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

Predição de Tempo e Dimensionamento de Recursos para Workflows Científicos em Nuvens Federadas

Michel Junio Ferreira Rosa

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Profa. Dra. Aletéia Patrícia Favacho de Araújo

Brasília
2017

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

Rosa, Michel Junio Ferreira
RR788p Predição de Tempo e Dimensionamento de Recursos para Workflows Científicos em Nuvens Federadas / Michel Junio Ferreira Rosa; orientador Aletéia Patrícia Favacho de Araújo. -- Brasília, 2017.
97 p.

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

1. Federação de Nuvem. 2. Predição de Recursos. 3. GRASP. 4. Bioinformática. 5. Workflow. I. Araújo, Aletéia Patrícia Favacho de, orient. II. Título.

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

Predição de Tempo e Dimensionamento de Recursos para Workflows Científicos em Nuvens Federadas

Michel Junio Ferreira Rosa

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Profa. Dra. Aletéia Patrícia Favacho de Araújo (Orientador)
CIC/UnB

Profa. Dra. Lúcia M. A. Drummond
IC/UFF

Profa. Dra. Maria Emília Machado Telles Walter
CIC/UnB

Profa. Dra. Célia Ghedini Ralha
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 24 de fevereiro de 2017

Dedicatória

Dedico esta dissertação aos meus pais Maria José e Ronaldo, pois, sem eles, meus objetivos jamais teriam sido alcançados.

Dedico, também, à minha orientadora Prof^{a} Dr^{a} Aletéia Patrícia, pela confiança, paciência, amizade e excelente orientação.

Sem o apoio de ambos, este trabalho jamais teria sido concluído.

Agradecimentos

Agradeço primeiramente ao meu DEUS, por estar sempre ao meu lado.

Ao meus queridos pais, que sempre me apoiaram a nunca desistir e sempre persistir, para que nosso objetivo fosse atingido com êxito.

A Profa. Dra. Aletéia Patrícia pela orientação e por sempre ter acreditado em mim e no potencial desse trabalho. E também por aqueles momentos de desespero de não conseguir progredir.

Agradeço à equipe BioNimbuZ, que por trabalhar em um só projeto, auxiliaram na madrugada durante o desenvolvimento deste trabalho, além de persistirmos para a conclusão do mesmo.

Agradeço aos meus colegas de trabalho no IPEA, Rodrigo Santos, Gustavo Maia e Gustavo Basso, pela ajuda no aprendizado dos métodos estatísticos e, também, pelas horas que deixei vocês sem dormir.

Agradeço aos meus colegas do Programa de Pós-graduação em Informática – PPGInf, pela amizade que adquirimos durante esses dois anos de trabalho.

Resumo

A computação em nuvem concebeu um modelo computacional interessante, que fornece um conjunto de recursos tais como armazenamento, banco de dados e poder de processamento, todos disponibilizados como serviços. Recentemente, o conceito de computação em nuvem se estendeu para a computação em nuvens federadas, nas quais diferentes provedores se interconectam para disponibilizarem mais recursos de maneira integrada e transparente ao usuário final. Assim, o uso de plataformas de nuvem tem sido amplamente incentivado em aplicações que demandam muito poder de processamento e/ou armazenamento, como por exemplo os *workflows* de bioinformática. Todavia, os usuários que operam tais *workflows* se deparam com uma variedade e quantidade muito grande de recursos disponíveis, sendo difícil a escolha correta dos mesmos para um determinado *workflow*. Esse dimensionamento está longe de ser trivial e, para tratar desse problema, este trabalho propõe uma abordagem chamada sPCR (Serviço de Predição de Custos e Recursos Computacionais), que mescla a metaheurísticas GRASP e o método de regressão linear múltipla, com o objetivo de dimensionar os recursos para os usuários de forma transparente, ao informar o custo financeiro e o tempo de execução antes mesmo de iniciar o *workflow*. Além disso, o sPCR permite que o usuário possa interagir e escolher entre execuções de alto desempenho, de baixo orçamento, ou definir o quanto quer pagar e em quanto tempo quer a finalização do *workflow*, tudo de forma automática e transparente. Os resultados mostram a adequação do sPCR para estimar os recursos, custos e tempos de execução dos *workflows* testados.

Palavras-chave: Federação de Nuvem, Predição de Recursos, GRASP, Bioinformática, Workflow.

Abstract

Cloud computing provides an interesting computational model which provides a set of features, such as storage, database, and processing power, all made available as services. Recently, the concept of cloud computing has been extended to cloud federations in which different providers interconnect to provide more resources to the end user in an integrated and transparent way. The use of cloud platforms has been widely encouraged in applications that require a lot of processing and / or storage power, such as bioinformatics workflows. However, users who operate such workflows are faced with a very large variety and quantity of available resources, making it difficult to choose the correct ones for a certain workflow. This design is far from trivial and, in order to address this problem, this work proposes an approach called sPCR (Service of Costs and Computational Resources Prediction) which merges GRASP metaheuristics and the multiple linear regression method, with the purpose of transparently measuring resources for users by reporting the financial cost and runtime before even starting the workflow. In addition, sPCR allows the user to interact and choose between high-performance, low-budget runs, or set how much to pay and how long to finish the workflow, all automatically and transparently. The results show the adequacy of the sPCR to estimate the resources, costs and execution times of the tested workflows.

Keywords: Cloud federation, Prediction Resources, GRASP, Bioinformatics, Workflow.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A computação está convergindo para um modelo que consiste em serviços que sejam entregues de um modo semelhante aos serviços tradicionais, tais como água, telefonia, eletricidade, gás e vários outros. A exploração destes serviços é apoiada em um modelo de pagamento baseado no seu uso, no qual as infraestruturas permitem a entrega dos serviços sempre que seus usuários necessitarem, sem que estes se preocupem com as etapas do processo. Alguns paradigmas computacionais prometem entregar a visão de computação utilitária, e estas incluem o *Grid Computing* [105] e, a mais recente, *Cloud Computing* [13].

A última simboliza a infraestrutura como uma "nuvem", na qual quaisquer usuários e corporações possam acessar serviços sob demanda sem a limitação geográfica, desde que tenham acesso à Internet. Desse modo, há uma evolução rápida da computação no sentido de desenvolver serviços para milhões de usuários consumirem os recursos de forma a atenderem suas necessidades, e sob demanda [13].

Logo, o surgimento da computação em nuvem busca a redução dos custos computacionais, o aumento da confiabilidade, a flexibilidade e o provisionamento de recursos e serviços sob demanda, no qual recursos e serviços estejam disponíveis de acordo com as necessidades de seus usuários, de forma transparente e dinâmica. Assim, na computação em nuvem, o processamento, a manipulação de dados e o armazenamento são vistos como serviços [12].

Além disso, a demanda por recursos de computação de alto desempenho não permeia somente grandes corporações, mas também pesquisadores que atuam com experimentos complexos, tais como *workflows* de Bioinformática, que consomem e produzem enormes conjuntos de dados e, inerentemente, demandam significativas quantidades de recursos computacionais [79].

Dessa forma, o poder computacional entregue pela computação em nuvem é muito importante para que os *workflows* científicos alcancem a automação durante seu proces-

samento. A execução de *workflows* científicos é computacionalmente intensiva e demanda alta disponibilidade de recursos computacionais confiáveis. Um *workflow* científico pode ser definido como a caracterização formal de um processo científico que consiste em uma sequência ordenada de atividades tais que, as saídas de um processo constituem-se em entradas para o próximo processo [25].

Assim, o gerenciamento eficiente de *workflows* científicos flexíveis, que busca a garantia da disponibilidade de recursos e a realização de seus objetivos, é uma tarefa árdua, na qual muitas vezes a disponibilidade de recursos pode decidir entre uma execução bem ou mal sucedida de um *workflow* científico complexo e caro computacionalmente [26].

Por outro lado, na busca por melhor atender os consumidores dos serviços de nuvem, surgiu a ideia de integrá-las, aumentando os recursos disponíveis para a disponibilização dos serviços computacionais. Dessa integração de nuvens emergiu a chamada federação de nuvens, que são conjuntos de nuvens que possuem todos os seus recursos gerenciados por meio de uma interface conectada a todas elas, de forma que, se uma nuvem não tiver recursos para atender determinada demanda, outras, que estejam com recursos disponíveis no momento, possam ser integradas, atendendo à demanda computacional [95].

Entretanto, todos esses recursos disponibilizados pela federação de nuvens devem ser escolhidos de forma eficiente, com o objetivo de suprir as necessidades dos usuários e suas aplicações, evitando problemas de *overprovisioning* ou *underprovisioning*. O primeiro indica uma extrapolação da capacidade dos recursos, ou seja, os recursos estão sendo utilizados acima dos limites estabelecidos, e o segundo indica ociosidade dos mesmos. Entretanto, a escolha desses recursos é, na maioria das vezes, incerta por parte dos usuários, uma vez que não é trivial prever as demandas de recursos durante a execução dos *workflows*, já que os recursos estão atrelados a vários parâmetros, tais como tempo de execução, processamento, armazenamento, rede e memória [16].

Por isso, ter um serviço de predição que possa auxiliar o usuário na escolha correta dos recursos a serem demandados é importante, pois a escolha eficiente traz melhorias nos custos e na redução de falhas durante as execuções.

Neste contexto, este trabalho propõe um serviço de provisionamento para ambiente de nuvens federadas, com o intuito de prever o tempo e o custo de cada *workflow*, por meio da utilização de uma base de dados histórica, concebida pelo monitoramento das execuções. Para validar o serviço de predição proposto para nuvem federada, optou-se por integrá-lo a uma plataforma real de federação. A plataforma escolhida foi o BioNimbuZ [5], [6], [65], [87], [93], [94], [97]. O BioNimbuZ é uma plataforma de federação que garante a integração entre diferentes nuvens de maneira simples, dinâmica e transparente.

Assim sendo, este trabalho utiliza como estudo de caso a plataforma de federação BioNimbuZ para implementar o Serviço de Predição de Custos e Recursos computacionais

– sPCR, com o intuito de auxiliar na predição de tempo e no dimensionamento dos recursos que serão utilizados durante a execução do *workflow*. No serviço proposto, é possível escolher entre execuções de baixo custo orçamentário ou de alto desempenho. Assim, a primeira escolha indica seleção de recursos mais baratos, implicando, assim, em maior tempo de duração. Já a segunda opção, indica execuções de alto custo, com recursos mais caros, resultando em um menor tempo de execução.

1.1 Motivação

A plataforma de federação de nuvens tem demonstrado sua capacidade de integrar diferentes infraestruturas, que possibilitam uma flexibilidade na escolha de provedores e uma ilusão de que os recursos computacionais são ilimitados. Porém, como a federação disponibiliza um leque grande de opções dos recursos a serem escolhidos para a execução de uma tarefa, há a necessidade de um serviço que auxilie o usuário final na melhor seleção de recursos a ser feita.

Neste cenário, a predição auxilia na tomada de decisão do usuário com o intuito de garantir execuções satisfatórias com custos adequados e, assim, evitar possíveis prejuízos com erros durante as execuções, causados por recursos insuficientes, ou até mesmo pela interrupção do usuário, caso esta não finalize no tempo determinado.

Dessa forma, o serviço de predição proposto neste trabalho tem por finalidade estimar recursos e auxiliar o usuário em uma decisão entre custo e *performance*, na qual será possível escolher entre uma execução de baixo orçamento e com maior tempo de execução, ou execuções rápidas com custos mais altos, ambas de forma transparente.

1.2 Problema

O leque de recursos disponíveis pela federação de nuvens possibilita a execução de *workflows* em bioinformática de forma satisfatória. Entretanto, a escolha ideal desses recursos não é trivial para os usuários que operam tais *workflows*. Isso se dá porque existem várias possibilidades de alocação de recursos, implicando em diferentes custos financeiros. Assim sendo, a falta de estimativas de demanda de recursos computacionais necessários para a execução de *workflows* causa prejuízo financeiro e de produtividade a usuários que não parametrizam recursos computacionais adequados para a execução de seus *workflows* científicos. Assim, as execuções se tornam uma caixa preta, com o risco de resultados imprecisos e de alto custo [16].

Neste cenário, um serviço de predição é fundamental porque auxilia os usuários na escolha dos recursos de forma eficiente, evitando prejuízos no projeto.

1.3 Objetivos

O objetivo principal deste trabalho é propor um serviço de predição para estimar custos financeiros, tempo de execução e recursos computacionais necessários para a execução de *workflows* científicos em ambiente de nuvens federadas, auxiliando usuários na tomada de decisões de forma simples, automática e transparente.

Para cumprir o objetivo principal, este trabalho tem os seguintes objetivos específicos:

- Criar históricos de execuções de diferentes *workflows* de Bioinformática na plataforma de federação BioNimbuZ;
- Propor um modelo para a realização da predição em um ambiente de nuvem federada;
- Realizar um estudo de caso, implementando o serviço de predição integrado aos serviços do ambiente de nuvem federada BioNimbuZ.
- Avaliar a qualidade do serviço de predição proposto.

1.4 Estrutura do Trabalho

Este trabalho contém, além deste capítulo introdutório, mais seis capítulo. No Capítulo 2, serão tratados os projetos com *workflows* de bioinformática, suas características, softwares e exemplos.

No Capítulo 3, será apresentada a computação em nuvem, suas características, sua arquitetura e seu modo de funcionamento. Além disso, será caracterizada a plataforma de nuvem federada, apresentada a plataforma BioNimbuZ, mostrando sua estrutura, sua organização, seus objetivos e as modificações sofridas desde a sua proposta original.

No Capítulo 4, serão apresentados os conceitos de predição de recursos computacionais, assim como as técnicas empregadas na literatura. Além disso, são abordados os principais trabalhos relacionados ao tema.

No Capítulo 5, será descrito o serviço de predição proposto neste trabalho. Para isso, serão apresentados a visão geral de cada etapa, as técnicas utilizadas para predição e o fluxo de execução. Neste capítulo, também serão apresentados os testes realizados, e os resultados obtidos com o serviço de predição proposto.

O Capítulo 6 apresenta a conclusão deste trabalho e alguns trabalhos futuros.

Capítulo 2

Workflow de Bioinformática

Neste capítulo, serão tratados os conceitos de *workflows* de Bioinformática, suas principais características, seus softwares e alguns exemplos. A Seção 2.1 apresentará os conceitos de *workflows* científicos a partir da Bioinformática. Na Seção 2.2 serão apresentadas as principais fases dos *workflows* em bioinformática e, por último, na Seção 2.3 serão apresentados os programas envolvidos nessas fases.

2.1 *Workflows* Científicos

Workflows científicos são conjuntos de tarefas a serem executadas para alcançar determinado objetivo [86]. Ele é composto por um conjunto de dados de entrada e de saída e por ferramentas, que são ordenadas de forma a atingirem o objetivo almejado [71]. Um *workflow* refere-se à automação de processos tal, que documentos, informações ou tarefas são entregues entre os membros, segundo um conjunto pré-definido de regras, para alcançar ou cooperar um objetivo global de um negócio. Ainda que um *workflow* possa ser manualmente organizado, na prática, em sua grande maioria eles são estruturados em um contexto de sistema de informação que provê apoio automatizado aos processos [49].

Workflows de Bioinformática estão relacionados ao processamento de dados biológicos e, em geral, aos processos de análises computacionais de dados gerados por equipamentos de sequenciamento de DNA (Ácido Desoxirribonucleico) ou RNA (Ácido Ribonucleico) [82]. Esses processos, inicialmente, visam descobrir qual é a sequência de bases que forma cada fragmento de DNA ou RNA de um determinado organismo que está sendo investigado. Assim, define-se como projeto genoma aqueles que consistem em pesquisar DNA genômico (cromossomos), enquanto que projetos transcritoma consistem no estudo dos transcritos[1] (RNA). A partir desses fragmentos, tanto de DNA, quanto de RNA, vários

[1] O transcrito ocorre quando uma enzima separa uma das fitas da hélice do DNA, que é copiada formando o RNA [45]

processos computacionais podem ser executados de acordo com os objetivos do projeto.

De forma geral, os projetos genoma e transcritoma possuem suporte computacional, nos quais são projetados *workflows* que transformam fragmentos de entrada, de forma a extrairem informações como funções biológicas e localização dentro da célula. O exemplo da Figura 2.1 mostra um *workflow* de três fases: filtragem, mapeamento/montagem e análise, descritas em detalhes na próxima seção.

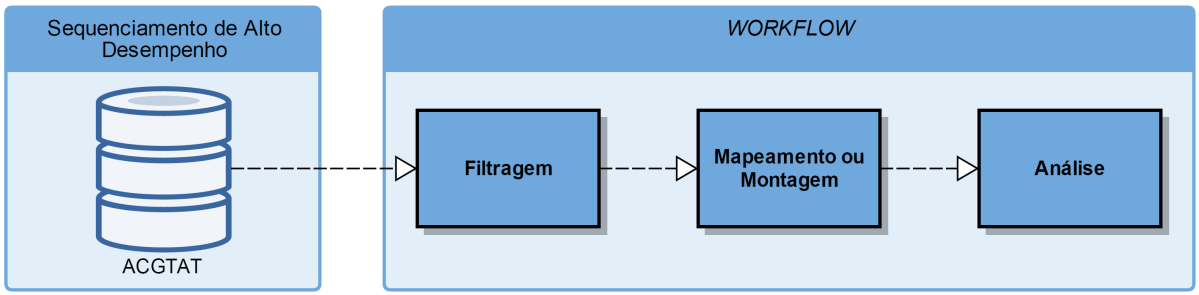


Figura 2.1: Exemplo de *Workflow* para Projetos Genoma e Transcritoma [82].

