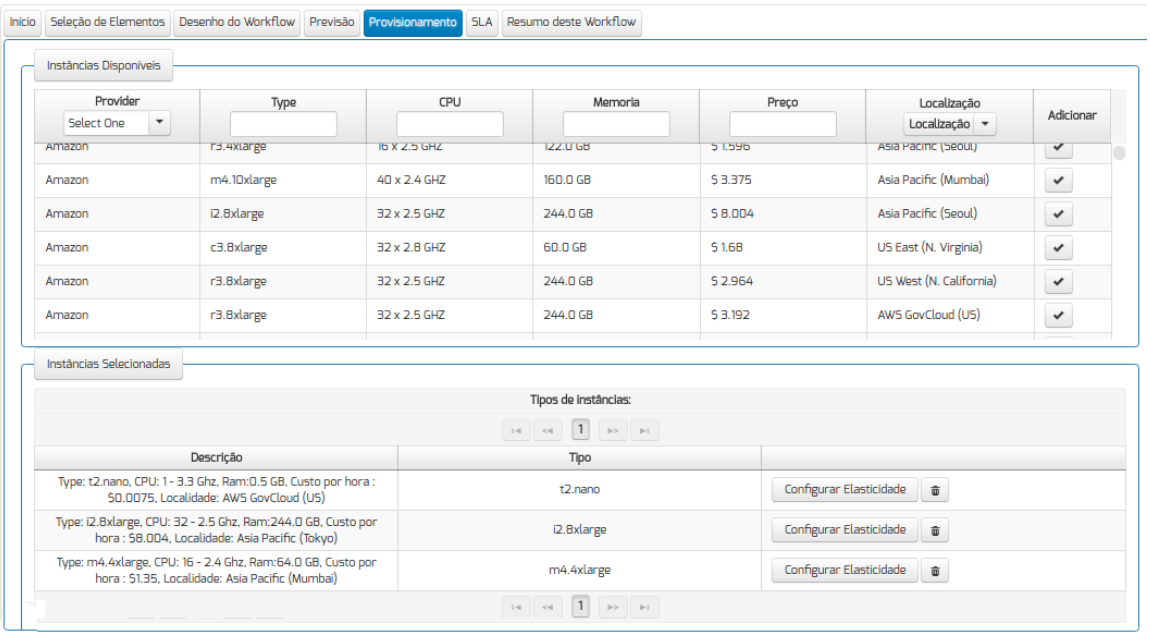

Figura 5.9: Tela de Escolha das Máquinas Virtuais.

O usuário então escolhe uma ou mais máquinas virtuais, baseado principalmente pelo *Instance Type* que é o tipo da máquina virtual, montando assim a lista *InstancesToCreate*. Essa lista será utilizada, posteriormente, pelo controlador de elasticidade para fazer o provisionamento das máquinas virtuais.

Outra configuração importante que o usuário pode fazer é em relação à ativar ou não a elasticidade para a máquina virtual selecionada. Para poder configurar essa opção o usuário tem que clicar no botão "configurar elasticidade", como pode ser visto na Figura 5.10.

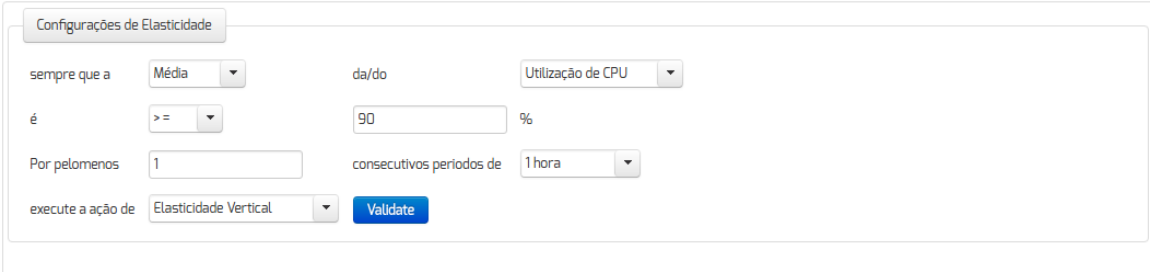

Figura 5.10: Tela de Configuração de Elasticidade.

NA tela apresentada na figura 5.10 é possível configurar se o usuário deseja ou não que a máquina virtual selecionada execute automaticamente as ações de elasticidade. O usuário decidindo utilizar a elasticidade automática, ele deve informar os parâmetros de elasticidade.

- **Etapa 4: Preenchimento do SLA**

  Nesta etapa o usuário preenche o Acordo de Nível de Serviço - SLA, que é o contrato do usuário junto ao BioNimbuZ. Neste contrato são informados os parâmetros de Serviço do BioNimbuZ, tais como a porcentagem de funcionamento mensal e, o *uptime*, como pode ser visto na Figura 5.11.

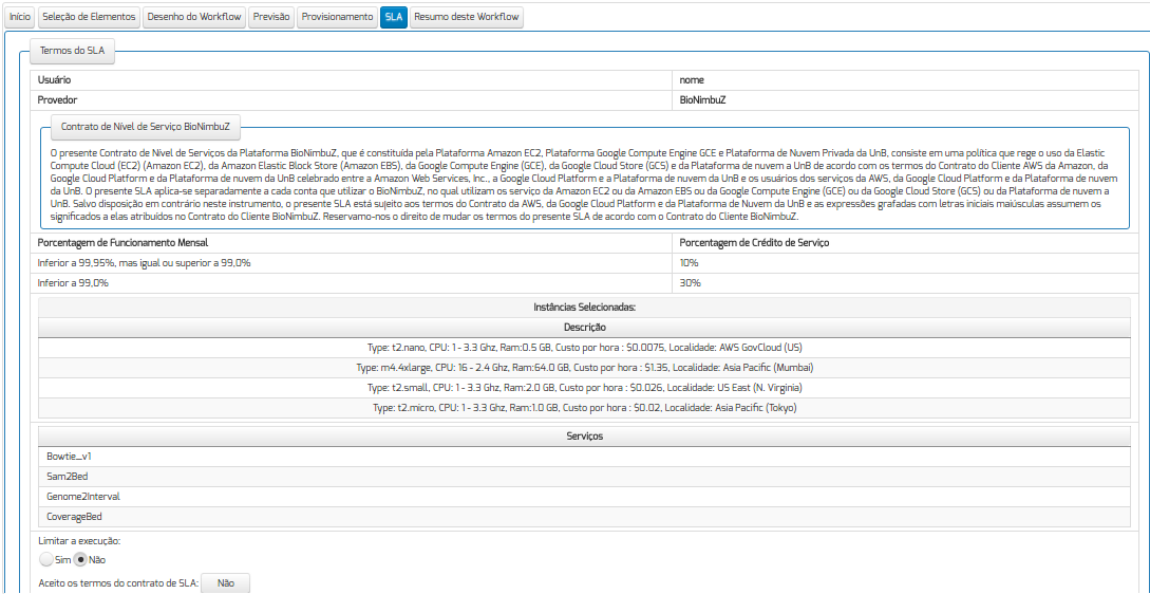

Figura 5.11: Tela do Acordo de Nível de Serviço do BioNimbuZ.

- **Etapa 5: Provisionamento**

  Após o usuário ter aceitado os termos de SLA, escolhendo a opção "Confirmar *Workflow*" o controlador de elasticidade utiliza a lista *InstancesToCreate* com os parâmetros das máquinas virtuais que devem ser provisionadas. Estas informações então são passadas para o controlador de elasticidade no Núcleo do BioNimbuZ.

  O controlador de elasticidade recebe este *request*, e através das API's das nuvens faz a criação das máquinas virtuais requisitadas, e retorna um *response* com os IP's públicos das máquinas virtuais criadas. Estas máquinas virtuais já são criadas com os *scripts* de configuração *node.yaml* e *conf.yaml*, para que possam se conectar no Zookeeper e, desta forma possam fazer parte da federação de máquinas virtuais do BioNimbuZ, como pode ser visto na Figura 5.12.

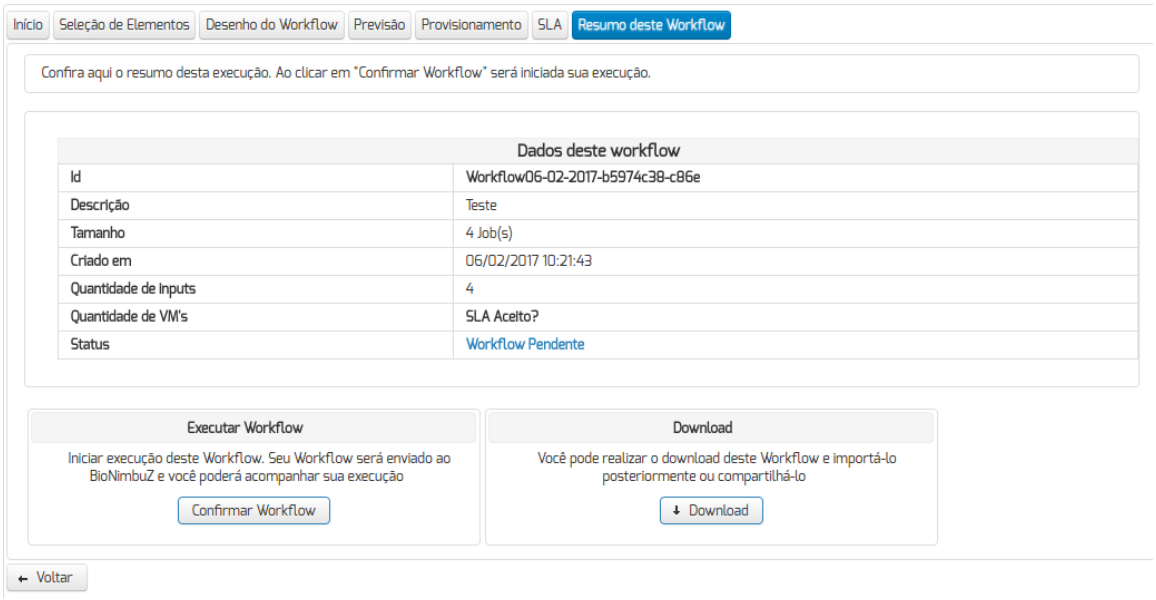

Figura 5.12: Tela para Confirmar *Workflows* e Provisionar as Máquinas Virtuais.

5.2.2 Elasticidade Automática

A elasticidade automática foi desenvolvida utilizando-se das API's para poder criar novas máquinas virtuais. Quando o usuário cria uma máquina virtual de sua escolha, ele tem a opção de ativar ou não o serviço de elasticidade automática. Esse módulo utiliza o Serviço de Monitoramento do BioNimbuZ, que foi modificado para que constantemente verifique a utilização de CPU das máquinas virtuais selecionadas. Essas métricas são obtidas através de *scripts* que são executados nas máquinas virtuais selecionadas para fazer elasticidade vertical. Esses *scripts*, então, são executados conforme os parâmetros definidos pelo administrador do BioNimbuZ na tela de configurações de elasticidade, como pode ser visto na Figura 5.10.

Nesse caso, é possível observar que a configuração foi definida para que sempre que a média de utilização de CPU for maior ou igual a 90%, por um período de uma hora, a máquina virtual selecionada executa a elasticidade. Para isso, primeiramente, é criada uma imagem da máquina virtual, depois é criada uma nova máquina virtual com essa imagem porém de um tipo maior, para que dessa forma a nova máquina tenha as mesmas configurações da anterior (a máquina virtual criada já sobe com as configurações para fazer parte da federação) e então a máquina virtual antiga é deletada. Essa implementação pode ser vista na Figura 5.13.

```
public void executeElasticity(String id, String instanceType) throws IOException        {
    this.setup();
    this.createinstance(instanceType, this.createami(id));
    this.terminate(id);


}
```

Figura 5.13: Função de Elasticidade Implementada.

No caso da Amazon as métricas puderam ser obtidas através do Amazon CloudWatch [25]. Através desta API pode-se requisitar as informações das máquinas conforme as regras estabelecidas na Figura 5.10. Essa implementação pode ser vista na Figura 5.14.

```
public GetMetricStatisticsRequest request(final String instanceId) {
    final long start = 1000 * 60 * 60 * 1;
    final int period = 60 * 15;
    return new GetMetricStatisticsRequest()
            .withStartTime(new Date(new Date().getTime() - start))
            .withNamespace("AWS/EC2")
            .withPeriod(period)
            .withDimensions(new Dimension().withName("InstanceId").withValue(instanceId))
            .withMetricName("CPUUtilization")
            .withStatistics("Average", "Maximum")
            .withEndTime(new Date());
}
```

Figura 5.14: Função de Coleta de Métricas da *Amazon Cloud Watch*.

5.2.3 Configurações e Estatísticas de Elasticidade

Uma parte importante deste trabalho foi também a adição do *menu* de Configurações e Estatísticas. Neste *menu* há três funcionalidades principais, que são os *submenus* de instâncias da federação, configurações de elasticidade e configurações de chaves de acesso, como pode ser visto na Figura 5.15 e serão explicados a seguir.

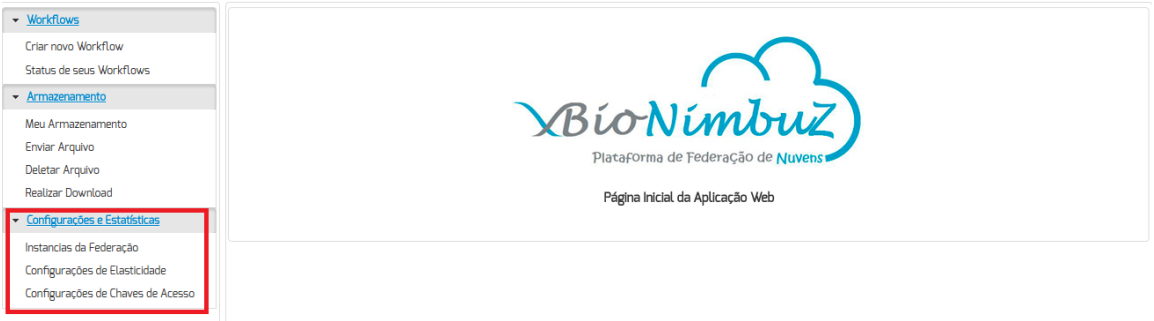

Figura 5.15: Configurações do BioNimbuZ.

- **Instâncias da Federação -** Todas as máquinas virtuais criadas pelo usuário podem ser vistas na tela Instâncias da Federação, como podem ser observados nas

Figuras 5.16 e 5.17. Assim, em cada máquina virtual é possível ter as seguintes funcionalidades: Ver informações, dar *start* em uma máquina parada, parar, reiniciar e terminar, Além de uma funcionalidade extra de Elasticidade, onde é possível iniciar um procedimento de elasticidade por replicação manualemte.

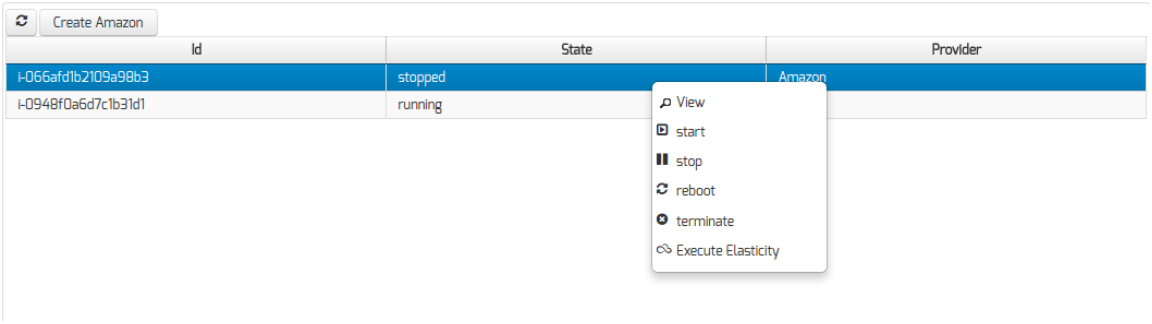

Figura 5.16: Tela Instâncias da Federação.