2.2 Fases de um *Workflow*

Nos experimentos realizados pelos biólogos, é comum executar, como apresentado na Figura 2.1, *workflows* com três fases. Inicialmente, os biólogos realizam processos laboratoriais de coleta e de replicação do material biológico. Em seguida, é realizado o sequenciamento de pequenas porções de DNA ou RNA (de acordo com o projeto), que são denominadas *reads*. O tamanho de cada *read* obtida depende do equipamento utilizado. Essas *reads* são formadas por bases nitrogenadas (Adenina, Citosina, Guanina e Timina) que formam as cadeias do DNA [64]. Geralmente, as bases das *reads* são associadas a um indicador de qualidade, que, durante o processo de filtragem, é usado para eliminar as *reads* de qualidade inferior, ou parte delas, com um certo grau de confiabilidade mínima.

A etapa de filtragem pode também detectar contaminantes e/ou erros laboratoriais [82]. Dessa forma, a fase de filtragem é uma tarefa simples que pode ser realizada localmente por *scripts* desenvolvidos pelo usuário, ou por programas mais completos com diversas ferramentas, como o pacote *FASTX-Toolkit* [58]. Esse pacote fornece ferramentas para manipulações de dados do tipo *FASTA* [83] e *FASTQ* [18], frequentemente, utilizados para armazenar *reads*. Dentre esses tratamentos, encontram-se filtros e conversores de formato, úteis nesta fase. Os arquivos utilizados e gerados nesta fase podem somar *gigabytes* de dados, incluído as *reads* e as informações adicionais, tais como o identificador

de cada sequência e a qualidade de cada base mapeada. Assim, estes dados constituem, em geral, grande parte do espaço em disco utilizado durante o experimento [82].

Na fase de mapeamento, o objetivo é encontrar, em um genoma de referência, o alinhamento para o maior número de *reads* filtradas. Assim, é possível agrupá-las em fragmentos maiores que são explorados *a posteriori*. Essa técnica é empregada em organismos que possuem seu DNA completamente (ou quase) sequenciado. A Figura 2.2 exemplifica um mapeamento, no qual é utilizado um genoma de referência conhecido em relação ao qual as *reads* filtradas são alinhadas.

Figura 2.2: Exemplo do Processo de Mapeamento, adaptado de [82].

Caso o genoma de referência seja desconhecido, é realizado o processo de montagem. Nele, o alinhamento é realizado entre as *reads* que geram sequências denominadas *contigs* [96]. A Figura 2.3 apresenta diversas *reads* que são alinhadas, e o resultado é um *contig* de maior tamanho.

Figura 2.3: Exemplo do Processo de Montagem, adaptado de [82].

Para o mapeamento de transcritos, há complexidade quando se tem *splice junctions* nas *reads* sequenciadas. A Figura 2.4 exemplifica o problema encontrado no alinhamento de sequencias de transcritos com *splice junctions*. Na Figura 2.4 (A), o processo de transcrição ocorre somente onde os *exons* são copiados para formar o transcrito. Logo, na Figura 2.4 (B), quando há o alinhamento da *read* com o transcrito contra o genoma de referência, não é encontrada a posição adequada porque existe um *intron* [96] no genoma

de referência. Esse fato faz com que várias *reads* sejam descartadas por não encontrarem um alinhamento adequado [76].

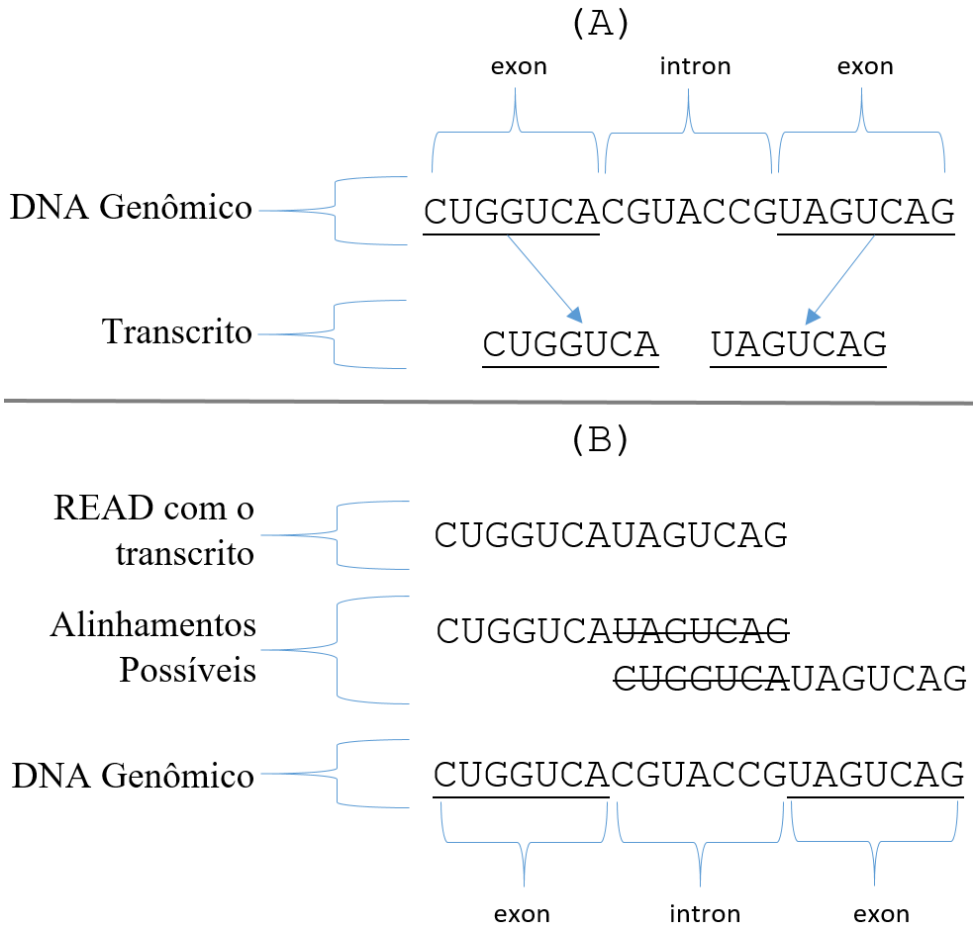


Figura 2.4: Exemplo do Processo de Transcrição (A), e do Problema de Alinhamento com *Splice Junctions* (B), adaptado de [82].

O processo de mapeamento, como pode ser visto na Figura 2.4, requer algoritmos mais sofisticados em relação ao mapeamento. Assim, são encontrados na literatura inúmeros softwares que oferecem diferentes recursos e especialidades. Um exemplo é o *Bowtie* [61], que apresenta resultados confiáveis para *reads* pequenas. Além disso, ele é desenvolvido para otimizar o uso eficiente de memória e utilização de múltiplos processadores, que são duas características importantes para o processo de alinhamento de grandes quantidades de *reads*.

Há também o *TopHat* [103], que utiliza o *Bowtie* para o alinhamento das sequências e possui recursos de tratamento de *splice junctions*, que identificam as uniões *exon-exon* [96], e criam um banco de dados com essas informações para serem utilizadas, *a posteriori*, no alinhamento. Existem também softwares específicos em alinhamento de sequência de

RNA, como por exemplo o CUFFLINKS [61] e o Clustal [48], sendo este último utilizado para o alinhamento múltiplo de sequências.

A última fase do *workflow* é a de análise, na qual grandes porções do DNA ou RNA sequenciadas, obtidas das etapas de mapeamento, podem ser analisadas a partir de diferentes processos, dependendo do objetivo do experimento. É possível citar como exemplo a expressão diferencial de humanos suscetíveis, ou não, à infecção de um determinado vírus. Diante disso, é realizado o mapeamento de diversas amostras de RNA de indivíduos diferentes que são suscetíveis, ou não, à infecção. Então, os transcritomas obtidos são comparados com o objetivo de encontrar trechos expressos do DNA que causam a deficiência dos indivíduos suscetíveis, ou a imunidade para os não suscetíveis. Assim, é possível criar formas de tratamento que ativem ou desativem a transcrição dessas porções de DNA [82].

Para a fase de análise, pode ser utilizado um software como o *BLAST* [1], empregado para predizer funções para os genes em determinados genomas. Outro exemplo de software é o *R* [102], que possui diversas ferramentas para análises estatísticas, por exemplo, expressão diferencial.

2.3 Programas Utilizados em *Workflows* de Bioinformática

Um dos desafios para a elaboração deste trabalho foi a escolha de quais programas analisar, uma vez que a Bioinformática possui diversas ferramentas que podem ser utilizadas para a execução de seus *workflows* científicos. A escolha das ferramentas foi feita a partir dos artigos [31], [46] e [77], que descrevem algumas das aplicações mais utilizadas nas execuções dos *workflows* de bioinformática. Cada uma dessas ferramentas são descritas a seguir:

- Prinseq [99] é uma ferramenta que constrói estatísticas de sequências e qualidade dos dados. É comumente utilizada para filtrar, formatar e remover dados de sequências biológicas. Esta ferramenta é utilizada na fase de filtragem;
- Trinity [44] é um método eficiente e robusto de reconstrução de transcritoma *de novo*[2], usado para o processamento de grandes volumes de *reads* de RNA-Seq.É comumente utilizada na fase de montagem;
- TopHat [103] é um programa de alinhamento de *reads* geradas por RNA-Seq que utiliza internamente o mapeamento de *short reads* do Bowtie [61], descrito a seguir;

[2]A montagem de trascritoma *de novo* é o método de criar uma transcritoma sem a ajuda de um genoma de referência [45].

- Bowtie [61] é um programa para alinhar conjuntos de *reads*, sendo usado na fase de mapeamento. Ele constitui a base de uma série de outras ferramentas como o TopHat [103];
- Blast [14] é utilizada na fase de análise. Ela é uma ferramenta de busca de similaridade utilizado em projetos de bioinformática, que possui diversas funcionalidades, as quais são divididas pelo tipo de dado na sequência de consulta e nas sequências de bancos de dados.

Todavia, a execução de *workflows* de bioinformática demanda expressiva capacidade de processamento e/ou armazenamento. Para isso, pode-se utilizar o poder computacional disponível na computação em nuvem. Assim, o próximo capítulo apresentará as definições, os conceitos e as características da computação em nuvem.

Capítulo 3

Computação em Nuvem

Neste capítulo, será explanada a computação em nuvem. Inicialmente, na Seção 3.1, serão apresentadas suas principais características, as suas arquiteturas e o modo como funciona. Em seguida, a Seção 3.2 apresentará as nuvens federadas, que surgiram com a necessidade de aumentar o poder computacional e melhorar a provisão de serviços no ambiente de nuvem. Por último, a Seção 3.2.1 apresentará a plataforma de federação de nuvens utilizada como estudo de caso neste trabalho.

3.1 Visão Geral

A computação em nuvem é um paradigma de computação distribuída, que surgiu como uma tendência para a provisão de recursos e serviços de forma rápida, barata, escalável e flexível [32]. Diversos tipos de recursos podem ser disponibilizados, tais como memória, capacidade de processamento, armazenamento entre outros. Todavia, não há uma única definição de computação em nuvem na literatura. Algumas definições relevantes são:

- Armbrust *et al.* [3] definem nuvem computacional como "a união de aplicações oferecidas como serviço pela Internet, com o hardware e o software localizados em *datacenters* de onde o serviço é provido";
- De acordo com Foster *et al.* [32], computação em nuvem é "um paradigma computacional altamente distribuído, direcionado por uma economia de escala, na qual poder computacional, armazenamento, serviços e plataformas abstratas, virtualizadas, gerenciadas e dinamicamente escaláveis são oferecidos sob demanda para usuários externos por meio da Internet";
- Buyya *et al.* [13] definem como "um tipo de sistema paralelo e distribuído, que consiste em uma coleção de computadores virtuais interconectados que são provisionados dinamicamente, e apresentados como um ou mais recursos computacionais

unificados, baseados em acordos de nível de serviço estabelecidos entre provedor de recursos e o consumidor".

A primeira definição é bastante abrangente e, infelizmente deixa margem para que haja confusão com outros paradigmas de computação distribuída. É possível, por exemplo, confundir com a definição de um *grid* computacional [105]. A segunda já apresenta um elemento importante para a definição mais completa de nuvem, que é a frase "oferecido sob demanda". Isto significa que na computação em nuvem o usuário terá tanto recurso quanto demandado, diferentemente de como ocorre nos outros paradigmas de computação distribuída, e isso é uma diferença fundamental. A terceira definição fala sobre os acordos de nível de serviço (*Service Level Agreement* - SLA), que são acordos negociados entre o consumidor e o fornecedor e que definem quais indicadores irão medir a qualidade do serviço oferecido na nuvem. Assim, a violação deste acordo pode levar ao pagamento de multas [13].

Dessa forma, a computação em nuvem possui diversas características que a diferem dos outros sistemas distribuídos existentes. As principais características são [105]:

- *Self-service*: na computação em nuvem, os serviços, tais como armazenamento e processamento, são contratados pelo usuário diretamente, sem a participação de algum administrador dos provedores de nuvem, e de forma que atenda às necessidades do usuário [72];
- Amplo Acesso: a ideia é que os recursos estejam disponíveis na Internet e possam ser acessados a partir de dispositivos heterogêneos, desde que estes possuam acesso à rede mundial de computadores;
- *Pooling* de Recursos: os recursos de um determinado provedor são organizados em um *pool* de recursos físicos e virtuais, de forma a facilitar o acesso de vários usuários e os ajustes de demanda;
- Elasticidade: caso seja necessário aumentar a utilização de determinado recurso, seja ele qual for, este aumento deve ocorrer de forma fácil ou até mesmo de forma automática;
- Serviços Mensuráveis: a utilização dos recursos deve ser monitorada a todo momento, de forma a garantir que seja cumprido o contrato de qualidade de serviço realizado entre o provedor de serviço e o usuário.

3.1.1 Arquitetura de uma Nuvem

Na literatura, há diferentes propostas de arquitetura para plataformas de nuvem [63] [112], e a Figura 3.1 apresenta uma destas propostas. Nela é possível identificar três camadas,

que são: Software como serviço (*Software as a Service*), Plataforma como Serviço (*Platform as a Service*) e Infraestrutura como Serviço (*Infrastructure as a Service*) [105].

Os provedores de infraestrutura disponibilizam recursos para que os provedores de serviços hospedem suas aplicações sem precisar se preocupar com as questões de estrutura física, a fim de ganhar escalabilidade e flexibilidade. Os usuários de serviço acessam as aplicações que foram colocadas a disposição através da Internet. Tanto usuários de serviço quanto os provedores de serviço pagam apenas por aquilo que consumirem, o chamado *pay-per-use* [105].

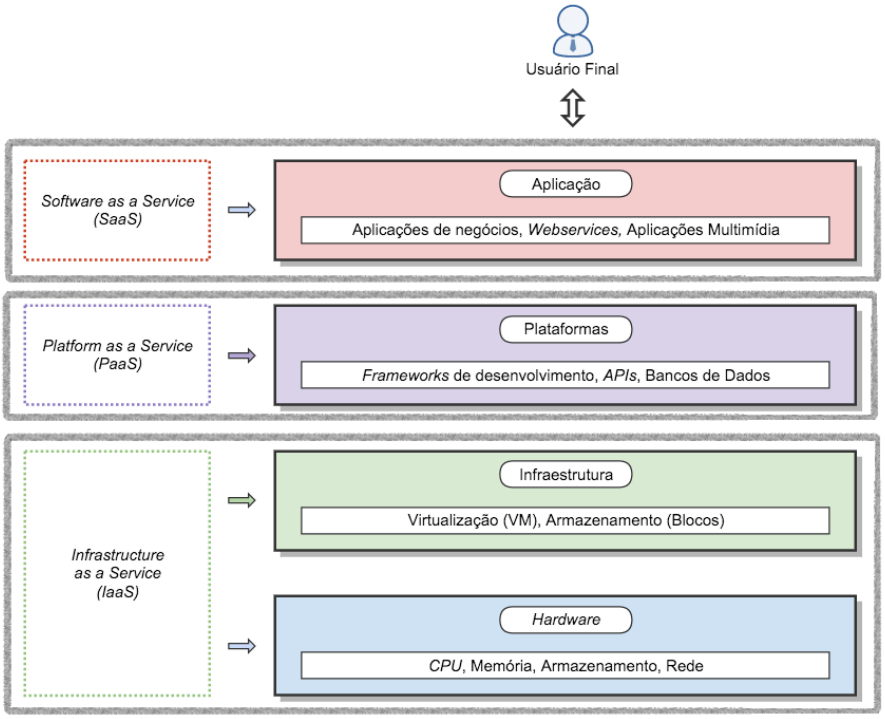


Figura 3.1: Arquitetura de Nuvem Computacional, proposta por Vaquero *et al.* [105].

A arquitetura de nuvem, apresentada na Figura 3.1 está dividida em três camadas distintas. A mais próxima do usuário é onde 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 [105].

Uma outra proposta de arquitetura foi feita por Foster *et al.* [32] e está representada na Figura 3.2. Nessa arquitetura a camada de infraestrutura contém os recursos computacionais, tais como os recursos de armazenamento e os de processamento. Esta camada é o hardware propriamente dito. 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 sob a camada de recursos unificados, tais como plataformas de desenvolvimento. E por fim, a camada de aplicação que contém as aplicações que executam na nuvem.

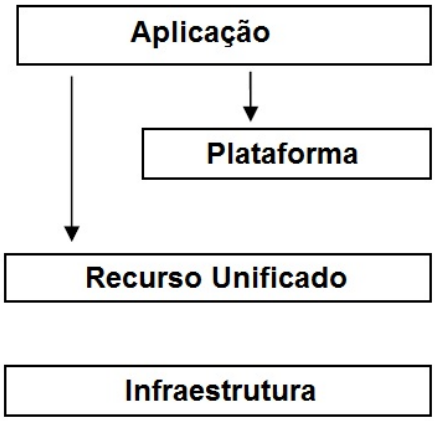


Figura 3.2: Arquitetura para a Plataforma de Nuvem, proposta por Foster *et al* [32].