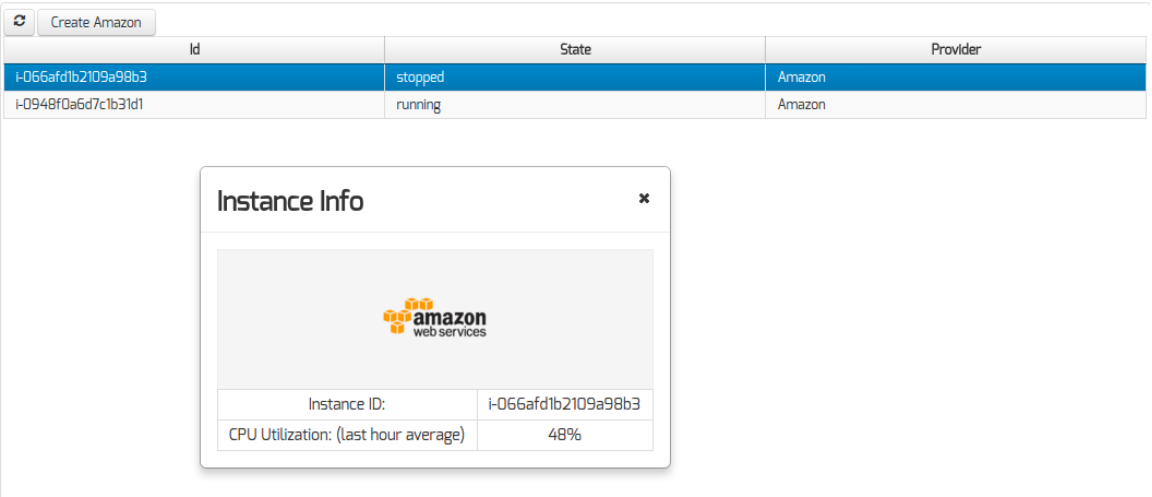

Figura 5.17: Tela Instâncias da Federação, com Informações da Instância no Detalhe.

- **Configurações de Elasticidade -** Nesta tela é possível reconfigurar as ações de elasticidade automáticas, previamente selecionadas no momento do provisionamento. Nesta tela devem ser feitas as escolhas dos parâmetros de elasticidade e, principalmente, se o usuário deseja que a máquina virtual execute a elasticidade automaticamente, conforme pode ser visto na Figura 5.10.
- **Configurações de Chaves de Acesso -** Para que as API's das nuvens públicas possam funcionar, é necessário que o usuário informe suas credencias de acesso. Essas credenciais podem ser configuradas nesta tela, como mostrado na Figura 5.18.

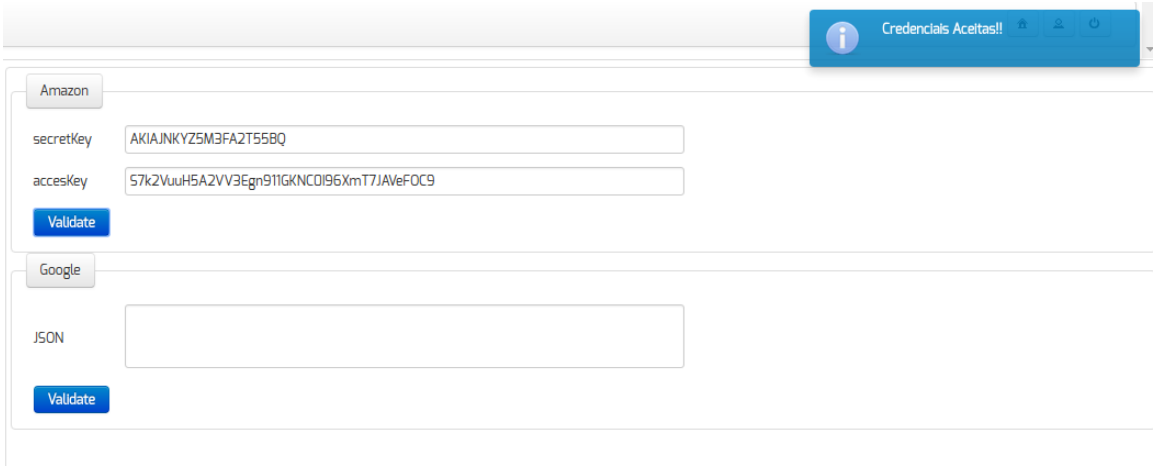

Figura 5.18: Tela de Configuração de Autenticação.

5.3 Mudanças nos Módulos do BioNimbuZ

Para testar o controlador de elasticidade proposto por este trabalho, foram necessários alguns ajustes na plataforma BioNimbuZ, adaptando a plataforma para receber o controlador de elasticidade.

Alguns desses ajustes foram feitos, na *interface web* (como mostrados na seção anterior), no Serviço de Tarifação, no Serviço de Monitoramento e no Serviço de Escalonamento. Assim essas mudanças serão explicadas nas seções seguintes.

- **Serviço de Tarifação:** O serviço de tarifação da plataforma, teve que ser adaptado para que o mesmo buscasse a informação de todos os tipos de instâncias disponíveis, na Amazon EC2 [50] e da Google Cloud Compute Engine [36], retornando então uma lista de instâncias para a aplicação.
- **Serviço de Monitoramento:** No serviço de monitoramento foi adicionado mais um parâmetro a ser analisado. Esse parâmetro contém informações do usuário, tais como *workflows*, instâncias, IP, programa, e qual usuário criou a máquina virtual. Essa mudança resultou na nescessidade de adicionar um nó a mais na estrutura lógica do zookeeper, que é no dos *workflows* do usuário, conforme apresentado na Figura 5.19.

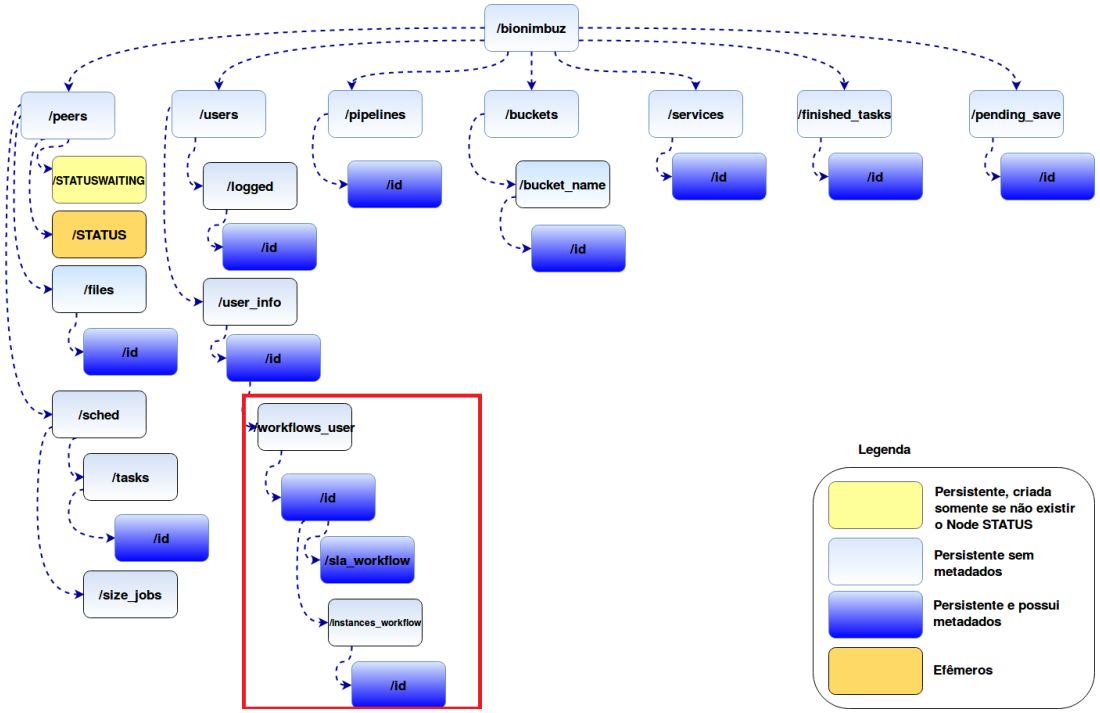

Figura 5.19: Nova Estrutura Hierárquica dos *Znodes* no BioNimbuZ.

- **Serviço de Escalonamento:** No algoritmo de escalonamento C99 [4], é levado em consideração todas as máquinas previamente provisionadas, resultando assim em um compartilhamento obrigatório dos recursos ativos no BioNimbuZ. Este escalonador teve que ser alterado, pois no estado atual do BioNimbuZ os usuários devem ter o seu próprio recurso, pois a execução de um *workflow* não pode influenciar na execução de outro. Para isso, implementou-se um novo escalonador, que escalona as tarefas somente para as máquinas que foram criadas pelo usuário .

5.4 Testes

Para analisar a eficiência e a funcionalidade do controlador proposto neste trabalho, os testes foram definidos em dois grupos. Os testes de elasticidade vertical e os de elasticidade horizontal. O primeiro deles foram os testes de elasticidade horizontal, aumentando/diminuindo a capacidade de execução da aplicação web do BioNimbuZ, e também da sua criação automática de máquinas virtuais para a federação. O segundo grupo de testes realizados foram os de elasticidade vertical, no qual optou-se por testar a elasticidade por meio da técnica da substituição, tanto da própria plataforma do BioNimbuZ quanto dos seus serviços.

5.4.1 Elasticidade Horizontal

A elasticidade horizontal proporciona ao BioNimbuZ a capacidade de se adaptar as oscilações de carga através da criação de novas máquinas virtuais. Para estes testes optou-se implementar o método da replicação, onde a máquina sobrecarregada é duplicada e todo o tráfego é balanceado entre as máquinas. Outro teste importante é o de criação automática de máquinas virtuais, esta funcionalidade permite ao BioNimbuZ a elasticidade horizontal para a federação, adicionando novas máquinas virtuais sempre que um novo *workflow* é submetido. Assim sendo, os próximos dois tópicos mostram como foram feitos os testes de elasticidade horizontal.

- Plataforma BioNimbuZ - Replicação:

  Para os testes de elasticidade horizontal automático foram feitos testes no servidor web da plataforma BioNimbuZ, devido a facilidade de simular novos clientes e novas tarefas. Para que haja sempre algum BioNimbuZ ativo, é feito, primeiramente, uma elasticidade horizontal para depois fazer uma elasticidade vertical.

  Para esse teste também foi usado o pacote do Linux *stress*, simulando mais usuários acessando o servidor web. O controlador de elasticidade foi setado para criar uma nova máquina virtual quando a capacidade total da utilização de CPU tivesse com a média acima de 90%, durante a última hora. Ou deletar uma máquina virtual, quando a utilização de CPU tivesse com a média abaixo de 30%, na última hora.

Figura 5.20: Elasticidade Horizontal por Replicação.

Como pode ser visualizado na Figura 5.20, quando a carga de utilização da CPU aumentou para mais de 90% na média por 1h, foi então crida uma nova máquina, a qual é replicada da já existente. Para que a distribuição de carga pudesse ser feita entre os novos servidores, foi utilizado um servidor de balanceamento automático da Amazon [50]. Assim, com a elasticidade horizontal implementada, aumenta-se o desempenho do servidor, pois todas as vezes que o servidor estiver próximo de ser saturado, ele é substituído por um com maior capacidade.

- Provisionamento Automático

  Para os testes do provisionamento automático foram criadas duas máquinas virtuais, uma na Amazon [50] e uma na Google [36]. Para cada máquina virtual foi medido o tempo de criação da máquina, e o tempo de gravação no Zookeeper [31]. Para isso, foi tirada a média do tempo de criação de 10 máquinas virtuais de cada tipo, resultando nos tempos apresentados na Tabela 5.1. Os tempos para deleção não foram considerados pois levam, na média, menos de 5 segundos para todos os tipos de máquina.

Tabela 5.1: Tempo Médio de Criação das Máquinas Virtuais.

Provedor	**Tipo**	**CPU**	**Memória**	**Tempo para criação**
Amazon	t2.nano	1	0.5	00:00:32
Amazon	t2.micro	1	1	00:00:38
Amazon	t2.small	1	2	00:00:53
Amazon	t2.medium	2	4	00:01:33
Amazon	t2.large	2	8	00:02:22
Amazon	t2.xlarge	4	16	00:03:54
Amazon	t2.2xlarge	8	32	00:04:38
Amazon	m4.large	2	8	00:02:25
Amazon	m4.xlarge	4	16	00:03:23
Amazon	m3.medium	1	3.75	00:01:12
Amazon	m3.large	2	7.5	00:02:26
Amazon	m3.xlarge	4	15	00:03:48
Google	f1-micro	1	0.6	00:00:48
Google	g1-small	1	1.7	00:00:52
Google	n1-standard-1	1	3.75	00:01:10
Google	n1-standard-2	2	7.5	00:01:12
Google	n1-standard-4	4	15	00:01:36
Google	n1-standard-8	8	30	00:02:20

Como pode ser observado na Tabela 5.1, o tempo médio para provisionar uma máquina na Amazon foi de um 1 minuto e 57 segundos (00:01:57). E o tempo médio para o provisionamento na Google foi de 1 minuto e 24 segundos (00:01:24). Resultando, assim, que o provisionamento na Google, em média, é mais rápido.

Assim, o provisionamento adiciona à plataforma do BioNimbuZ um *overhead* inerente a criação das máquinas virtuais. Porém, se for considerado que essa tarefa era feita de maneira totalmente manual, e que um usuário experiente levaria pelo menos 10 minutos para escolher e provisionar manualmente qualquer máquina virtual, esse *overhead* acaba sendo ignorado para os casos de provisionamento automático.

5.4.2 Elasticidade Vertical

A elasticidade vertical proporciona ao BioNimbuZ a capacidade de se adaptar as oscilações de carga por meio do aumento da capacidade de execução de uma máquina virtual específica. Para estes testes optou-se implementar o método da substituição, onde a máquina sobrecarregada substituída por outra de maior capacidade de execução. Foram feitos dois tipos de testes, onde o primeiro deles a máquina sobrecarregada era o servidor *web* do BioNimbuZ, e o segundo teste foi feito nos serviços executados pelo BioNimbuZ. Assim sendo, os próximos dois tópicos mostram como foram feitos os testes de elasticidade vertical.

- Plataforma BioNimbuZ - Substituição:

  Para a realização destes testes foi utilizado um cenário com quatro máquinas virtuais. Dessas máquinas, três foram criadas na Google e uma máquina na Amazon, compondo a federação de nuvem. As três máquinas virtuais da Google são todas do tipo *n1-standard-2*, com as configurações de 2 núcleos e 7,5 GB de memória RAM. Dessas três máquinas, uma foi usada para o servidor web, outra para o núcleo do BioNimbuZ e Zookeeper, e a terceira máquina, chamada V1, foi usada para a execução das tarefas. A máquina virtual criada na Amazon, foi do tipo *t2.medium*, com 2 núcleos e 4 GB de memória RAM, chamada de VM2. Esse cenário é mostrado na Figura 5.21.

  Em seguida foi setado que o servidor web pudesse fazer a elasticidade vertical automática. Devido ao fato de ser um servidor que constantemente recebe novas requisições e não tem problema algum em parar por alguns minutos.

  Além de coletar as estatísticas de CPU e de memória RAM das máquinas V1 e V2, também foram coletadas informações do servidor web e do Núcleo do BioNimbuZ, como podem ser observadas nas Figuras 5.22 e 5.23.