3.1.2 Modelos de Serviço e Tipos de Nuvens

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 [91], o mais comum é dividir os serviços em *Infrastructure-as-a-Service*, *Platform-as-a-Service* e *Software-as-a-Service*, descritos a seguir:

- *Infrastructure-as-a-Service* – *IaaS*: os serviços que são oferecidos se caracterizam por serem de infraestrutura, podendo ser desde capacidade de processamento, de armazenamento ou até de máquinas virtuais completas. Um provedor desta camada que se destaca neste modelo de serviço a Amazon, com o *Elastic Compute Cloud* (EC2) [68] e com o *Simple Storage Service* (S3) [69];
- *Platform-as-a-Service* – *PaaS*: o que caracteriza este nível é a disponibilização de um ambiente no qual o usuário possa programar, testar e executar aplicações. O Google App Engine [41] é um exemplo nesta categoria, pois é um ambiente no qual aplicações podem ser desenvolvidas sem terem nenhum tipo de programa instalado na sua máquina, tudo é feito através da Internet;
- *Software-as-a-Service* – *SaaS*: são as aplicações disponibilizadas pelos provedores como serviços aos usuários comuns, de forma gratuita ou cobrando pela sua uti-

lização. Um exemplo é o Google Docs [40], no qual são disponibilizados editores de textos, editores de planilhas e ferramentes para a criação de apresentações. Os usuários podem acessar estes serviços em qualquer lugar, sem a limitação de ter o software instalado na sua máquina.

Além disso, as nuvens podem ser divididas em quatro tipos diferentes de implantação, que são nuvens públicas, privadas, comunitárias e híbridas, as quais são descritas a seguir:

- Nuvem Pública: este tipo de nuvem é aquele mantido por um fornecedor, geralmente, uma grande companhia, para um usuário comum ou uma empresa. É a nuvem no sentido tradicional do senso comum, na qual os recursos são provisionados dinamicamente e através da Internet [91];
- Nuvem Privada: é a nuvem que está dentro de um contexto empresarial e não está acessível a todas as pessoas, sendo restrita apenas aos funcionários e aos parceiros da empresa. Pelo fato de não estar disponível na Internet, apresenta vantagens em termos de segurança e largura de banda da rede;
- Nuvem Comunitária: infraestrutura que é compartilhada por organizações que mantém algum tipo de interesse em comum (jurisdição, segurança, economia), e pode ser administrada, gerenciada e operada por uma ou mais destas oganizações;
- Nuvens Híbrida: ambiente no qual ambos os tipos de nuvens anteriormente descritos podem ser utilizados em conjunto. É interessante para alguns modelos de negócios, pois arquivos sigilosos ou sistemas que manipulam dados sigilosos podem ser mantidos na nuvem privada, enquanto que os outros dados e sistemas podem ficar na nuvem pública, por exemplo.

Apesar da computação em nuvem ser escalável, atualmente, há aplicações que podem demandar por recursos computacionais superiores aos disponibilizados por uma úncia nuvem. A solução foi integrar mais de uma nuvem computacional, a qual foi chamada de federação de nuvens, que será descrita em detalhes na próxima seção.

3.2 Federação de Nuvens

Com o passar dos anos novas necessidades foram surgindo e a utilização de nuvens, de forma isolada, passou a não ser mais suficiente para algumas aplicações. Além disso, empresas começaram a criar *datacenters* ao redor do mundo para aumentar a segurança em caso de queda em algum dos servidores, e também como uma forma de atender às solicitações em menor tempo, diminuindo a distância entre o usuário e os servidores. Dessa

forma, integrar nuvens passou a ser algo necessário para continuar fornecendo serviços de forma rápida, eficiente e escalável. Assim sendo, a federação de nuvens computacionais pode ser definida como um conjunto de provedores de nuvens públicos e privados, conectados através da Internet [95].

Bittman [9] 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 [42], Amazon [68] e Microsoft [74]. 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 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, isto é, a etapa em que os provedores de nuvem trabalham exclusivamente sozinhos tem a tendência de acabar. Assim, tem-se alcançado a etapa em que os pequenos provedores se aliam horizontalmente.

Realizar uma federação, no entanto, não é algo simples devido as diferenças entre as nuvens, cada uma com suas particularidades, tanto no hardware (arquitetura do processador, por exemplo) quanto no software (sistema operacional ou outros softwares utilizados).

Contudo, para que haja a federação é necessário que os seguintes requisitos sejam atendidos [15]:

- Automatismo e Escalabilidade: uma nuvem, utilizando mecanismos de descoberta, deve ser capaz de escolher, dentre as nuvens existentes, qual aquela que atende às suas necessidades e deve ser capaz de reagir a mudanças;
- Segurança Interoperável: é necessário a integração entre diversas tecnologias de segurança, de forma que uma nuvem não precise alterar suas políticas de segurança para se juntar à federação.

Outros desafios para a criação de nuvens federadas foram identificados e são descritos a seguir [12]:

- Previsão de Comportamento da Aplicação: é importante que o sistema seja capaz de prever o comportamento das aplicações, para que ele possa tomar decisões inteligentes quanto à alocação de recursos, dimensionamento dinâmico, armazenamento ou largura de banda. Um modelo deve ser construído para tentar realizar esta previsão. Este modelo deve levar em consideração estatísticas de padrões de utilização dos serviços e ajustar as variáveis sempre que for necessário, para que o modelo seja o mais próximo possível da realidade;

- Mapeamento Flexível de Recursos e Serviços: algo que sempre é levado em consideração em qualquer projeto é o custo. É importante que o sistema seja o mais eficiente possível, sempre tentando encontrar a melhor configuração de hardware e software que atenda às demandas de qualidade de serviço que foram estabelecidas entre o usuário e o provedor. Esta é uma tarefa bastante complicada devido ao comportamento não previsível das aplicações e dos serviços que são utilizados na nuvem;
- Modelo Econômico Impulsionado por Técnicas de Otimização: o problema da tomada de decisões orientadas ao mercado é um problema de otimização combinatória, que visa encontrar a melhor combinação entre serviços e planos. Os modelos de otimização visam otimizar tanto os recursos centralizados (utilização, disponibilidade e incentivo) quanto os centrados nos usuários (tempo de resposta, o orçamento gasto e a justiça);
- Integração e Interoperabilidade: muitas empresas possuem dados sigilosos e não se sentirão confortáveis de colocá-los na nuvem. Esse medo é tanto pela questão da segurança, ou seja, medo de que alguma pessoa não autorizada tenha acesso a dados confidenciais, como também é pela forma como as aplicações que já existem na empresa irão interagir com os dados e as aplicações que estão na nuvem;
- Monitoramento Escalável dos Componentes do Sistema: atualmente as técnicas que são utilizadas para o monitoramento e o gerenciamento dos componentes da nuvem utilizam uma abordagem centralizada. Em sistemas distribuídos manter uma singularidade sempre é um problema, principalmente, por este poder se tornar um gargalo para o sistema, caso o volume de requisições seja muito alto, como também por questões de segurança, visto que algum problema pode prejudicar toda a federação de nuvens. É importante que este monitoramento seja feito de forma distribuída, para que não haja problema de escalabilidade, de desempenho e de confiabilidade.

3.2.1 Arquitetura de uma Federação

Para que os requisitos anteriormente citados sejam atendidos e os desafios superados, arquiteturas para a federação de nuvens foram propostas [12] [15]. Celesti *et al.* [15] propuseram uma arquitetura para federação em nuvem, veja a Figura 3.3. Neste modelo eles dividiram as nuvens em dois grupos, local e estrangeiro. A nuvem local consiste naquele provedor que já atingiu a saturação, não consegue mais atender às demandas e, portanto, deve repassar requisições para o restante da federação. Os provedores que vão

receber essas requisições são chamados de estrangeiros, e podem ou não cobrar por ceder seus recursos ociosos para atender às demandas que foram repassadas a ele.

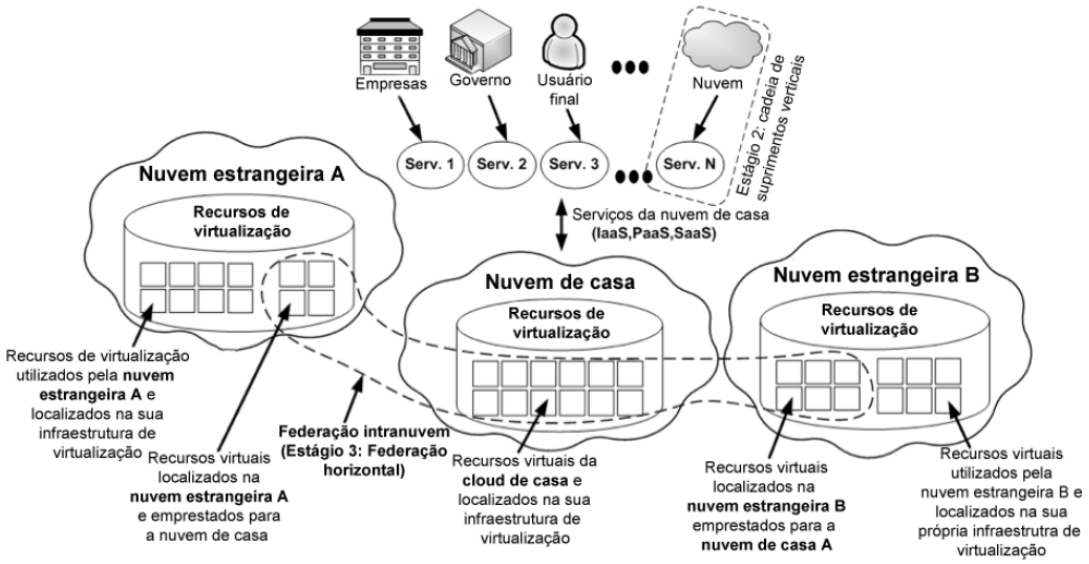


Figura 3.3: Arquitetura para Federação de Nuvens, proposta por Celesti *et al.* [15].

Vale ressaltar que uma nuvem pode ser local e estrangeira ao mesmo tempo, basta que ela esteja saturada em um determinado recurso, e por isso precise da ajuda de outros provedores para suprir essa demanda, e ao mesmo tempo está com outro recurso ocioso e o disponibiliza para outro provedor. Este repasse de requisições de uma nuvem para outra deve ser feito de forma transparente ao usuário, e de forma a atender os contratos de serviço que foram efetuados previamente.

Na arquitetura proposta por Celesti *et al.* [15], para cada provedor existente na federação existe um gerenciador chamado de *Cross-Cloud Federation Manager* (CCFM). O CCFM é o responsável por realizar a gerência das nuvens, verificando quando uma nuvem já está saturada, e tentando encontrar nuvens que estejam dispostas a contribuir com seus recursos ociosos. Ele está divido em três subcomponentes, cada um responsável por realizar uma fase dentre as três que dão nome a arquitetura, os quais são:

- *Discovery Agent*: é o agente responsável por realizar o processo de descoberta na federação. Para que este processo seja feito de forma dinâmica e distribuída é necessário que cada provedor de nuvem disponibilize suas informações para um local centralizado (apesar de ser logicamente centralizado é implementado de forma distribuída), e sempre que uma nuvem precisar de informações sobre os outros provedores basta consultar este local;

- *Match-Making Agent*: este agente é o responsável por escolher dentre todas as nuvens estrangeiras, qual é a que melhor se encaixa nos requisitos pretendidos. Depois de feita a escolha, ele também é responsável por garantir que os acordos de níveis de serviço sejam cumpridos e a qualidade de serviço seja mantida;
- *Authentication Agent*: agente que tem como função realizar a autenticação dos usuários dentre as diversas nuvens da federação. É uma tarefa complicada, pois cada nuvem pode ter vários usuários autenticados, e essas autenticações podem variar a todo instante. Outro problema é que cada nuvem pode utilizar uma tecnologia de autenticação diferente da outra, e este agente é responsável por realizar a interoperabilidade entre essas tecnologias.

Uma proposta alternativa de arquitetura foi apresentada por Buyya *et al.* [12], veja a Figura 3.4. Nesta proposta, o usuário utiliza um componente externo aos provedores para realizar qualquer tipo de interação com a federação. Este componente é chamado de *Cloud Broker* (CB). Ele é o responsável por intermediar a comunicação entre o usuário e os provedores, e também por identificar os recursos que estão disponíveis em cada provedor, de forma a atender os requisitos de qualidade de serviço (QoS) que foram definidos através de um contrato de SLA.

Para conseguir esses dados sobre os provedores de nuvem, o CB consulta o *Cloud Exchange* (CEx), um outro componente da arquitetura. A principal função do CEx é servir como um registro, que é consultado pelo CB a fim de obter informações sobre a infraestrutura, o custo de utilização, os padrões de execução e os recursos disponíveis, além de mapear as requisições dos usuários aos provedores.

Em cada provedor presente na federação existe um outro componente, chamado *Cloud Coordinator* (CC), responsável por incluir a infraestrutura disponível na federação e expor estas informações aos interessados. Para atender às requisições dos usuários, o CC realiza uma autenticação e estabelece um acordo de qualidade de serviço com cada CB, que também é responsável por encaminhar as tarefas para a execução.

Como não existe um padrão para a federação de nuvens computacionais para aplicações de Bioinformática, uma nova arquitetura foi proposta, chamada de BioNimbuZ [94], que tem como objetivo garantir de forma dinâmica, transparente e escalável a execução de aplicações no nível de *Software-as-a-Service*. Esta proposta é detalhada nas próximas seções.

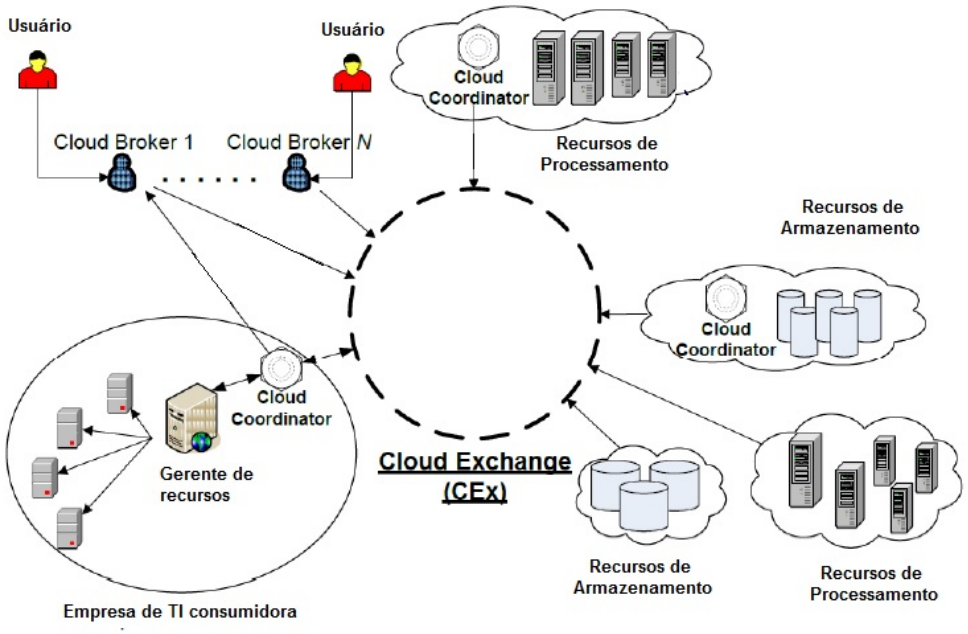


Figura 3.4: Arquitetura para Federação de Nuvens, proposta por Buyya *et al.* [12].

3.3 Plataforma BioNimbuZ

O BioNimbuZ é uma plataforma para federação de nuvens híbridas, que foi proposta originalmente por Saldanha [94], e que tem sido aprimorada constantemente em outros trabalhos [5], [6], [65], [87], [93], [94], [97]. Ele foi desenvolvido para suprir a demanda de plataformas de nuvens federadas na execução das aplicações de bioinformática, visto que a utilização de nuvens de forma isolada já não atende, em muito casos, às necessidades de processamento, de armazenamento, entre outras.

O BioNimbuZ permite a integração entre nuvens de diversos tipos, tanto privadas quanto públicas, deixando com 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) [101]. Porém, para alcançar os objetivos desejados de escalabilidade e de flexibilidade, percebeu-se a necessidade de alterar a forma de comu-

nicação entre os componentes da arquitetura do BioNimbuZ, pois a utilização de uma rede de comunicação P2P não estava mais suprindo as necessidades nestes dois quesitos. É importante ressaltar que os outros objetivos propostos por Saldanha [95], 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, Moura *et al.* [65] implementaram a utilização da Chamada de Procedimento Remoto (RPC) [57]. Para isso, adotou-se o Apache Avro [33] pois ele realiza a comunicação de forma transparente, permitindo a chamada de procedimentos que estão localizados em outras máquinas, sem que o usuário perceba [104]. E para auxiliar na organização e na coordenação do BioNimbuZ, utilizaram um serviço voltado à sistemas distribuídos chamado ZooKeeper Apache [34] e o protocolo SFTP [51], para transferências dos arquivos. Além disso, Moura *et al.* [65] implementaram uma política de armazenamento que considera a latência e o local em que o serviço será executado.

Após essa evolução, Azevedo e Freitas Junior [5] melhoraram a política de armazenamento na qual passou a considerar a quebra de arquivos e o cálculo da largura de banda entre o cliente/servidor em sua decisão de compactação e de transferência de arquivos. Ramos [87] implementou a interface gráfica do ambiente e um controlador de *jobs*, responsável por interconectar a Camada de Interface do usuário ao núcleo do BioNimbuZ.

Ainda na linha histórica, com o objetivo de melhor distribuir as tarefas da plataforma, Barreiros Júnior [6] 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.

Recentemente, 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 [97] modificou o serviço de armazenamento para que fosse possível utilizar o serviço de armazenamento ofertado pelos provedores de nuvem.