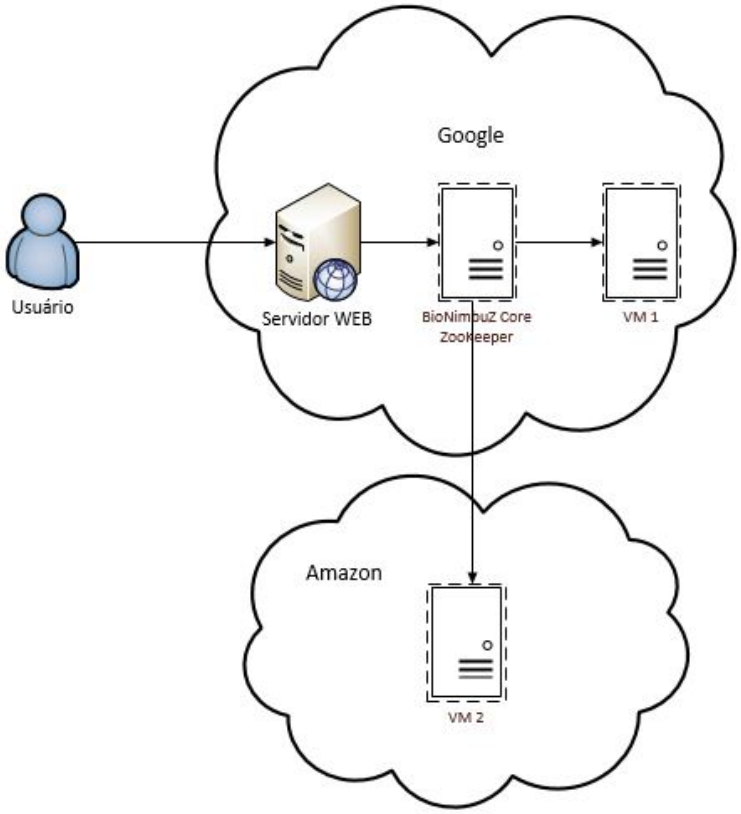

Figura 5.21: Infraestrutura Utilizada para o Teste do *Workflow*.

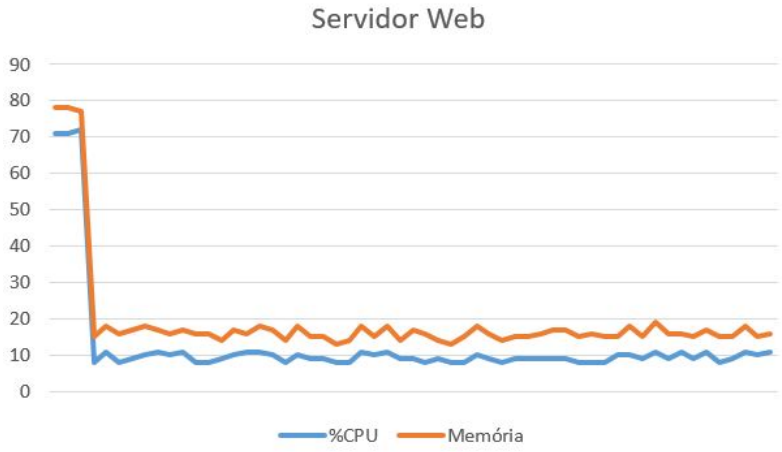

Figura 5.22: Monitoramento do Servidor Web.

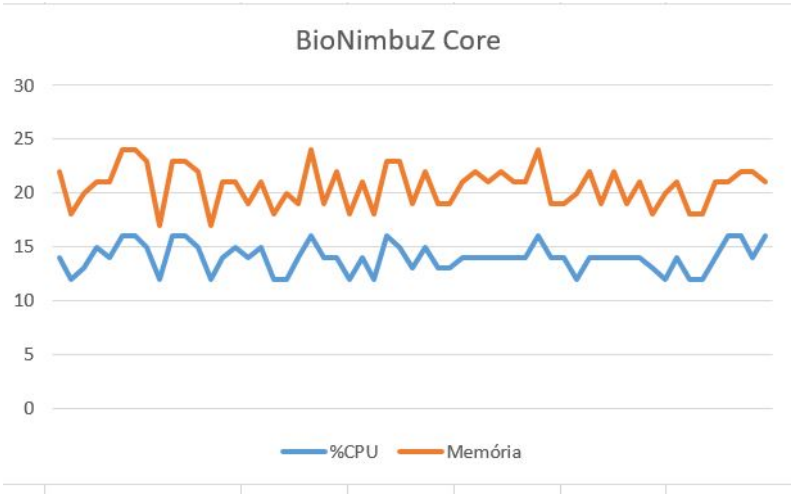

Figura 5.23: Monitoramento do Núcleo do BioNimbuZ.

Como apresentado nas Figuras 5.22 e 5.23 o consumo de CPU e de memória RAM, tanto no servidor web quanto no Núcleo do BioNimbuZ, mantiveram-se ligeiramente constantes, não necessitando assim executar ações de elasticidade, portanto, fez-se necessário realizar algumas testes simulados de carga, que pudessem forçar a ativação do serviço de elasticidade que serão mostrados na próxima seção.

Os testes de elasticidade vertical automática foram feitos no servidor web da plataforma BioNimbuZ, devido a sua característica de poder parar a sua execução para substituir o servidor por um maior.

Para isso, foi utilizado um pacote do Linux, chamado *stress* para o teste de carga. Nesse pacote é possível simular uma carga (maior ou menor) de CPU em qualquer máquina, e assim, disparar os eventos de elasticidade. O controlador de elasticidade foi setado então para aumentar a capacidade da máquina virtual quando a utilização de CPU tivesse com a média acima de 90% durante uma última hora.

A simulação para aumentar a carga foi feita provisionando uma máquina virtual na Amazon (*t2.xlarge*), com 4 núcleos de processamento e 16GB de memória RAM. Durante a primeira hora foi utilizado 25% de CPU, na hora seguinte 50% e na terceira hora 100%, disparando o serviço de elasticidade. Depois de ter disparado o evento de elasticidade, foi tirada uma imagem da máquina anteriormente provisionada, criada uma nova máquina virtual (*t2.2xlarge*) com 8 núcleos de processamento e 32GB de memória RAM. Em seguida foi deletada a máquina virtual antiga. Um gráfico deste teste pode ser visto na Figura 5.24. Este processo de elasticidade durou ≈ 9 minutos para que o Núcleo do BioNimbuZ detectasse novamente que havia um novo servidor web.

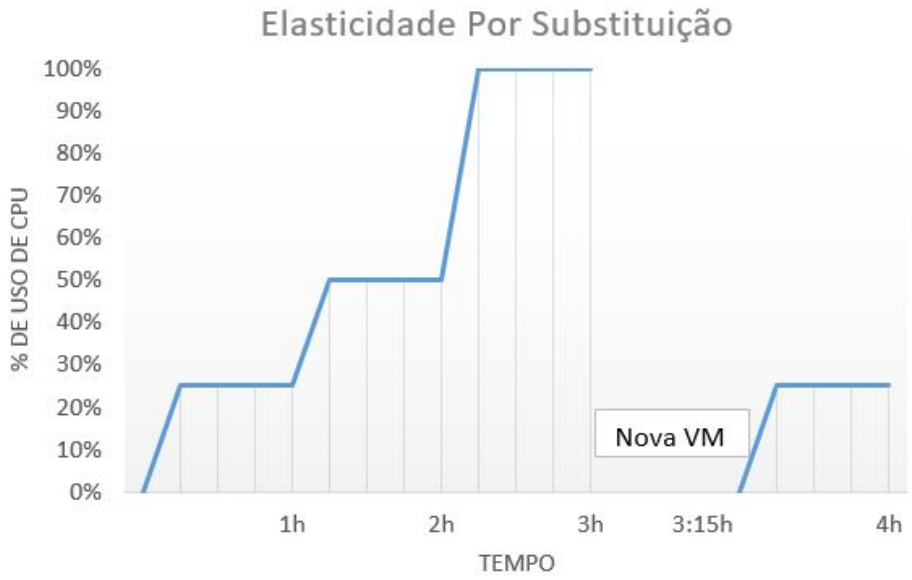

Figura 5.24: Elasticidade Vertical por Substituição - Nova VM.

- Serviços da Federação - Substituição:

  Para a realização dos testes de elasticidade vertical para os serviços que são executados no BioNimbuZ foi escolhido a execução de um software de Bioinformática chamado *trinity* [37].

  Primeiramente, foi executado o *trinity* em uma máquina virtual com somente 1 CPU. Essa execução resultou em um tempo total de execução de aproximadamente 3 horas e 40 min, como pode ser visto na Figura 5.25.

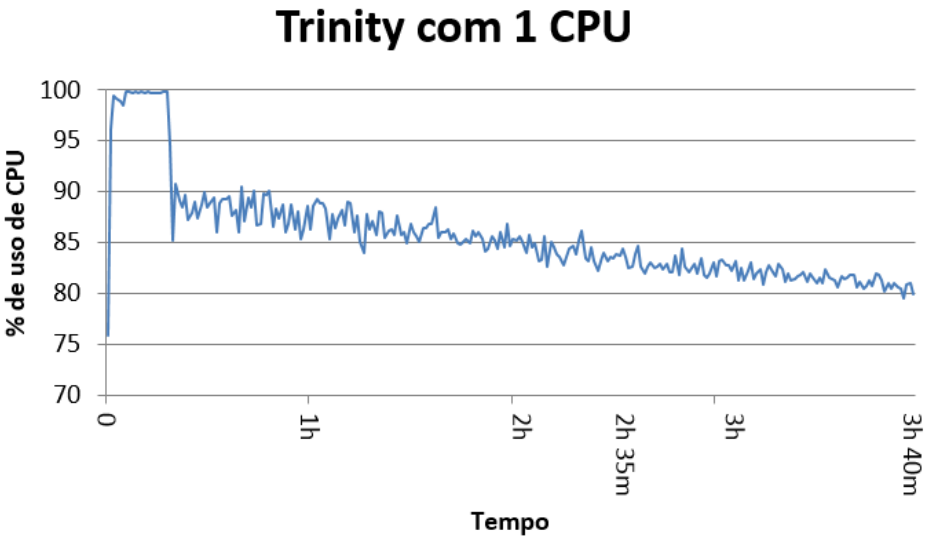

Figura 5.25: Elasticidade Vertical por Substituição - Trinity 1 CPU.

Diante disso, foi configurado no controlador proposto que o mesmo realizasse uma elasticidade vertical através de substituição, sempre que o consumo de CPU em uma máquina virtual fosse maior do que 90% por uma hora. Assim, pode ser percebido na execução mostrada na Figura 5.25, durante a primeira hora houve uma média de 91% de consumo de CPU, resultando então em um gatilho para a elasticidade. Um detalhe do consumo desta primeira hora pode ser observada na Figura 5.26.

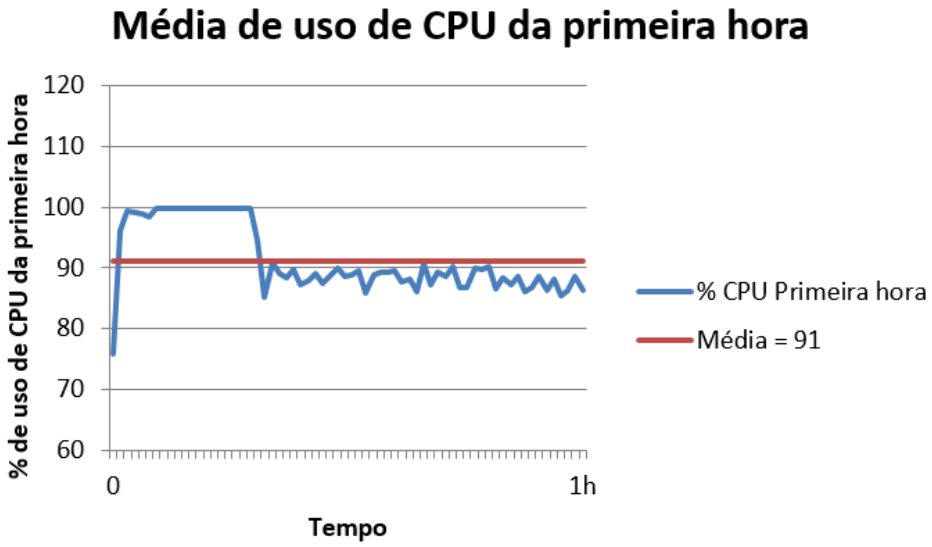

Figura 5.26: Elasticidade Vertical por Substituição - Média.

Depois dessa primeira hora, foi verificado, a necessidade da elasticidade da máquina virtual, que foi substituída por outra máquina maior (com 2 núcleos), e então foi reiniciada a execução da tarefa nesta nova máquina virtual, finalizando, assim, o procedimento de substituição.

A nova execução pode ser observada na Figura 5.27. Nesta nova execução podemos observar que na primeira hora a tarefa executa na máquina virtual com uma única CPU, e como o percentual de uso da CPU foi na mádia maior do que o limite de 90% configurado, a máquina com 1 CPU é substituída por uma máquina com 2 CPUs. O procedimento de substituição realizado, demorou 15 minutos. Assim a re-execução da mesma tarefa na máquina mais robusta demorou um tempo total de 2 horas e 35 minutos.

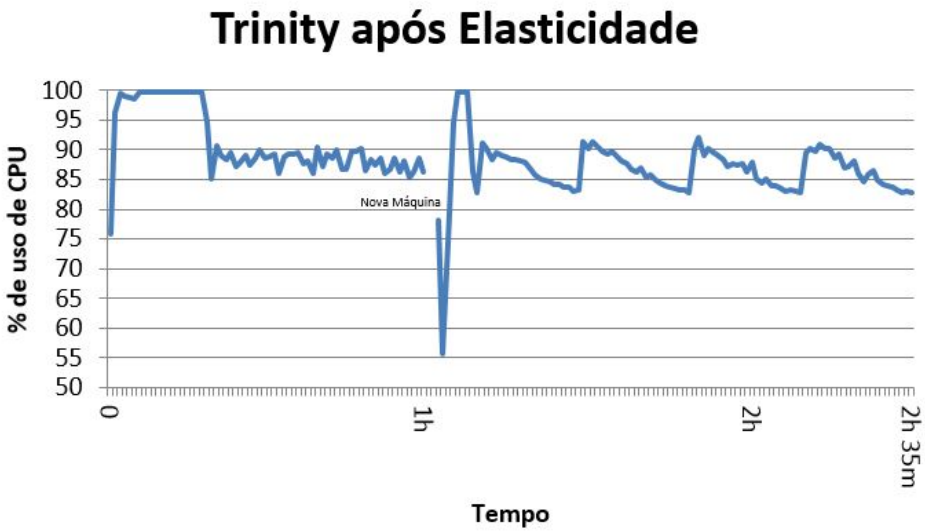

Figura 5.27: Elasticidade Vertical por Substituição - Execução da Elasticidade.

Dessa forma, é possível notar, que o procedimento de elasticidade diminuiu o tempo total de execução desta tarefa, de 3 horas e 40 minutos, para 2 horas e 35 minutos, tendo um ganho de aproximadamente 30%.

5.5 Considerações Finais

Este capítulo abordou em detalhes a proposta de um controlador de elasticidade para nuvens federadas. Outro tópico abordado foi o estudo de caso na plataforma de nuvens Federadas BioNimbuZ, mostrando como esse controlador foi implementado na plataforma. Também foram feitos testes para avaliar o controlador proposto. O Capítulo 6 apresentará as conclusões e alguns trabalhos futuros.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um controlador de elasticidade para um ambiente de nuvens federadas, que suporte o provisionamento/desprovisionamento automático e sob demanda de máquinas virtuais, bem como a elasticidade horizontal e vertical dos recursos alocados.

Com base nos testes realizados, o controlador de elasticidade proposto mostrou que ele á capaz de garantir elasticidade em ambientes de nuvens federadas, proporcionando melhor eficiência para as aplicações executadas neste contexto. O controlador atuou diretamente na criação e na exclusão de máquinas virtuais. Para isto, o controlador proposto levou em consideração os parâmetros informados pelo usuário, tais como tempo máximo de duração da execução e uma definição entre uma execução de baixo custo ou de alto desempenho.