Com a implementação da interface gráfica, do novo escalonador e da melhoria no serviço de armazenamento, a plataforma sofreu algumas mudanças, resultando na arquitetura apresentada na Figura 3.5 e detalhada na próxima seção.

3.3.1 Arquitetura do BioNimbuZ

O BioNimbuZ faz uso de uma arquitetura hierárquica distribuída, conforme apresentada na Figura 3.5. Ela representa a interação entre as quatro camadas principais: Aplicação, Integração, Núcleo e Infraestrutura.

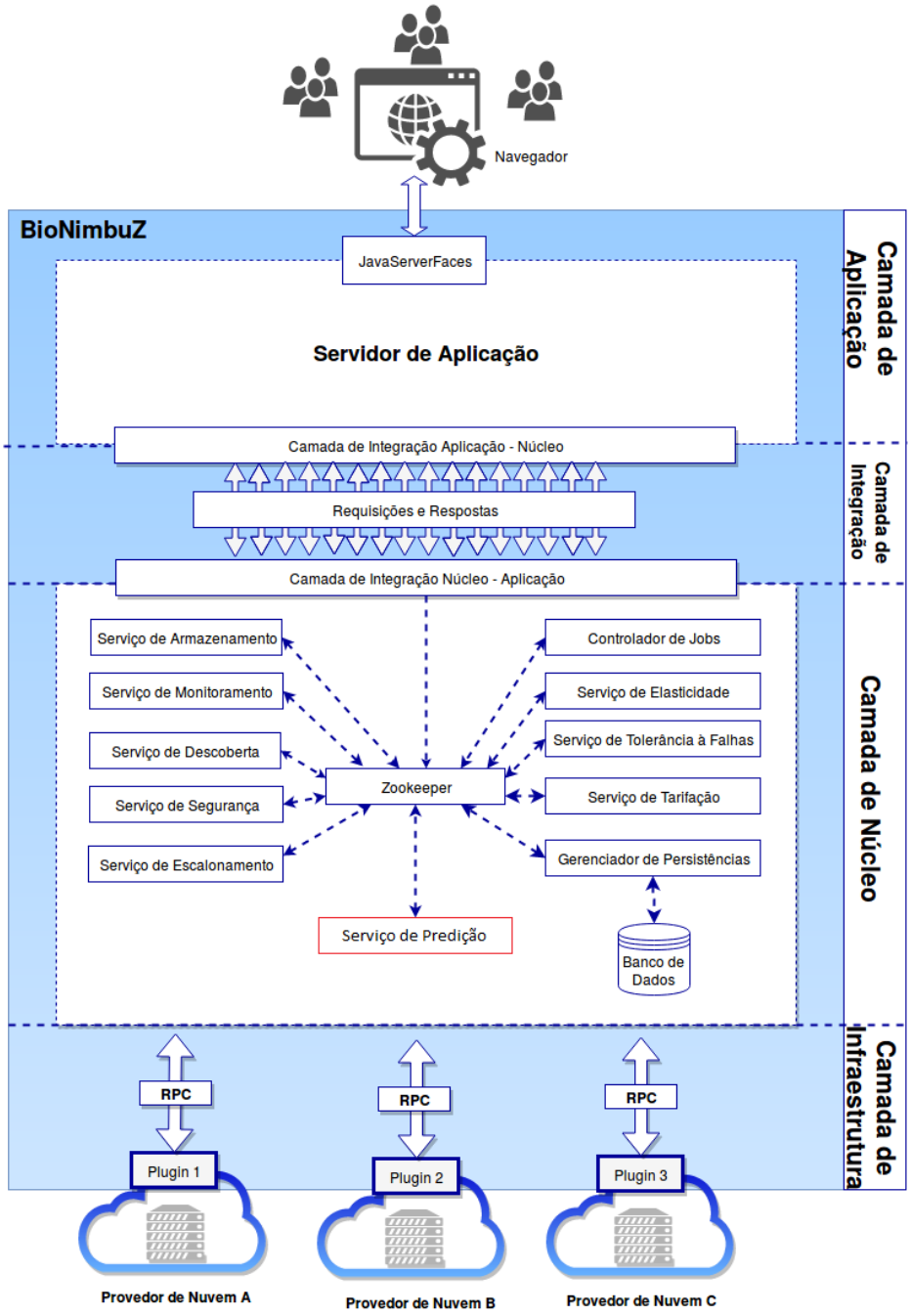


Figura 3.5: Arquitetura da Plataforma BioNimbuZ.

A primeira camada - Camada de Aplicação, permite a integração entre a aplicação do usuário e os serviços do núcleo da plataforma. A interface com o usuário ocorre por meio de uma interface gráfica ou linha de comando, e é nela que devem ser inseridos os *workflows* que serão executados nas diversas nuvens. A aplicação, então, comunica-se com a camada de Núcleo por meio da segunda camada, a Camada de Integração, que é responsável pela transição dos dados entre a aplicação e o núcleo. A terceira camada, Camada de Núcleo, é a responsável por realizar toda a gerência da plataforma e seus

serviços, e por se comunicar com os *plugins* de cada provedor, que estão na quarta camada de Infraestrutura. Os *plugins* mapeiam as requisições e as enviam para cada provedor. A seguir serão descritos o funcionamento e as características de cada uma destas camadas em detalhes.

E para realizar a comunicação de forma transparente, permitindo a execução de chamadas de procedimento que estão localizados em outras máquinas, sem que o usuário perceba, o sistema de RPC e de serialização Apache Avro [33] é utilizado. Para auxiliar na organização e na coordenação do BioNimbuZ, utiliza-se um serviço voltado à sistemas distribuídos chamado Apache ZooKeeper [34]. Esses serviços são apresentados a seguir:

Apache Avro

O Apache Avro [33] é um sistema de RPC e de serialização de dados desenvolvido pela Fundação Apache [36]. Algumas vantagens deste sistema são o fato de ser livre, a utilização 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 [22]. Em relação aos protocolos, pode-se utilizar tanto o protocolo HTTP [30] ou o protocolo próprio do Avro, que é o protocolo RPC [35].

O Avro foi criado para ser utilizado com um grande volume de dados, e possui algumas características [35] definidas pela própria Fundação Apache, tais 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.

O Avro foi escolhido como *middleware* de Chamada Remota de Procedimento para o BioNimbuZ, por ser livre, flexível e possuir integração com várias linguagens. Ele funciona de modo em que as requisições de execuções de procedimentos, sejam serializadas para a execução remota e ao receber a requisição, a máquina responsável pela execução recebe essa execução já deserializada.

Apache Zookeeper

O Zookeeper [34] 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 [52].

No Zookeeper 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 partipante 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 3.6 pode ser visto um exemplo da estrutura hierárquica dos *znodes* que foi implementada no BioNimbuZ, no qual os metadados da aplicação são armazenados de maneira em que seja eficiente a forma de recuperação de informação da federação.

Assim, como pode ser observado na Figura 3.6, 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, em que 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.

Os *watchers* funcionam como observadores de mudanças, e enviam alertas sobre as alterações ocorridas em algum dos *znodes*. Este conceito é útil para sistemas distribuídos, pois permitem o monitoramento constante do sistema, deixando para que *watchers* avisem quando um provedor estiver fora do ar, por exemplo, ou então quando um novo recurso estiver disponível. Na plataforma BioNimbuZ esses *watchers* são instanciados pelos serviços e controladores, por meio do *framework* chamado Curator [53], o qual é disponibilizado pelo Apache para facilitar o *CRUD* de *znodes*, e o instanciamento dos *watchers* para os serviços no Zookeeper [113].

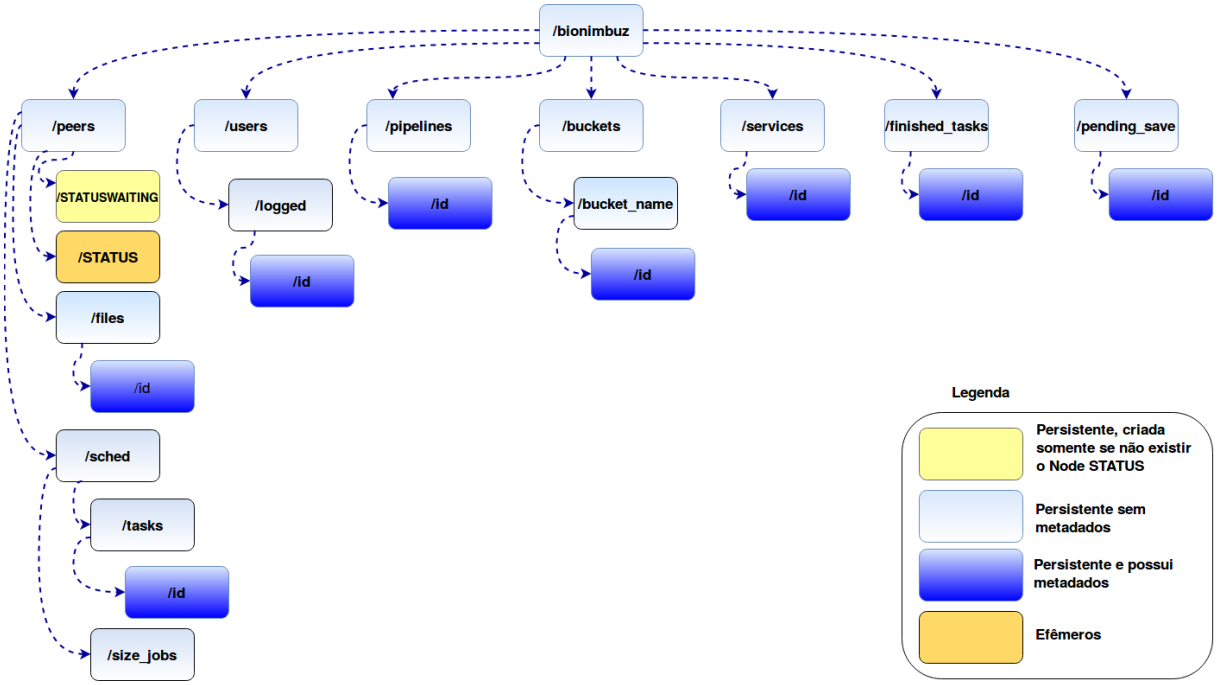


Figura 3.6: Estrutura Hierárquica dos *Znodes* no BioNimbuZ [87].

3.3.2 Camada de Aplicação

A Camada de Aplicação é a responsável por fazer toda a comunicação com o usuário. Por meio desta interface os usuários devem inserir as ações que desejam executar na federação. A interação com o usuário é realizada por meio de páginas web, ou seja, interface gráfica (GUI).

Atualmente, esta camada é composta por uma aplicação *web* que implementa um sistema gerenciador de *workflows* científicos. Essa interação inclui tarefas como *uploads* de arquivos e envio de *workflows* a serem executados. Além disso, também é responsabilidade da Camada de Aplicação exibir informações de *feedback* do sistema para o usuário, tais como listagem de arquivos armazenados e informações sobre uma determinada execução, por exemplo tempo, preço e horário de término, conforme são apresentados nas Figuras 3.7, 3.8, 3.9, 3.10 e 3.11.

Na Figura 3.7, é possível observar a tela inicial do sistema, que é a primeira tela após o *login* do usuário, nessa tela, e em todas as outras, é mostrado um *menu* com as seguintes funcionalidades disponíveis ao usuário:

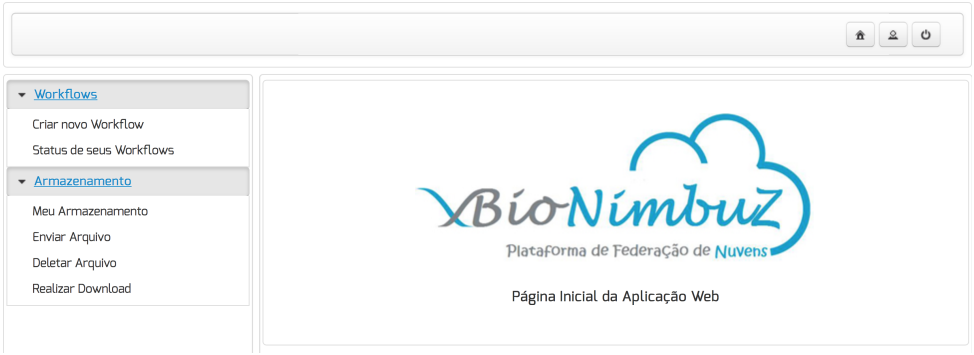


Figura 3.7: Tela Inicial da Aplicação *Web* do BioNimbuZ [87].

- *Workflows*: funcionalidades referentes ao gerenciamento dos *workflows* do usuário, tais como:
  - Criar novo *Workflow*: a partir desta opção o usuário pode iniciar ou importar um *workflow*;
  - *Status* de seus *Workflows*: possibilita ao usuário visualizar o estado e o histórico de execução de seus *workflows*.
- Armazenamento: funcionalidades referentes ao gerenciamento do armazenamento do usuário, sendo elas:
  - Meu Armazenamento: nessa opção o usuário pode verificar a utilização de sua cota de armazenamento;
  - Enviar Arquivo: possibilita ao usuário realizar o *upload* de arquivos à plataforma;
  - Deletar Arquivo: opção referente à deleção de arquivos do usuário;
  - Realizar *Download*: utilizada caso o usuário queira fazer o *download* dos arquivos enviados à plataforma, ou dos arquivos gerados como saída de seus *workflows*.

Ao clicar na opção, "Criar novo *Workflow*" do *menu* apresentado na Figura 3.7, o usuário é direcionado para uma outra tela, na qual ele iniciará a criação do novo *workflow*, preenchendo a descrição, e selecionando os serviços que serão executados no *workflow*. Após a escolha dos serviços, o usuário é redirecionado para a tela seguinte (mostrada na Figura 3.8), passando para a fase de *design* do *workflow*. Nela são apresentados os elementos escolhidos pelo usuário, dispostos de maneira gráfica, conforme mostrado na Figura 3.8.

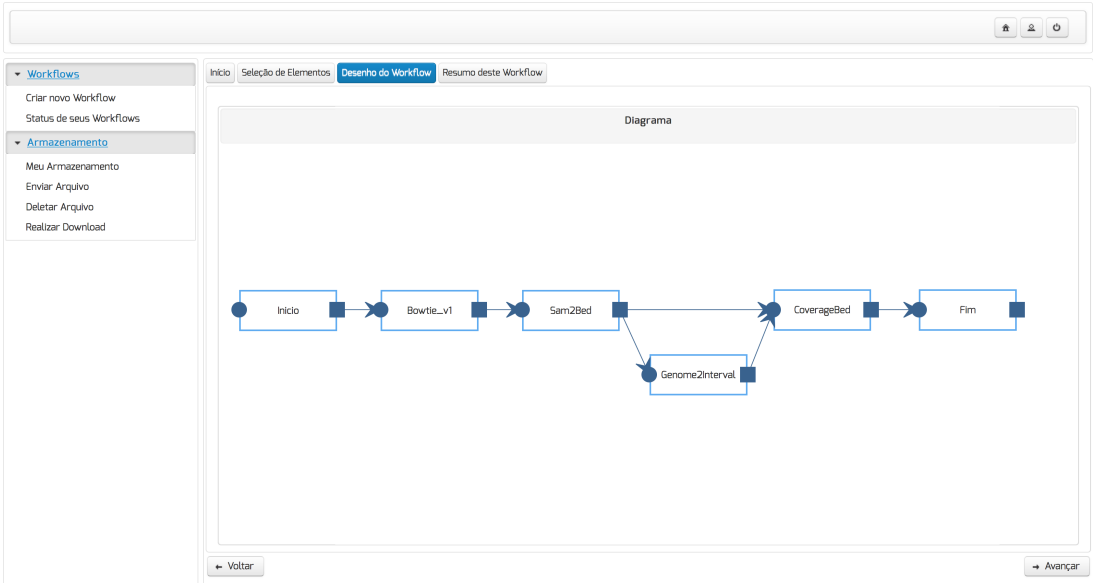


Figura 3.8: Tela de Montagem de Fluxo do *Workflow* da Aplicação [87].

Com o objetivo de compor um *workflow*, o usuário precisa fornecer alguns dados, e um desses dados são os arquivos de entrada, que são o objetivo de análise para a execução do *workflow*. Para isso, é necessário enviar os arquivos de entrada para a plataforma do BioNimbuZ, a partir da opção "Enviar Arquivo", do *menu* apresentado na Figura 3.7, que redireciona para a tela mostrada na Figura 3.9, possibilitando o envio desses arquivos.

Figura 3.9: Tela de *Upload* de Arquivos [87].

A partir do momento em que o usuário escolhe submeter o *workflow* à plataforma BioNimbuZ, ele pode acompanhar o estado de sua execução na tela de *status*, selecionando a opção "*Status* de seus *Workflows*", do *menu* apresentado na Figura 3.7. Essa opção permite que o usuário visualize o *status* de cada *workflow* submetido por ele, conforme é apresentado na Figura 3.10.

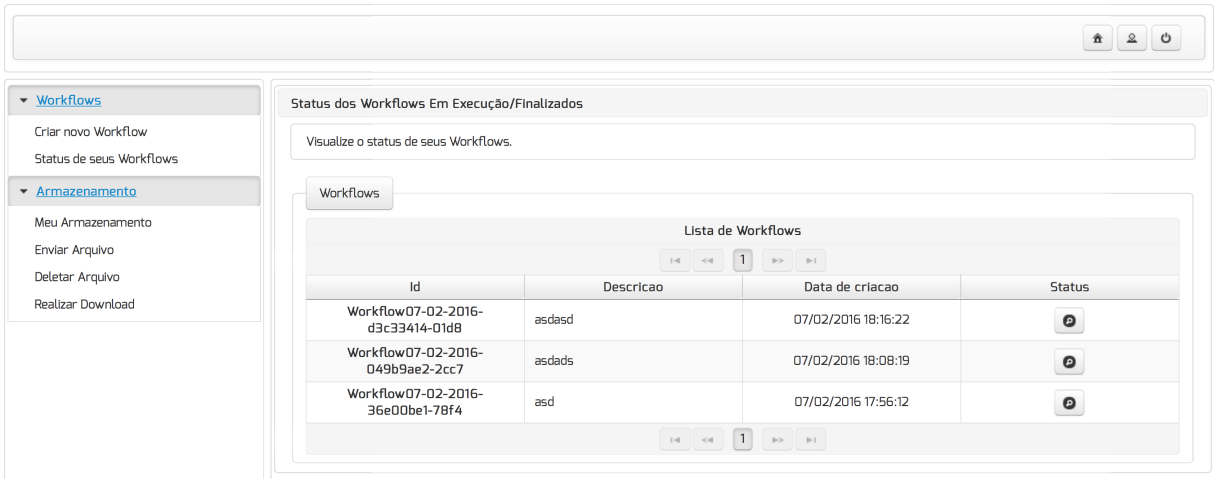


Figura 3.10: Tela de Listagem das Execuções dos *Workflows* do Usuário [87].

E para o usuário rastrear a execução de determinado *workflow*, e realizar o *download* dos arquivos de saída gerados pela execução do *workflow*, a tela apresentada na Figura 3.11 possui *logs* e também os arquivos de saída disponíveis para *download*, possibilitando esse acompanhamento mais detalhado da execução. Ao clicar em "*Download*", o navegador do usuário inicia a transferência do arquivo do BioNimbuZ para a máquina local do usuário.

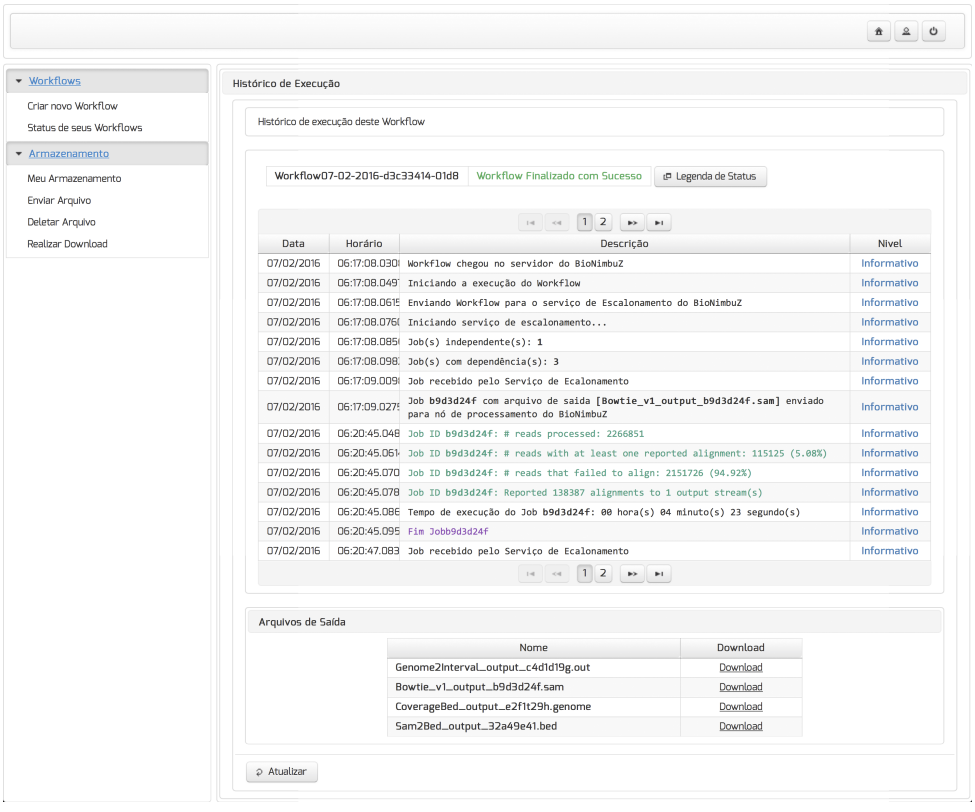


Figura 3.11: Tela de Monitoramento da Execução do *Workflow* [87].

3.3.3 Camada de Integração

A Camada de Integração é a responsável por integrar as Camadas de Aplicação e de Núcleo. Ela realiza essa tarefa por meio da troca de mensagens entre essas camadas, via serviços *web*. Assim, a Camada de Integração permite disparar requisições da Camada de Aplicação para o Núcleo, e receber respostas do Núcleo para a Camada de Aplicação.

Para tratar as requisições recebidas a partir da Camada de Aplicação e enviar respostas para ela, utiliza-se um *framework* REST, chamado Resteasy [88]. A principal vantagem ao se desenvolver interfaces baseadas em REST é que sistemas com a capacidade de enviar requisições HTTP pela Internet, podem ser integrados a outros sistema com essas mesmas caracteríticas. Dessa forma, a interface desenvolvida na Camada de Núcleo do BioNimbuZ é independente da aplicação que irá acessá-lo [87].

3.3.4 Camada de Núcleo

A Camada de Núcleo, também chamada apenas de Núcleo, é a responsável por toda a gerência da federação, tendo como atividades o descobrimento de provedores e recursos, o escalonamento de tarefas, o gerenciamento de tarefas, o armazenamento de arquivos e o controle de acesso dos usuários, entre outras. Para cada uma destas atividades existe um serviço responsável. A seguir serão descritos os serviços que contemplam o Núcleo da plataforma BioNimbuZ:

- Controlador de *Jobs*: é quem recebe a requisição que vem da aplicação, verificando as credenciais dos usuários com o Serviço de Segurança, para permitir ou não a execução das tarefas. Além disso, o Controlador de *Jobs* é o 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 [87];
- Serviço de Elasticidade: é o serviço responsável por dinamicamente aumentar ou diminuir o número de instâncias de máquinas virtuais, ou então reconfigurar os parâmetros de utilização de CPU, de memória, de largura de banda, entre outros recursos que são disponibilizadas pelos provedores de nuvem. A elasticidade pode ser vertical, quando há um redimensionamento dos atributos de CPU, de armazenamento, de rede ou de memória; como também pode ser horizontal, quando o número de instâncias de máquinas virtuais é modificado para mais ou para menos.

  Para tomar as ações de elasticidade do BioNimbuZ, tem-se um Controlador de Elasticidade, como é apresentado na Figura 3.12, que foi proposto por Vergara [106].

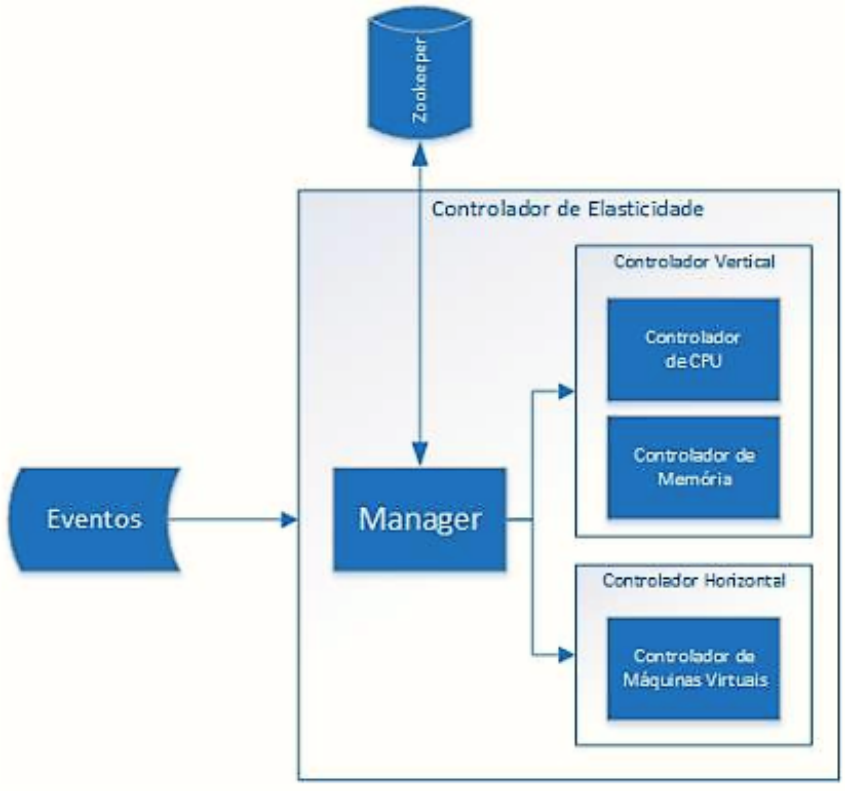


Figura 3.12: Arquitetura do Serviço de Elasticidade [106].

Esse controlador possui três módulos principais, os quais são o *Manager*, o Controlador de Elasticidade Horizontal e o Controlador de Elasticidade Vertical. O *Manager* é o responsável por receber os eventos de disparo da elasticidade, por decidir entre elasticidade horizontal ou vertical, e por atualizar o ZooKeeper com as informações das máquinas virtuais. O segundo módulo é o Controlador de Elasticidade Horizontal, o qual é responsável por escolher qual o tipo de máquina virtual deve ser criada e instância-la. Além disso, ele também é responsável por remover máquinas virtuais que estão inutilizadas. Por último, tem-se o Controlador de Elasticidade Vertical, responsável por aumentar/diminuir o número de CPU e de memória das máquinas virtuais;

- Serviço de Monitoramento: é o serviço responsável por realizar um acompanhamento dos recursos da federação, junto ao Zookeeper, que é capaz de tratar 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 rescontruções 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, e 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 de alguma tarefa - aqui chamado de *job* - e as distribui dinamicamente entre os provedores disponíveis, divididas em instâncias menores - *tasks*.

  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. O Serviço de Escalonamento do BioNimbuZ possui, atualmente, duas políticas de escalonamento, as quais são o ACOSched [80] e o C99 [6]. Ambos implementam técnicas de escalonamento distintas, sendo o último usado atualmente. O algoritmo C99 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, mais detalhes sobre o C99 podem ser vistos no trabalho de Barreiros Júnior [6];

- 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 implementado com o trabalho de Bacelar e Moura [78]. Em seguida, Gallon [37] desenvolveu outra política que levou em consideração alguns critérios a mais para a escolha da melhor nuvem para o armazenamento dos arquivos. Esses critérios são, além da latência, o custo de armazenamento, os núcleos de processadores e a carga de trabalho. Azevedo e Freitas Junior [5], aprimoraram o Serviço de Armazenamento do BioNimbuZ, adicionando a largura de banda e a verificação da necessidade de se compactar o arquivo que

será realizado o *upload* na métrica de cálculos, e nas métricas de decisão de envio do arquivo.

Santos [97] adicionou ao serviço de armazenamento um módulo que permite dispor e armazenar os arquivos da aplicação nos serviços de armazenamento em nuvem, possibilitando, assim, que o serviço de armazenamento tenha dois modos diferentes de armazenamento. O primeiro modo é o armazenamento nas próprias máquinas virtuais que estão executando o *workflow*, conforme feito anteriormente. O segundo modo utiliza-se da tecnologia dos serviços de armazenamento por objeto [27], oferecidos pelos provedores externos utilizados na federação, mais especificamente, o Amazon S3 [69] e o Google *Cloud Storage* [39]. Nesse segundo modo, o armazenamento dos arquivos é realizado em *buckets*, que são volumes nos quais os arquivos são gravados. 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. Outra estratégia adotada no armazenamento dos arquivos no BioNimbuZ é a replicação em mais de um provedor, garantindo que os arquivos estejam disponíveis quando as aplicações forem utilizá-los;

- Serviço de Tolerância a Falhas: tem como objetivo garantir que todos os principais serviços estejam sempre disponíveis. Assim, em caso de falhas, garantir que os serviços sejam recuperados. Essa recuperação exige que todos os objetos que forem afetados pela falha voltem ao estado anterior. O serviço de tolerância a falhas possui atuação de forma distribuída na plataforma BioNimbuZ, e está presente em outros serviços.

  No Serviço de Armazenamento, por exemplo, quando um alerta de indisponibilidade de um recurso é lançado por um *watcher*, é iniciada uma rotina de recuperação para os arquivos que este recurso continha. Como os arquivos são armazenados de forma duplicada na federação, a recuperação ocorre de forma a identificar quais arquivos foram perdidos e realizar uma nova duplicação em outro servidor do BioNimbuZ.

  No Serviço de Escalonamento, a recuperação está presente quando é recebido um alerta de indisponiblidade de um determinado recurso, e todas as tarefas que foram escalonadas e ainda não haviam sido executadas, ou estavam em execução, são novamente escalonadas, agora para outras máquinas na federação. Essas recuperações são possíveis devido aos *watchers*, que disparam alertas aos responsáveis, avisando sobre os problemas em algum recurso da federação.

- Serviço de Tarifação: é o serviço responsável por 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, obtendo 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 recepção de recursos de diversas nuvens autônomas.

Assim, no BioNimbuZ é função do Serviço de Tarifação 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 de que há recursos de diversas nuvens. Diante do exposto, foi definida uma arquitetura para o Serviço de Tarifação do BioNimbuZ, arquitetura essa que pode ser observada na Figura 3.13.

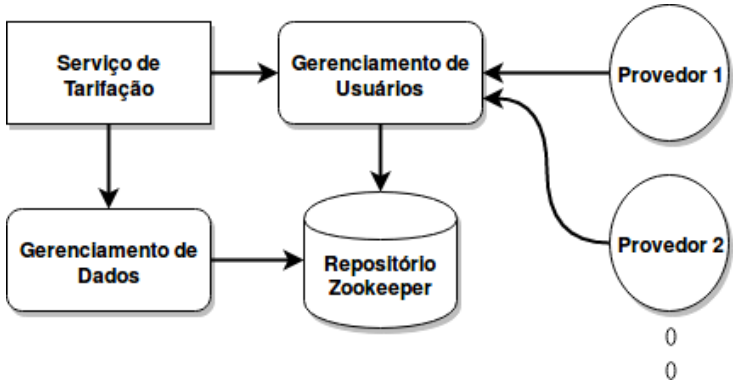


Figura 3.13: Arquitetura do Serviço de Tarifação do BioNimbuZ.

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 de Gerenciamento de Dados e de Gerenciamento de Usuário sejam mantidos no ambiente. Dessa maneira, todas as informações do Serviço de Tarifação ficam disponíveis para que os outros serviços possam usá-las.

- Serviço de Segurança: segurança em nuvens federadas 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. Após a autenticação, vem a autorização, que consiste em verificar se o usuário pode realizar as ações que deseja. Muitos outros aspectos podem ser abordados 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 [87] [98].

3.3.5 Camada de Infraestrutura

E por fim, a última camada da arquitetura do BioNimbuZ a Camada de Infraestrutura que consiste em todos os recursos que os provedores de nuvens colocam a disposição da federação, somados aos seus respectivos *plugins* de integração. O principal objetivo desta camada é prover uma interface de comunicação entre o Núcleo do BioNimbuZ e os provedores de nuvens, utilizando para tal os *plugins* que mapeiam as requisições provenientes da arquitetura, para comandos específicos para cada provedor, individualmente.

Contudo, é necessário prestar um serviço transparente e confiável no que diz respeito ao custo e/ou demanda de recursos necessários para cada execução. Assim, um módulo que expressa ao usuário custos e/ou recursos que serão utilizados é importante para garantir a transparência e possibilitar que o próprio usuário tome as decisões de custo e de desempenho relacionados aos experimentos executados. Nesse contexto, o próximo capítulo abordará uma melhor compreensão do tema proposto por este trabalho, que é o Serviço de Predição de Custos e Recursos Computacionais.

3.3.6 Considerações Finais

Diante do exposto neste capítulo, o BioNimbuZ terá na Camada de Núcleo, após a conclusão deste trabalho, o Serviço de Predição, que é o responsável por avaliar a entrada do *workflow* pelo usuário e estimar, através de suas técnicas implementadas, o custo, o tempo e os recursos para determinada execução. Além disso, o serviço de predição deve permitir que o usuário escolha por uma execução de baixo custo, ou de alta performance. O próximo capítulo descreve detalhadamente a predição de recursos.

Capítulo 4

Predição de Recursos

Os objetivos deste capítulo são mostrar o estado da arte, as definições e as técnicas que vem sendo usadas para a predição de recursos em computação em nuvem, e em nuvens federadas. Para isso, a Seção 4.1 apresentará uma visão geral sobre predição de recursos. A Seção 4.2 apresentará as técnicas comumente utilizadas e a Seção 4.3 apresentará os trabalhos relacionados.

4.1 Visão Geral

A predição de recursos surge da necessidade de analisar um *workflow* para que seja possível estimar o dimensionamento de recursos próximo do ideal, com o objetivo de eliminar problemas de ociosidade ou de falta de recursos, resultantes de escolhas errôneas. Este cenário se torna ainda mais complexo ao considerar um ambiente de nuvem federada, pois os diferentes tipos de recursos a serem escolhidos crescem exponencialmente [17] [20].

Para os provedores, a predição do dimensionamento de recursos é importante no sentido de manter a utilização dos recursos de acordo com a demanda, sem desperdícios de recursos, podendo, por exemplo, ser aplicado para garantir a Qualidade do Serviço – QoS [55]. Além disso, provedores também oferecem seus recursos com preços diferentes para modelos de execução distintos e, com isso, o custo para *workflows* diferentes podem diferir bastante, dependendo dos recursos escolhidos para a execução. Há também modelos em que se pode alocar a longo prazo determinadas instâncias locais que garantem a baixa latência na troca de mensagens. Assim, os provedores são capazes de classificar o modelo adequado de preços com base nas características do *workflow* e das informações prestadas pelos usuários através da predição [50].