A arquitetura do controlador de elasticidade possui três módulos, os quais são o *Menager*, o Controlador de Elasticidade Horizontal e o Controlador de Elasticidade Vertical. O *Menager* é o responsável por receber os eventos de disparo da elasticidade através do monitoramento das VMs, e por decidir entre elasticidade horizontal ou vertical através de uma verificação de qual máquina está sobrecarregada.

O Controlador de Elasticidade Horizontal é o responsável por criar novas máquinas virtuais. Essas máquinas podem ser criadas para executar alguma tarefa nova, que foi requisitada pelo cliente da federação; Ou para aumentar a capacidade de execução das máquinas já provisionadas. Por outro lado, o Controlador de Elasticidade Vertical pode executar a expansão de uma máquina virtual de duas maneiras, por redimensionamento ou por substituição.

Para validar o controlador proposto, ele foi integrado a plataforma de nuvens federadas BioNimbuZ, dando para a plataforma a capacidade de executar um rápido provisionamento no ambiente de nuvem federada, com tempo de criação de no máximo 4 min e de deleção com menos de 10 segundos. Por fim, foram executados alguns testes que demonstraram com um *workflow* real de Bioinformática, a capacidade do controlador de fazer o provisionamento e a elasticidade para uma aplicação real, sendo divididos em elasticidade

horizontal e vertical. Nos testes de elasticidade vertical, optou-se em executá-los no servidor web do BioNimbuZ, dando a capacidade elástica para a própria aplicação, já para os testes de elasticidade horizontal nas própria aplicações executadas pelo BioNimbuZ, e tendo um ganho de aproximadamente 30% quando a elasticidade é ativada.

Como trabalho futuro é sugerido que o controlador funcione de maneira pró-ativa, de tal forma que o mesmo seja capaz de prever e executar o trabalho de alocar/desalocar máquinas virtuais, antes do ambiente de nuvens saturar ou ficar ocioso. Além disso, espera-se implementar futuramente a migração de máquinas virtuais em ambientes de nuvem federada.

Referências

[1] Azevedo, D. R., and Freitas Júnior, T. B. Uma nova política de armazenamento para a plataforma BioNimbuZ de nuvem federada. `http://bdm.unb.br/handle/10483/11141`, 2015. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 29, 30, 37

[2] Badger, L., Grance, T., Patt-Corner, R., and Voas, J. Draft cloud computing synopsis and recommendations. *Recommendations of the National Institute of Standards and Technology* (2011). 6

[3] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review 37*, 5 (2003), 164–177. 26, 27

[4] Barreiros Júnior, W. O. Escalonador de tarefas para o plataforma de nuvens federadas BioNimbuZ usando beam search iterativo multiobjetivo. `http://bdm.unb.br/handle/10483/13146`, 2016. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 29, 30, 37, 60

[5] Beernaert, L., Matos, M., Vilaça, R., and Oliveira, R. Automatic elasticity in openstack. In *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management* (2012), ACM, p. 2. 24, 26, 27

[6] Bittman, T. The evolution of the cloud computing market. gartner blog network, 2008. 12

[7] Borges, C. A. L. Escalonamento de tarefas em uma infraestrutura de computação em nuvem federada para aplicações em bioinformática. 900. 37

[8] Buyya, R., Broberg, J., and Goscinski, A. M. *Cloud computing: Principles and paradigms*, vol. 87. John Wiley & Sons, 2010. 1, 5

[9] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience 41*, 1 (2011), 23–50. 25

[10] Calheiros, R. N., Vecchiola, C., Karunamoorthy, D., and Buyya, R. The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems 28*, 6 (2012), 861–870. 26, 27

[11] Celesti, A., Tusa, F., Villari, M., and Puliafito, A. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (2010), IEEE, pp. 337–345. 12

[12] Chapman, C., Emmerich, W., Marquez, F. G., Clayman, S., and Galis, A. Elastic service management in computational clouds. In *12th IEEE/IFIP NOMS2010/International Workshop on Cloud Management (CloudMan 2010)* (2010), pp. 19–23. 26, 27

[13] Chieu, T. C., Mohindra, A., Karve, A. A., and Segal, A. Dynamic scaling of web applications in a virtualized cloud computing environment. In *E-Business Engineering, 2009. ICEBE'09. IEEE International Conference on* (2009), IEEE, pp. 281–286. 1

[14] Chirigati, F. S. Computação em nuvem. *Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ* (2009). 11

[15] Cloudsigma. `http://www.cloudsigma.com/`. Acessado online em 4 de fevereiro de 2016. 21

[16] Costa, H. H. d. P. M. Controle de acesso na plataforma de nuvem federada bionimbuz. `http://bdm.unb.br/handle/10483/13199`, 2015. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 29, 38

[17] Coutinho, E. F., de Carvalho Sousa, F. R., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. Elasticity in cloud computing: a survey. *annales des télécommunications 70*, 7-8 (2015), 289–309. x, 16, 17

[18] Crockford, D. Json. `http://json.org/`. Acessado online em 4 de fevereiro de 2016. 41

[19] Dawoud, W., Takouna, I., and Meinel, C. Elastic vm for cloud resources provisioning optimization. In *International Conference on Advances in Computing and Communications* (2011), Springer, pp. 431–445. 50

[20] Dawoud, W., Takouna, I., and Meinel, C. Elastic virtual machine for fine-grained cloud resource provisioning. In *Global Trends in Computing and Communication Systems*. Springer, 2012, pp. 11–25. 27, 50

[21] Deelman, E., Gannon, D., Shields, M., and Taylor, I. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems 25*, 5 (2009), 528–540. 2

[22] Diaz-Montes, J., AbdelBaky, M., Zou, M., and Parashar, M. Cometcloud: Enabling software-defined federations for end-to-end application workflows. *IEEE Internet Computing 19*, 1 (2015), 69–73. 12, 13

[23] Dustin oWens, B. Securing elasticity in the cloud. *Communications of the ACM 53*, 6 (2010). 24

[24] Factor, M., Meth, K., Naor, D., Rodeh, O., and Satran, J. Object storage: the future building block for storage systems. In *2005 IEEE International Symposium on Mass Storage Systems and Technology* (June 2005), pp. 119–123. 37

[25] Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., and Lynn, T. A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing 74*, 10 (2014), 2918–2933. 57

[26] Feo, T. A., and Resende, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization 6*, 2 (1995), 109–133. 40

[27] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext Transfer Protocol–http/1.1 , Request For Coments (RFC 2616). `http://tools.ietf.org/pdf/rfc2616.pdf`. Acessado online em 4 de fevereiro de 2016. 41

[28] Fielding, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. 35

[29] Foster, I., Zhao, Y., Raicu, I., and Lu, S. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08* (2008), Ieee, pp. 1–10. x, 7, 8

[30] Foundation, A. S. Apache avro). `http://avro.apache.org/`. Acessado online em 4 de fevereiro de 2016. 41

[31] Foundation, T. A. S. Apache zookeeper. `http://zookeeper.apache.org/`. Acessado online em 4 de fevereiro de 2016. 30, 41, 62

[32] Galante, G. *Explorando elasticidade em nivel de programacao*. PhD thesis, Universidade Federal do Paraná, 2014. x, 21, 22, 24, 25

[33] Galante, G., and de Bona, L. C. E. A survey on cloud computing elasticity. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on* (2012), IEEE, pp. 263–270. x, 16, 18

[34] Gogrid. `http://www.gogrid.com/`. Acessado online em 4 de fevereiro de 2016. 21

[35] Google. Cloud storage. `https://cloud.google.com/storage/`. Acessado online em 10 de janeiro de 2017. 37

[36] Google. Google compute engine. `https://cloud.google.com/compute`. Acessado online em 4 de fevereiro de 2016. 12, 59, 62

[37] Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., Couger, M. B., Eccles, D., Li, B., Lieber, M., et al. De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. *Nature protocols 8*, 8 (2013), 1494–1512. 66