Assim sendo, para um serviço de predição, é fundamental que o sistema seja capaz de prever as flutuações de demanda e ações dos serviços, para que, de forma inteligente, tome decisões relacionadas a custos, tempo, qualidade de serviço, dimensionamento dinâmico

etc. Assim, a especulação das necessidade futuras em termos de computação, de armazenamento, de banda de rede, de requisições etc, para julgar custos, tempo e recursos, implica em uma previsão eficaz de um ambiente controlável, que aumenta a confiabilidade do sistema [50].

Com isso, o provisionamento de recursos ideal envolve tipicamente abordagens de construção de modelos que preveem as demandas de recursos durante a execução e a utilização do modelo periodicamente para prever as demandas futuras e, automaticamente, alocar recursos, utilizando os requisitos previstos. O modelo pode ser construído usando várias técnicas, incluindo Aprendizado de Máquina [11] [24] [50] [59], Estatística [50] [54], Metaheurísticas [19] [20] [81], dentre outras.

Não menos importante é o tempo necessário para que os modelos estipulem os recursos computacionais que serão necessários para a execução do *workflow*. Logo, é necessário escolher técnicas com tempo de resposta hábil, que não influenciem expressivamente no tempo final de execução do *workflow*. Neste contexto, a utilização de metaheurísticas tem sido boa escolha para estes cenários, pois com o uso delas é possível obter soluções eficazes em menor tempo, quando comparado com técnicas de aprendizado de máquina [20] [81], por exemplo. Com isso, a próxima seção apresenta as técnicas comumente utilizadas para a predição de recursos.

4.2 Técnicas de Predição

Análise preditiva de recursos, como visto na seção anterior, é a chave para várias decisões, tais como gerenciamento de carga, dimensionamento de recursos, custo financeiro, tempo de execução entre outros.

Logo, é necessário investir em técnicas de predição eficazes que ofereçam soluções viáveis de forma rápida e eficiente. As principais técnicas usadas na literatura para predição de recursos tem sido: Aprendizado de Máquina [11] [24] [50] [59], Estatística [50] [54], Metaheurísticas [19] [20] [81]. Essas técnicas serão detalhadas nas próximas seções.

4.2.1 Aprendizado de Máquina

O aprendizado de máquina é um campo da Inteligência Artificial responsável pelo estudo e pelo desenvolvimento de mecanismos computacionais de aperfeiçoamento, com a capacidade de aprender automaticamente a partir de grandes quantidades de dados e produzir hipóteses úteis [70].

O aprendizado de máquina possui diferentes tipos de algoritmos que propõem soluções ou modelos para diversos problemas. Esses algoritmos são classificados a partir de diferentes abordagens, sendo algumas delas o aprendizado supervisionado e o não-supervisionado

[47]. O primeiro, também conhecido como classificação e regressão, durante a fase de treinamento recebe exemplos de entrada, que pertencem a mesma classe da informação da saída desejada. Por outro lado, o aprendizado não-supervisionado recebe entradas que não estão na classe dos resultados de saída esperado, ou seja, *a priori* não há informação da classe que o conjunto de treinamento pertence [70]. Algumas das abordagens do aprendizado de máquina são descritas em seguida.

As Máquinas Vetores de Suporte – MVS [21], basicamente, são máquinas de aprendizagem que se baseiam na Teoria da Aprendizagem Estatística, treinadas por um algoritmo supervisionado. Conceitualmente, vetores do espaço de entrada são mapeados não linearmente em um espaço de características de alta dimensionalidade, através de um mapeamento estabelecido a *priori*. A partir desse espaço é construída uma superfície de decisão linear, constituída de um hiperplano de separação ótima, que expressa propriedades que garantem aptidão na habilidade de generalização da máquina de aprendizagem [21]. A Figura 4.1 mostra um esquema básico da técnica utilizada por uma MVS, que busca separar ao máximo os dados em conjuntos, através de hiperplanos. No contexto da predição de recursos, a técnica pode ser utilizada para prever o tempo de execução de tarefas a partir de um conjunto de informações anteriores, uma vez que utiliza preditores baseados em regressões estatísticas [59]. Porém, o aprendizado depende de um conjunto massivo de dados para um aprendizado eficaz.

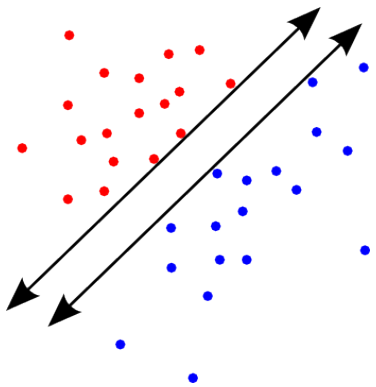


Figura 4.1: Esquema de Classificação de uma MVS, adaptado de [109].

Outro exemplo são as Redes Neurais – RN [107], que basicamente utilizam um mapeamento não linear de um vetor de espaço de entrada para um vetor de espaço de saída, construído através de camadas de funções de ativação ou neurônios artificiais. Nesses, as coordenadas de entradas são computadas de acordo com seus respectivos pesos, que produzem uma saída simples, ativa ou não, de acordo com o nível de disparo. A Figura 4.2 mostra um exemplo didático do funcionamento de uma rede neural, na qual as setas da esquerda representam um conjunto de dados de entrada, que disparam um conjunto

de ativações apresentadas pelos círculos em verde denominado camada de entrada, em seguida, as camadas intermediárias, em azul, realizam o processamento, através de conexões ponderadas, que podem ser consideradas como extratoras de características. Por último em laranja, a camada de saída apresenta a saída esperada. É importante ressaltar que, como as MVS, as redes neurais também estão classificadas como aprendizado supervisionado.

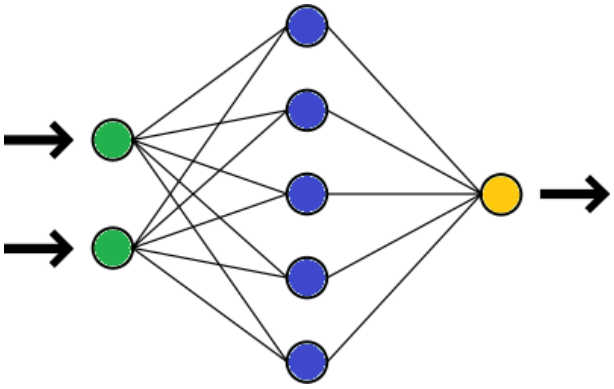


Figura 4.2: Diagrama de uma Rede Neural. Adaptado de [92].

Assim sendo, redes neurais também são comumente utilizadas na predição de recursos, principalmente, na construção de modelos que são capazes de dimensionar recursos para ambientes de nuvem através da predição de recursos, tais como CPU, memória, I/O, disco, etc, prevendo aumento da demanda dos recursos e auxiliando no provisionamento em tempo real, como demonstrado no trabalho de Islam [50].

Todavia, um dos problemas da utilização das técnicas listadas de aprendizado de máquina é a necessidade de um conjunto de dados massivo para um treinamento de qualidade, pois resultam em boas previsões, na maioria das vezes, desde que construídas de forma eficaz. Porém, em alguns cenários, como o da estimativa de recursos para execução de *workflows*, no qual é oneroso obter um conjunto representativo de execuções, essa técnica não é adequada [50].

4.2.2 Métodos Estatísticos

Há também técnicas estatísticas para a previsão de recursos. Nessa área, autores comumente utilizam Regressão Linear como modelo preditivo [50]. Regressão linear, em geral, trata da possibilidade de se estimar um valor condicional não esperado, ou seja, uma técnica estatística para modelar e investigar a relação entre variáveis. A Regressão Linear Simples — RLS [73] procura resumir a relação entre duas variáveis, representada pela Equação 4.1 [75].

$$y = \beta_0 + \beta_1 x + \epsilon \tag{4.1}$$

A Equação 4.1 é linear o termo ϵ é o erro aleatório, reconhecido como normal e independente distribuído, com variância – σ^2 contante e desconhecida, além de possuir média zero [75].

Em um mesmo contexto, também há a Regressão Linear Múltipla — RLM [73], que possui um conjunto de técnicas estatísticas que viabilizam a avaliação de uma variável dependente, conhecida também como preditando, além de diversas outras variáveis independentes, conhecidas como preditoras. Nesses preditores se tem uma equação linear utilizada para estimar o preditando como uma combinação linear dos preditores. Assim, a principal diferença entre a RLM e a RLS é a quantidade de variáveis preditoras, sendo que na Equação 4.2, o k é o número de variáveis preditoras, já na RLS tem-se o $k = 1$ (veja a Equação 4.1) [75].

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon \tag{4.2}$$

Os parâmetro $\beta_0, \beta_1, ..., \beta_k$ da Equação 4.2 são coeficientes de regressão, sendo o β_0 o intercepto da reta, e $\beta_1, ..., \beta_k$ representam a inclinação da reta [75].

4.2.3 Metaheurísticas

Metaheurísticas são procedimentos que combinam diferentes heurísticas [2] [89] [90]. Elas se diferem basicamente pelos mecanismos utilizados para escapar de ótimos locais. As metaheuristicas são divididas em duas categorias, segundo o seu critério de exploração do espaço de soluções, que são os métodos populacionais e os métodos baseados em busca local [10].

Os métodos populacionais mantém um conjunto de boas soluções e, através dessas, tentam combiná-las para formar soluções ainda melhores. Um exemplo deste procedimento são os Algoritmos Genéticos [2] [38]. Os Algoritmos Genéticos são algoritmos de otimização estocástica, que trabalham de forma aleatória, orientada de acordo com regras probabilísticas e possuem um funcionamento semelhante à evolução humana [38].

Inicialmente, é criada uma população de indivíduos (genes) factíveis a serem solução. Em seguida, são realizados cruzamentos entre os indivíduos, através da permutação, que são trocas entre as soluções, além disso, são realizadas mutações nos indivíduos, em busca de melhorias. Com isso, a população inicial passa por um processo de seleção natural, no

qual somente os mais aptos, que possuem maior (maximização) ou menor (minimização) valor em uma equação de mérito são mantidos [38]. A Figura 4.3 apresenta o fluxo básico do processo de execução de um algoritmo genético.

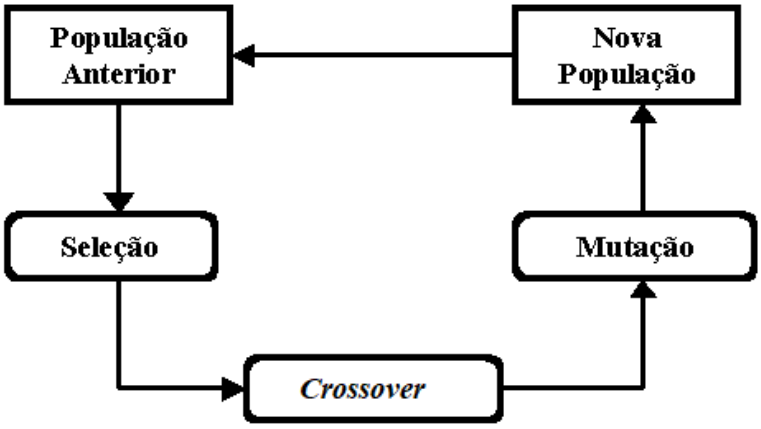


Figura 4.3: Exemplo de um Fluxo Básico da Execução de um Algoritmo Genético, adaptado de [84].

A utilização de algoritmos genéticos para a predição de recursos pode ser vista no trabalho de *Oliveira et al.* [24] no qual é construído um modelo para estimar a quantidade de recursos que são necessários para executar *workflows* paralelos. Eles são coordenados a partir de sistemas gerenciadores de *workflows*, em especial o SciCumulus [23]. Nesse sentido, cada indivíduo, foi definido como um tipo de recurso disponibilizado pelo provedor de nuvem [24].

Por outro lado,em metaheurísticas baseadas em busca local, a exploração das soluções é feita pelas movimentações a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Um exemplo desta categoria que se destaca na literatura é a metaheurística GRASP (do inglês, *Greed Randomized Adaptive Search Procedure*), que é um processo iterativo, no qual as iterações são independentes umas das outras e consiste de duas fases: construção e busca local [28], conforme apresentado no Algoritmo 1.

Feo e Resende [29] descreveram a metaheurística GRASP como um procedimento de busca local com recomeços, na qual cada iteração é composta por uma fase de construção e uma fase de busca local. A fase de construção combina técnicas gananciosas com seleção aleatória, para produzir um conjunto diversificado de soluções de boa qualidade que servem de ponto inicial para a fase de busca local.

A fase de construção tem por objetivo o desenvolvimento de uma solução factível para o problema original. Essa construção é feita elemento a elemento por uma função gulosa adaptativa, através de uma Lista Restrita de Candidatos – LRC. Cada iteração

define a escolha do próximo elemento que integrará a solução por meio da ordenação dos elementos restantes, da LRC. A função precisa medir o benefício da escolha de cada um dos elementos restantes, com o processo de construção adaptativo, pois, após escolher um certo elemento da lista, a solução deve ser adaptada, de forma a refletir o impacto da última escolha [28].

A segunda fase, definida como busca local, em cada iteração usa como base a solução inicial S, obtida na fase de construção. Tal solução não possui garantias de ser localmente ótima. Portanto, faz-se uma busca na estrutura da vizinhança de S, relativa à solução. Tal procedimento resulta em sucessivas trocas da solução atual, sempre que for encontrada uma melhor solução na vizinhança de S. Quando não são encontradas melhores soluções, o procedimento termina [2] [28].

Algoritmo 1: Pseudo Código da Metahueristica GRASP [28].

Entrada: $CriterioParada, semente$;

Saída: S^*;

```
1  custo* ← ∞;
2  enquanto NOT.CriterioParada faça
3  |   S ← ConstrucaoGulosaAleatoria(LRC);
4  |   S' ← BuscaLocal(S);
5  |   se Custo(S') < custo* então
6  |   |   S* ← S';
7  |   |   custo* ← Custo(S');
8  |   fim
9  fim
10 retornaS*;
```

A seção seguinte apresenta alguns trabalhos relacionados à predição e alocação de recursos no ambiente de nuvem computacional.

4.3 Trabalhos Relacionados

Vários autores empregam técnicas distintas para solucionar problemas específicos, como é relacionado no trabalho de Islam *et al.* [50], que objetiva analisar o problema da oferta de recursos do ponto de vista do provedor. Nesse trabalho, o objetivo é garantir que as aplicações em execução, possam tomar decisões autônomas de escala, avaliando o uso de

recursos futuros (por exemplo, CPU, memória, rede, requisições etc.). Para isso, nesse trabalho, foi desenvolvido um modelo de previsão baseado no histórico de execução das aplicações ao utilizar técnicas de redes neurais e regressão linear.

Por outro lado, o trabalho de Kiran *et al.* [56] apresenta uma solução de previsão do tempo de execução de tarefas em ambientes de *grid*, sendo este um dos fatores que definem o uso eficiente dos recursos nestes ambientes. Ao estimar o tempo de execução das tarefas é possível reduzir a espera nas filas de execução, e permitir o planejamento da alocação de recursos com antecedência. Nesse trabalho, foram utilizadas técnicas de análise estática do código fonte, *benchmarking* analítico de desempenho do ambiente e uma abordagem baseada em compilador, que extrai informações adicionais. A combinação dessas três técnicas formam o módulo de predição proposto pelos autores.

Kianpisheh *et al.* [55] propuseram uma estimativa de parâmetros de QoS em *workflows* executados em ambientes de *grid*. O objetivo foi aprimorar a eficácia da distribuição das tarefas, sendo que em ambientes heterogêneos de *grid*, componentes poderão ser desligados para economizar energia, reiniciados ou até sofrerem falha de hardware. Com isso, os autores propuseram, através de históricos de execução e um sistema multi-estado, estimar a confiabilidade e a probabilidade do *workflow* concluir sem falhas, além de definir custos para a execução.

Além disso, Coutinho *et al.* [20] investigaram a execução de *workflows* científicos em ambientes de nuvem federada com o intuito de minimizar o tempo de execução, controlar os custos financeiros e estimar recursos de acordo com a necessidade, tudo antes da execução de cada etapa do *workflow*. Os autores propõem uma ferramenta de predição que seja integrada a sistemas gerenciadores de *workflows* – SWfMS [66], os quais são sistemas que definem, gerenciam e executam *workflows* com a garantia de que as atividades ocorram na sequência definida. Além disso, utilizam uma solução proposta por Ogasawara *et al.* [79] que mapeia as atividades dentro do *workflow*, que são gerenciados por operadores algébricos que consomem e produzem conjuntos de tuplas (relações). Nesse caso, cada ativação irá despender recursos do ambiente, e com isso é possível estimar os recursos necessários para cada atividade. Para estimar recursos específicos para cada atividade, os autores adotaram a metaheurística GRASP. O algoritmo proposto considera uma lista de recursos disponíveis na nuvem para predizer qual será o recurso que melhor atende as atividades mapeadas ao considerar o custo financeiro em cada execução.

A partir do estudo dos trabalhos relacionados à predição de recursos em ambientes de nuvem computacional, foram levantadas as soluções para organização interna da infraestrutura de nuvem federada, como por exemplo a eficiência no provisionamento de recursos [50] [111], a eficiência na qualidade de serviço, a estimativa de recursos, e a minimização dos custos e do tempo de execução [20] [55].

Na Tabela 4.1 é apresentado um resumo de cada trabalho descrito anteriormente. Como pode ser observado, alguns trabalhos utilizam ambientes de nuvens e outros de *grid*. Além disso, as técnicas usadas variam desde técnicas de aprendizado de máquina até métodos estatísticos e metaheurísticas.

Todavia, nenhum desses trabalhos investe nos interesses dos usuários, tais como custo financeiro, tempo e *feedback* da execução do *workflow*, e nem na possibilidade do usuário poder escolher entre uma execução de baixo custo ou de alto desempenho. Além disso, notou-se que nenhum desses trabalhos faz a utilização do modelo de regressão múltipla, que possibilita utilizar estimativas com maior número de variáveis preditoras, e a com-

binação da utilização da metaheuristica, com intuito de obter soluções viáveis em tempo hábil.

Diante do exposto, este trabalho propõe uma abordagem chamada sPCR (Serviço de Predição de Custos e Recursos Computacionais), com o objetivo de dimensionar os recursos computacionais, o tempo e custo das execuções dos *workflows*. Esta abordagem será detalhada no Capítulo 5.

Tabela 4.1: Trabalhos Relacionados à Predição de Recursos.

Trabalhos	**Ambiente**	**Técnica**	**Custos**
Islam *et al.* (2012) [50]	Nuvem	Regressão Linear; Redes Neurais	-
Kiran *et al.* (2009) [56]	*Grid*	Análise do código fonte	-
Kianpisheh *et al.* (2014) [55]	*Grid*	Histórico de Execuções	-
Coutinho *et al.* (2015) [20]	Nuvens Federadas	GRASP	-
Esta dissertação	Nuvens Federadas	GRASP; Regressão Múltipla	X

Capítulo 5

Serviço de Predição para Nuvem Federada

Neste capítulo serão apresentados os métodos utilizados para a implementação do Serviço de Predição para nuvens federadas proposto neste trabalho. Para isso, a Seção 5.1 apresenta a visão geral do serviço proposto. Na Seção 5.2 mostrada a estrutura do serviço de predição. Além disso, a Seção 5.3 detalha todo o fluxo de execução do serviço. A seção 5.4 mostra os *workflows* utilizados como estudo de caso para essa dissertação. Em seguida, a Seção 5.5 apresenta a metodologia utilizada para a construção do modelo preditivo de Regressão Linear Múltipla. A Seção 5.6 mostra a validação do modelo preditivo de Regressão Linear Múltipla e a Seção 5.7 apresenta os resultados das estimativas dos tempos de execução dos *workflows*. Seguidamente, a Seção 5.8 mostra as definições e detalhes da implementação da metaheuristica GRASP e por último a Seção 5.9 traz os resultados alcançados por meio da metaheurística.

5.1 Visão Geral

O Serviço de Predição de Custos e Recursos Computacionais – sPCR proposto neste trabalho tem por objetivo 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.

A ideia é que este serviço possa auxiliar na escolha adequada dos recursos a serem alocados. A escolha a ser feita pelo usuário, com auxílio do serviço proposto, pode priorizar uma execução mais rápida, ou mais barata, de acordo com os interesses dos usuários.

Isso é importante em um ambiente de nuvem federada porque usuários demandam execuções de *workflows* de grande escala, alocando recursos e pagando somente pelo que usam. Todavia, estimar a quantidade necessária de recursos está longe de ser trivial. Assim, a má escolha na quantidade ou até mesmo na capacidade do recurso poderá produzir impactos negativos na execução do *workflow* e afetar o custo financeiro.

Essa tarefa de escolha de recursos é muito complexa, porque os provedores de nuvem oferecem uma ampla variedade de possibilidades (por exemplo baixa performance de CPU, alta performance de CPU, *clusters*, *clusters* de GPU etc), cada uma associada a um custo financeiro e desempenho.

A Figura 5.1 ilustra o cenário em que uma nuvem federada possui uma grande dimensão de recursos, visto que estão interconectados quatro provedores, sendo que cada provedor disponibiliza o seu conjunto de recursos. Nesse caso, é preciso uma estrategia eficiente, que forneça uma estimativa adequada para o usuário. Assim, muitas vezes os procedimentos de força bruta encontram soluções para esse cenário. Contudo, esses procedimentos são extremamente ineficientes, pois o tempo necessário para a escolha dos recursos é extremamente elevado, e cresce de maneira exponencial a diversidade de recursos, disponibilizada pelas nuvens. E para o método de predição funcionar o tempo gasto na escolha deve ser insignificante perante a execução total de um *workflow*. Dessa forma, o uso de heurísticas e metaheurísticas fornecem, em um tempo razoável, boas, e algumas vezes até ótimas soluções [20].

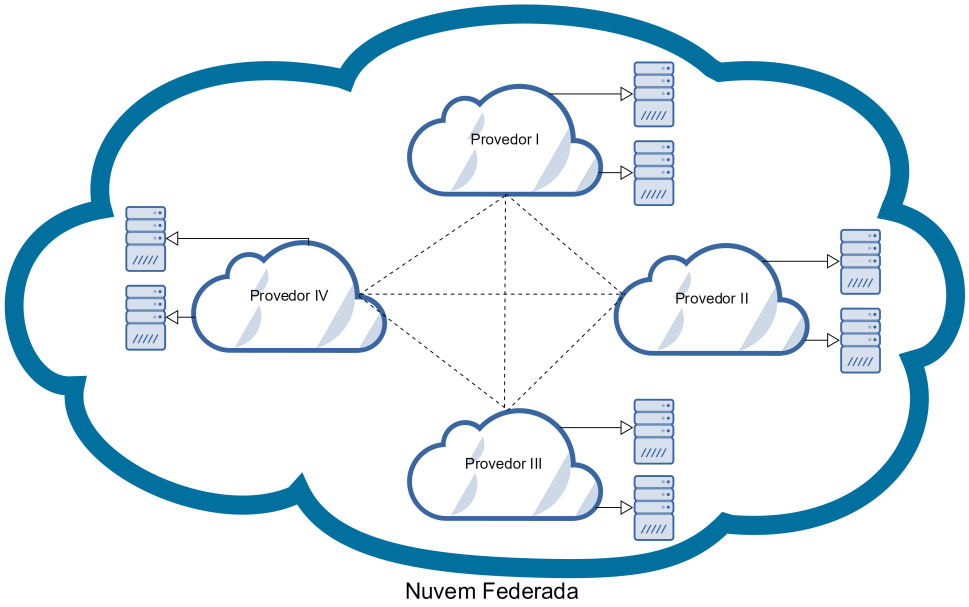


Figura 5.1: Representação de uma Nuvem Federada com Vários Recursos.

Contudo, em um ambiente de nuvens federadas, várias características devem ser consideradas para desenvolver um modelo de predição. Dentre essas características, destacam-se os variados tipos de máquinas virtuais oferecidas pelos provedores, o número máximo de máquinas virtuais que podem ser alocadas por um provedor específico, o custo da comunicação entre os recursos de diferentes nuvens que compõem o ambiente federado, o custo da transmissão de dados (*download, upload*), e o custo da transmissão de dados do armazenamento etc.

Dessa forma, neste trabalho foram consideradas as variáveis que envolvem as restrições do usuário e do ambiente de nuvem federada. Essas varáveis definem a estrutura do serviço de predição, e são baseadas no estudo de Coutinho *et al.* [19]. Assim, a lista de variáveis consideradas neste trabalho foi:

- Variáveis relacionadas ao usuário:
  - Custo financeiro máximo permitido para a execução do *workflow*;
  - Tempo máximo de duração da execução do *workflow*;

- Variáveis relacionadas ao ambiente de nuvem federada:
  - Conjunto de máquinas virtuais oferecidas por um provedor x;
  - Custo da alocação da máquina virtual x por um determinado período de tempo;
  - Capacidade de armazenamento da máquina virtual x;
  - Capacidade de memória RAM da máquina virtual x;
  - Poder de processamento da máquina virtual x;
  - Custo da comunicação entre máquinas virtuais de diferentes provedores;
  - Custo do *upload* para uma máquina virtual x;
  - Custo do *download* a partir de uma máquina virtual x;
  - Custo do armazenamento dos dados transmitidos;
  - Tamanho médio dos dados transmitidos;
  - Custo da comunicação entre máquinas virtuais de um mesmo provedor.

Essas informações implicam em diversas possibilidades, e para estimar uma solução viável é necessário aplicar heurísticas com o intuito de ter boas soluções em um tempo adequado. Assim, neste trabalho, foi utilizada uma abordagem baseada na metaheurística GRASP [28] para a escolha dos recursos. Esta abordagem foi escolhida porque é altamente eficiente para resolver problemas de alocação em *workflows* científicos e outros domínios, como no trabalho de Coutinho *et al.* [20].

Além da abordagem definida para a otimização da escolha dos recursos, este trabalho também implementou um método estatístico, uma regressão linear múltipla, que visa estimar o tempo de execução do *workflow* e, consequentemente, o custo financeiro. Assim, ao utilizar a regressão e o GRASP, é possível definir soluções com recursos compatíveis com as escolhas do usuário, tais como uma execução com um custo previamente conhecido, ou em um tempo máximo pré-definido de execução. Assim, a próxima seção apresenta detalhes do Serviço de Predição proposto.

5.2 Estrutura do sPCR

Como exposto na seção anterior, o sPCR tem como objetivo estimar recursos para a execução de *workflows* de bioinformática, com as estimativas de custo e de tempo, além de considerar os interesses dos usuários para esses dois últimos. Para isso, a estrutura do serviço de predição foi dividida em quatro fases, que são: Coleta de Informações, Pré-processamento, Processamento e *Feedback*, conforme apresentado na Figura 5.2.

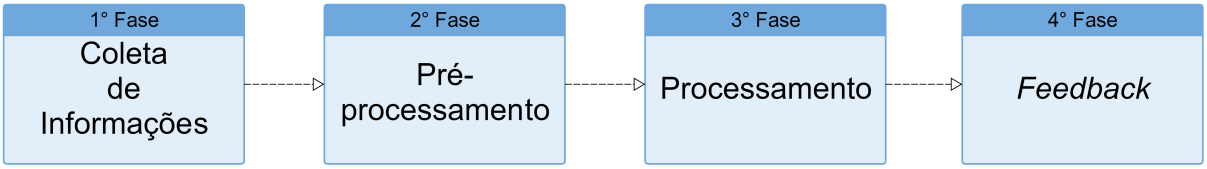


Figura 5.2: Fases de Execução do sPCR.

A 1° fase – Coleta de Informações consiste na busca das informações do *workflow* a ser executado, sendo informações fornecidas através de uma interface. O usuário deve inserir 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. Essas informações irão especificar a lista de recursos compatíveis com as restrições impostas.

A 2° fase – Pré-Processamento é composta pela base de dados histórica e pela comunicação com o serviço de tarifação do ambiente de nuvem. A base de dados histórica auxilia 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, na qual 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 etc.

A base de dados é utilizada pela regressão para construir um modelo estatístico confiável, que estime os custos e o tempo total de execução dos *workflows*. Neste contexto, é sabido que para previsões de softwares desconhecidos, as estimativas podem não ser confiáveis. Assim, esta fase também engloba mecanismos que criam, a partir de pequenas porções de dados de entrada dos *workflows*, denominadas de dados sintéticos, informações de execução do software. Essas execuções são replicadas em máquinas virtuais com um número diferente de núcleos. As execuções são adicionadas à base de dados histórica a fim de obter melhor acurácia no modelo estatístico. Em contrapartida, o serviço de tarifação da plataforma de nuvem deve informar a lista de recursos disponíveis, indicando a capacidade computacional de cada máquina virtal, os custos financeiros e os detalhes do provedor, tal como os indicados na lista das variáveis do ambiente de nuvem federada, descrita na Seção 5.1. 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 3° fase do processo de predição.

A 3° fase – Processamento é composta por um algoritmo baseado na abordagem GRASP, que consome as informações da fase anterior, com o objetivo de minimizar custos e tempo de execução, levando em consideração as opções descritas pelo usuário. O objetivo desta fase é disponibilizar soluções sub-ótimas, através da utilização da metaheurística, em tempo viável.

Após a construção da solução sub-ótima, o serviço de predição envia para a interface do usuário, a partir da 4° fase, o *feedback* da solução encontrada e as estimativas de tempo e custo da execução. Além destas informações, a 4° fase de *Feedback*, alimenta a base de dados histórica, objetivando registrar os dados da predição para garantir melhorias para as próxima predições.

5.3 Fluxo de Execução do sPCR

Todo o fluxo de processamento do sPCR é apresentado no diagrama da Figura 5.3, que explica detalhadamente cada etapa de sua execução. Assim, a Figura 5.3 apresenta desde a coleta dos dados do *workflow* até a disponibilização da lista de recursos, e as estimativas de tempo e custo financeiro.

Os elementos serão definidos através de passos, que são mostrados nas caixas azuis do diagrama. Inicialmente, na fase de Coleta de Informações, o usuário irá cadastrar na

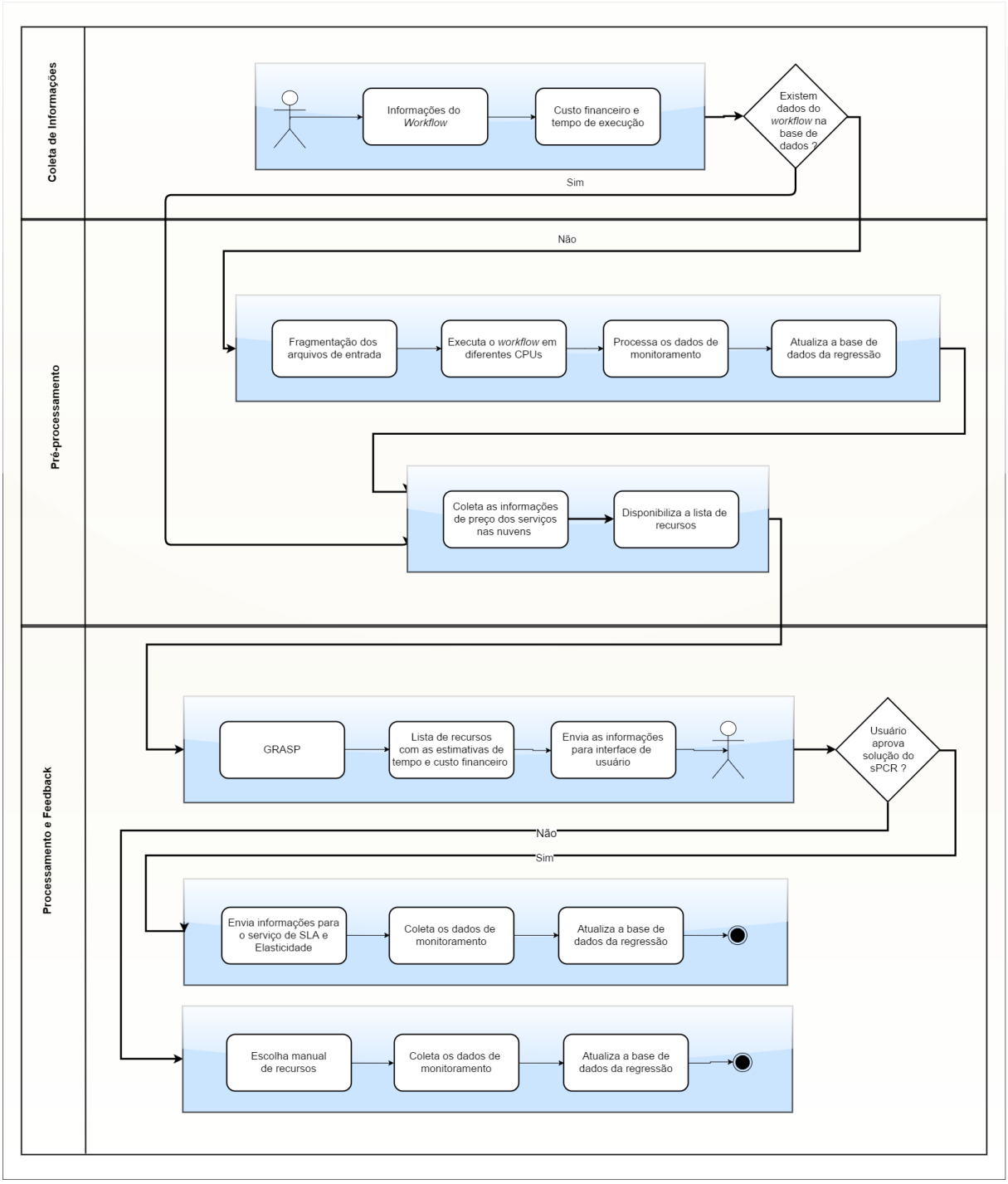


Figura 5.3: Fluxo de Processamento das Atividades do Serviço de Predição.

interface do ambiente de nuvem federada um *workflow* para execução. No cadastramento, devem ser especificados os arquivos de entrada e os softwares que compõem o *workflow*. Em seguida, o usuário escolhe entre uma execução de alto desempenho (α_1) ou baixo custo (α_2).

A primeira define execuções de menor tempo de execução, uma vez que serão priorizados recursos com maior poder de processamento e alto custo financeiro, a segunda opção define execuções com tempo mais elevado, sendo que são priorizados recursos de baixo desempenho e menor custo financeiro. A formulação matemática dos dois parâmetros é discutida na Seção 5.8. Além disso, o usuário, opcionalmente, pode definir máximos para o tempo de execução e/ou custo financeiro do *workflow*.

No último passo da etapa Coleta de Informações, é verificado se existem informações de execução na base de dados dos softwares que compõem o *workflow*, que decide qual passo iniciar na etapa de Pré-processamento.

Dessa forma, a fase de Pré-processamento é composta por dois fluxos de execuções, sendo que o primeiro considera que existem informações de execução dos softwares que compõe o *workflow*. Assim, o fluxo inicia com a criação da lista de máquinas virtuais disponíveis na federação. Em seguida, a lista de máquinas é processada para eliminar máquinas preemptivas, que se usadas podem incorrer em erros nas estimativas, pois o ambiente em questão poderia estar sendo compartilhado. Neste sentindo, o monitoramento das execuções sofreriam interferências em seus itens, tais como uso de CPU, memória, disco, e, principalmente, tempo de execução. Com isso, a fase de Pré-Processamento termina.