[38] Han, R., Guo, L., Ghanem, M. M., and Guo, Y. Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on* (2012), IEEE, pp. 644–651. 24

[39] Heath, A. P., Greenway, M., Powell, R., Spring, J., Suarez, R., Hanley, D., Bandlamudi, C., McNerney, M. E., White, K. P., and Grossman, R. L. Bionimbus: a cloud for managing, analyzing and sharing large genomics datasets. *Journal of the American Medical Informatics Association 21*, 6 (2014), 969–975. 12

[40] Herbst, N. R., Kounev, S., and Reussner, R. H. Elasticity in cloud computing: What it is, and what it is not. In *ICAC* (2013), pp. 23–27. 16

[41] Java. `https://www.java.com/`. Acessado online em 4 de fevereiro de 2016. 21

[42] Keahey, K. Nimbus: open source infrastructure-as-a-service cloud computing software. In *Workshop on adapting applications and computing services to multi-core and virtualization, CERN, Switzerland* (2009). 26

[43] Konstanteli, K., Cucinotta, T., Psychas, K., and Varvarigou, T. Admission control for elastic cloud services. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on* (2012), IEEE, pp. 41–48. 27

[44] Kumar, R., Jain, K., Maharwal, H., Jain, N., and Dadhich, A. Apache cloudstack: Open source infrastructure as a service cloud computing platform. *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science* (2014), 111–116. 50

[45] Kwei-Jay, L., and Gannon, J. Atomic remote procedure call. *IEEE Transactions on Software Engineering SE-11*, 10 (1985), 1126–1135. 30

[46] Leite, A. F. *A user-centered and autonomic multi-cloud architecture for high performance computing applications*. PhD thesis, Universidade de Brasília, 2014. 26, 27

[47] Lenk, A., Klems, M., Nimis, J., Tai, S., and Sandholm, T. What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing* (2009), IEEE Computer Society, pp. 23–31. 7

[48] Lim, H. C., Babu, S., Chase, J. S., and Parekh, S. S. Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds* (2009), ACM, pp. 13–18. 25

[49] Lima, D., Moura, B., Ribeiro, E. ; Aráujo, A. P. F., Walter, M. E., Holanda, M. T., and Oliveira, G. A storage policy for a hybrid federated cloud platform executing bioinformatics applications. *C4BIE 2014: Cloud for Business, Industry and Enterprises* (2014). Proceedings of the 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014). 12, 29, 30, 42

[50] LLC, A. W. S. Amazon elastic compute cloud (EC2). `http://aws.amazon.com/pt/ec2/`. Acessado online em 4 de fevereiro de 2016. 12, 21, 25, 41, 59, 62

[51] LLC, A. W. S. Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3/. Acessado online em 4 de fevereiro de 2016. 37, 41

[52] Marshall, P., Keahey, K., and Freeman, T. Elastic site: Using clouds to elastically extend site resources. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (2010), IEEE Computer Society, pp. 43–52. 26, 27

[53] Mell, P., and Grance, T. Draft nist working definition of cloud computing. *Referenced on June. 3rd 15* (2009). 1, 5, 6, 9, 10, 11, 16

[54] Microsoft. Microsoft azure. `http://azure.microsoft.com/pt-br/`. Acessado online em 4 de fevereiro de 2016. 12

[55] Milojičić, D., Llorente, I. M., and Montero, R. S. Opennebula: A cloud management tool. *IEEE Internet Computing 15*, 2 (2011), 11–14. 26

[56] Moore, L. R., Bean, K., and Ellahi, T. Transforming reactive auto-scaling into proactive auto-scaling. In *Proceedings of the 3rd International Workshop on Cloud Data and Platforms* (2013), ACM, pp. 7–12. 25

[57] Moura, B. R., and Bacelar, D. L. Política para armazenamento de arquivos no zoonimbus, 2013. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 29, 37

[58] Naskos, A., Gounaris, A., and Sioutas, S. Cloud elasticity: A survey. In *Algorithmic Aspects of Cloud Computing*. Springer, 2016, pp. 151–167. x, 16, 19

[59] Nguyen, H., Shen, Z., Gu, X., Subbiah, S., and Wilkes, J. Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)* (2013), pp. 69–82. 24

[60] Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. An algebraic approach for data-centric scientific workflows. *Proc. of VLDB Endowment 4*, 12 (2011), 1328–1339. 2

[61] Oliveira, G. S. S. d. Acosched: um escalonador para o ambiente de nuvem federada Zoonimbus, 2013. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 37

[62] Owens, D., and Americas, B. Securing elasticity in the cloud. *Communications of the ACM 53*, 6 (2010). 16

[63] Paula, R. d. Proveniência de dados em workflows de bioinformática. http://repositorio.unb.br/handle/10482/12699, 2013. x, 14

[64] Petcu, D., Di Martino, B., Venticinque, S., Rak, M., Máhr, T., Lopez, G. E., Brito, F., Cossu, R., Stopar, M., Šperka, S., et al. Experiences in building a mosaic of clouds. *Journal of Cloud Computing: Advances, Systems and Applications 2*, 1 (2013), 12. 12, 13

[65] Php. http://www.php.net/. Acessado online em 4 de fevereiro de 2016. 21

[66] Profitbricks. https://www.profitbricks.com/. Acessado online em 4 de fevereiro de 2016. 21

[67] Rackspace. http://www.rackspace.com. Acessado online em 4 de fevereiro de 2016. 21

[68] Ramos, V. A. Um sistema gerenciador de workflows científicos para a plataforma de nuvens federadas BioNimbuZ. http://bdm.unb.br/handle/10483/13145, 2016. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. x, 29, 30, 31, 32, 33, 34, 42

[69] RightScale. Rightscale. http://www.rightscale.com/. Acessado online em 4 de fevereiro de 2016. 25

[70] Rimal, B. P., Choi, E., and Lumb, I. A taxonomy and survey of cloud computing systems. *INC, IMS and IDC* (2009), 44–51. 9

[71] Ruby. https://www.ruby-lang.org/. Acessado online em 4 de fevereiro de 2016. 21

[72] Saldanha, H., Araújo, A., Borges, C., Ribeiro, E., Setubal, J. C., Walter, M. E., Holanda, M., Gallon, R., and Togawa, R. *Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows.* INTECH Open Access Publisher, 2012. 2

[73] Saldanha, H. V. Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática. Master's thesis, Departamento de Ciência de Computação, Universidade de Brasília, 2012. 2, 29, 30, 50

[74] Santos, L. F. N. Novo serviço de armazenamento na plataforma de nuvem federada BioNimbuZ. http://bdm.unb.br/handle/, 2016. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 29, 30, 37

[75] Sefraoui, O., Aissaoui, M., and Eleuldj, M. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications 55*, 3 (2012). 26

[76] Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on* (2011), IEEE, pp. 559–570. 24, 27

[77] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (2011), ACM, p. 5. 24

[78] Sluzala, G. F., and de Araújo, A. P. F. Sistema de tarifação pay-per-use para nuvens federadas. 38

[79] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Transactions on 11*, 1 (2003), 17–32. 29

[80] Taurion, C. Cloud computing: computação em nuvem: transformando o mundo da tecnologia da informação. *Rio de Janeiro: Brasport 2*, 2 (2009), 2–2. 11

[81] Van Der Aalst, W., and Van Hee, K. M. *Workflow management: models, methods, and systems*. MIT press, 2004. 13

[82] Van Steen, M., and Tanenbaum, A. *Distributed Systems: Principles and Paradigms*. Prentice Hall; 2 edition, 2006. 30

[83] Vaquero, L. M., Rodero-Merino, L., and Buyya, R. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review 41*, 1 (2011), 45–52. 22, 24

[84] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review 39*, 1 (2008), 50–55. x, 1, 5, 7, 8

[85] Yang, C., and Huang, Q. *Spatial cloud computing: a practical approach*. CRC Press, 2013. 50

[86] Youseff, L., Butrico, M., and Da Silva, D. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE'08* (2008), IEEE, pp. 1–10. 7

[87] Yu, J., and Buyya, R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Sci. Program. 14*, 3,4 (Dec. 2006), 217–230. 14