Considerando o fluxo alternativo, no qual os dados de execuções não estão disponíveis na base de dados, o mesmo tem por objetivo disponibilizá-los a partir da execução do *workflow* em uma versão reduzida dos dados de entrada. O fluxo inicia na fragmentação dos arquivos de entrada do *workflow*, reduzindo em pequeno número de *reads* para execuções em menor tempo. O tamanho do arquivo é definido a partir da quantidade de *reads* definidas empiricamente, por exemplo 50, 100 e 150 *reads*.

A próxima fase consiste em executar o *workflow* com os dados de entrada fragmentados em diferentes máquinas virtuais, também definidas empiricamente. Essa escolha é feita baseada no número de núcleos do processador, geralmente, 2, 4 e 8 núcleos. A escolha destes parâmetros parte da premissa de que o crescimento linear de ambos são fatores principais do tempo de execução do *workflow*, além de almejar mínimo *overhead* para finalização das estimativas. A Tabela 5.1 apresenta o experimento definido para o segundo passo, o que permitiu iniciar a construção da base de dados histórica.

Tabela 5.1: Quantidade de *Reads* e Núcleos de CPU para Execução.

Qnantidade de *Reads*	50.000	100.000	150.000
Nº de Núcleos da CPU	2, 4, 8	2, 4, 8	2, 4, 8

Cada execução do experimento foi monitorada, por meio da *Versatile Resource Statistics Tool – DSTAT* [110], uma ferramenta gratuita para monitoramento, disponível para sistemas operacionais baseados em Linux. A partir do monitoramento foi possível obter, em cada segundo, as informações de uso de CPU, de memória RAM, de disco rígido e de rede. Após cada execução, as informações de monitoramento foram tratadas e enviadas para a base de dados históricas, que permite obter estimativas do *workflow*, até então

desconhecidas. Em seguida, os passos para geração da lista de recursos utilizados pela fase de Processamento são executados, e a fase de Pré-processamento é finalizada.

A última fase, definida como *Feedback*, inicia com a execução do GRASP, que a partir da lista de recursos encontra um conjunto sub-ótimo de recursos para a execução do *workflow*. Além disso, atribui-se para cada solução o tempo de execução e o custo a partir da regressão utilizada. As definições da implementação do GRASP estão descritas na Seção 5.8. Assim, o próximo passo desta fase é enviar a solução encontrada pelo GRASP para a interface do usuário. Neste fluxo, o usuário avalia a solução proposta pela metaheurística, sendo que o usuário deve aprovar ou não a mesma.

Caso seja aprovada, as informações seguem para os Serviços de SLA e Elasticidade da plataforma de nuvem federada, para inicialização do *workflow*. Se não for aprovada, o usuário utilizará a escolha manual de recursos. Para ambos os casos, a execução será monitorada, assim como na fase de Pré-processamento, para alimentar a base de dados histórica, objetivando melhor acurácia. Com isso, esta fase é finalizada e a estimativa dos recursos é passada para o usuário.

Antes de definir o modelo de regressão para as estimativas de tempo e custo, e o modelo de predição de recursos, é necessário apresentar os *workflows* de bioinformática utilizados neste trabalho. Assim, a próxima seção apresenta esses *workflows*.

5.4 *Workflows* Executados

Neste trabalho foram executados dois *workflows* de bioinformática, com o objetivo de obter dados de suas execuções, que permitiram construir a base de dados para as estimativas de tempo. Os *workflows* executados foram:

- *Workflow 1:* Dados de experimento de RNA-Seq do fungo *Schizosaccharomyces pombe* [46].
- *Workflow 2:* Dados de experimento de RNA-Seq do fungo *Aspergillus furmigatus* [62].

O *workflow 1*, como ilustrado na Figura 5.4, é composto das fases de Montagem, Mapeamento e Anotação. As ferramentas utilizadas foram Trinity [43], Bowtie [60], TopHat [103] e Blast [14]. A fase de mapeamento é representada pela cor roxa, a montagem pela cor azul, e a anotação pela cor marrom e os dados de entrada são arquivos do tipo fasta e fastq contendo as *reads*.

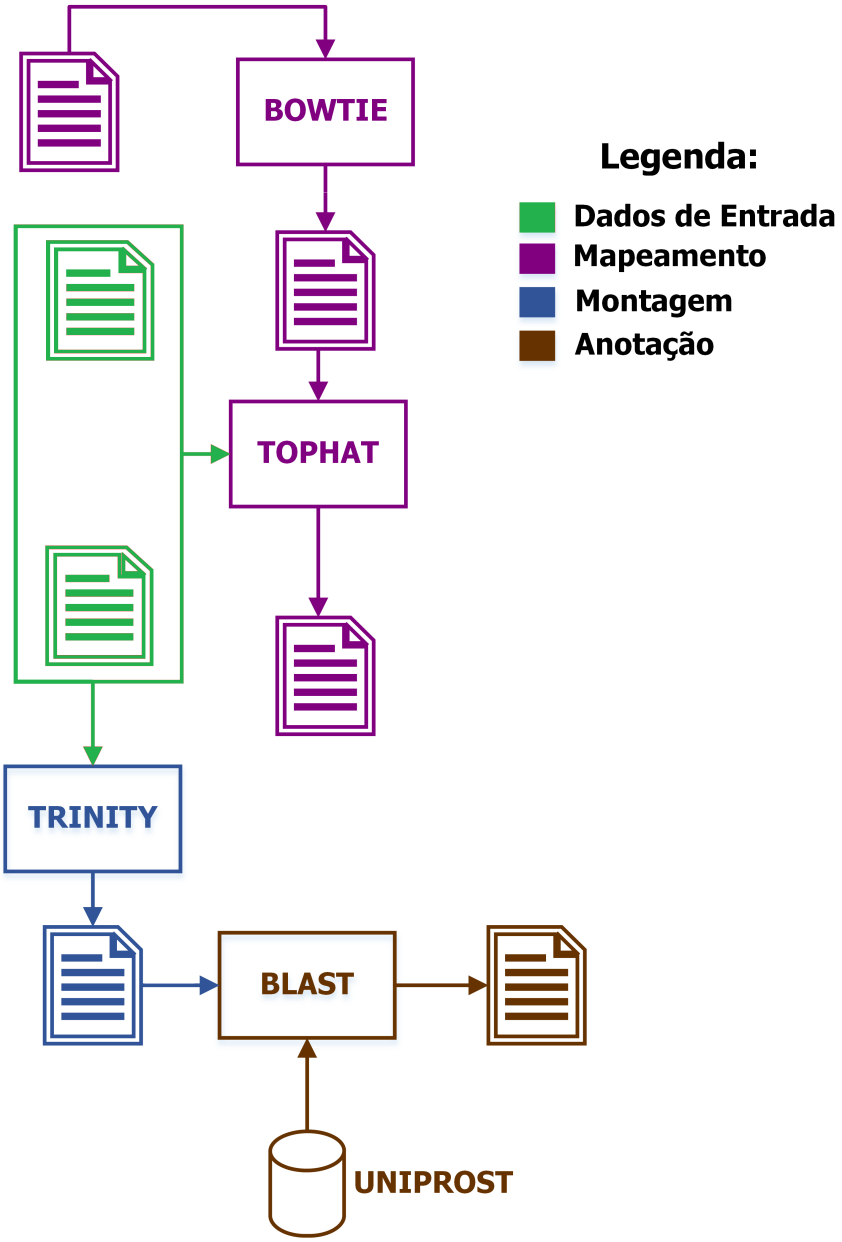


Figura 5.4: *Workflow 1* com as Fases de Mapeamento/Montagem e Anotação.

O *Workflow 2* possui as fases de Filtragem, Mapeamento, Montagem e Anotação. As ferramentas utilizadas para a execução foram Bowtie [60], TopHat [103], Trinity [44] e Blast [14]. A Figura 5.5 ilustra o *workflow 2*, nela os arquivos de entrada estão representados com a cor verde, o mapeamento em roxo, montagem em azul e anotação em marrom.

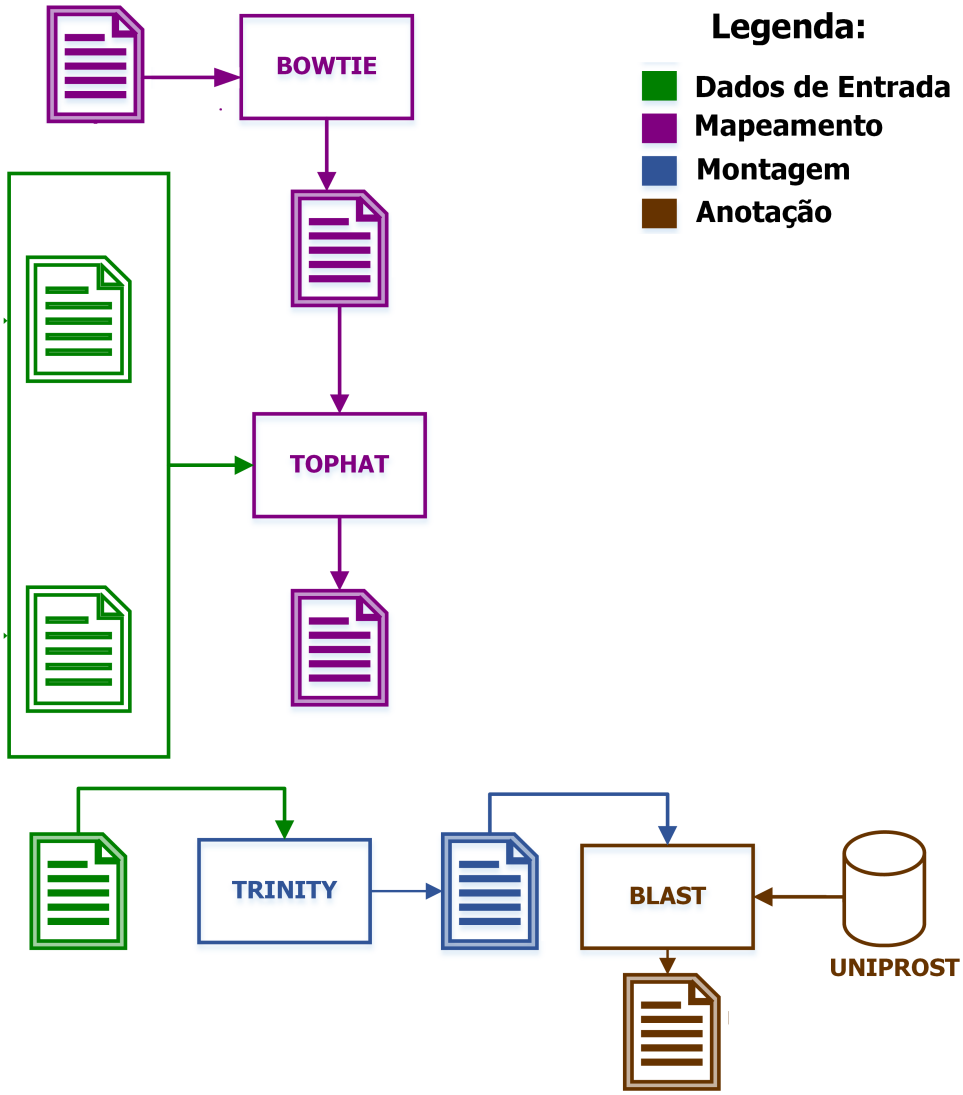


Figura 5.5: *Workflow 2* com as Fases de Mapeamento/Montagem e Anotação.

Após as execuções dos *workflows* foram coletados os dados de monitoramento que compuseram a base de dados utilizada para as estimativas.

As próximas seções apresentam as definições da implementação do método estatístico utilizado para estimar o tempo e o custo financeiro das execuções, e da metaheurística GRASP, para predição dos recursos a serem utilizados durante a execução.

5.5 Regressão Linear Múltipla

A Regressão Linear Múltipla – RLM faz parte do processo de estimativas definido no sPCR, especificada para estimar o tempo, e, consequentemente, o custo de um software contido em um *workflow*.

Para o entendimento do modelo usado, contudo, a metodologia foi dividida em fases. Primeiramente, foram elencadas as variáveis que vão compor o modelo. Durante a escolha dessas variáveis foram selecionadas empiricamente vinte variáveis, especificadas na Tabela

5.2, que *a priori* representavam a execução dos softwares. Neste contexto, foi necessário validar a base de dados, para a implementação do modelo de forma a obter menor erro na predição.

Em seguida, foi necessário definir um método para a escolha das variáveis que iriam compor a equação da regressão de forma eficiente. Um dos métodos que poderiam ser utilizados era o de força bruta, na qual são adicionadas e removidas cada uma das variáveis ao modelo de regressão, e verificado a partir de algum critério de seleção a eficiência (ou não) dessa escolha. Entretanto, o método se define oneroso [7]. Nesse sentido, foi implementado neste trabalho, a partir do software R[1] [85], o método semi-automático denominado *stepwise* [8]. O método *stepwise* é usado para selecionar quais variáveis mais influenciam no modelo e, assim, otimizar o número de variáveis a compor a equação de regressão.

Assim, o procedimento iterativamente constrói modelos de regressão pela adição ou remoção de variáveis em cada etapa, através de um critério expresso em termos de um teste parcial F, formado pela comparação de um modelo com o seu submodelo. Inicialmente, tem-se um modelo com uma variável preditora com a mais alta correlação com a variável resposta, de tal forma que a cada nova adição de variável é realizada uma verificação para eliminação de variáveis redundantes. A finalização do procedimento ocorre quando não há mais variáveis a serem incluídas ou removidas do modelo. Em alguns casos, é possível adicionar preditores transformados, no sentido de aumentar a força da relação, tais como $log(x)$. Assim sendo, as variáveis deste trabalho, estabelecidas pelo procedimento de *stepwise* para compor o modelo preditivo podem ser vistas na Tabela 5.3 [7] [8].

Nesse contexto, as transformações atribuídas às variáveis do modelo, foi aplicado o uso da função logarítmica e uma padronização. A primeira foi aplicada na variável $qntReads$ que representa a quantidade de *reads* em um arquivo, além de ter a suposição de que o tempo segue uma distribuição exponencial, logo, foi usada com a variável objetivo em $log(tempo)$.

A relevância da transformação é mostrada nos histogramas das Figuras 5.6 e 5.7. O primeiro histograma (Figura 5.6) mostra que o impacto das observações próximas a 0 é tão grande, que não é possível analisar precisamente qualquer diferença externa. Em outras palavras, a quantidade de observações próximas a 0 fazem as outras observações pouco relevantes. O segundo histograma (Figura 5.7) mostra como a transformação ajuda a balancear o peso de cada uma das observações, permitindo, assim, que a regressão consiga calcular com maior precisão a consequência daquela observação na variável objetivo. Por último, a variável $fatorCor = 4/qntCPU$ foi criada na fase de formulação do modelo, com o objetivo de interpretar melhor possíveis correlações entre as variáveis, visto que como as CPUs do conjunto avaliado variam entre 2 e 8, foi atribuído um valor intermediário. Neste caso, o valor 4, que faz o conjunto de CPUs variar em 0.5, 1 e 2, em vez de 2, 4, 8, ou seja, uma mudança de escala que não altera as propriedades estatísticas do modelo.

[1]R é uma linguagem e também um ambiente de desenvolvimento de código aberto integrado para cálculos estatísticos e gráficos.

Tabela 5.2: Variáveis de Monitoramento.

Variáveis	Descrição
time	Tempo de execução do software em segundos
usr	Uso de CPU pelos processos de usuário
sys	Uso de CPU pelos processos de sistema
idl	CPU ociosa
wai	Processos em espera
used	Uso de memória
buff	Uso em *buffer*
cach	Uso em cache
free	Memória livre
read	Operações de leitura em disco
writ	Operações de escrita em disco
recv	*Bytes* recebidos pela rede
send	*Bytes* enviados pela rede
run	Processos em execução
blk	Processos bloqueados
new	Processos novos
programa	Software contido no *workflow*
qntReads	Quantidade de *reads* de um arquivo
qntCPU	Número de CPUs
tamArq	Tamanho em MB do arquivo de entrada do software

Tabela 5.3: Variáveis do Modelo Preditivo após o Método do *Stepwise*.

Variáveis	Descrição
time	Tempo a ser estimado
qntReads	Quantidade de *reads* de um arquivo
fatorCor	Número de CPUs
programa	Software contido no *workflow*

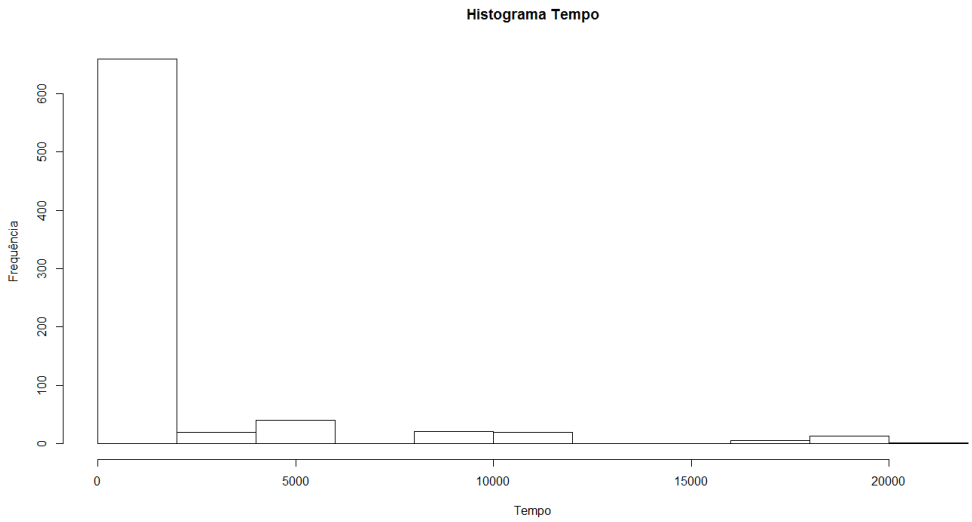


Figura 5.6: Histograma da Variável Tempo.

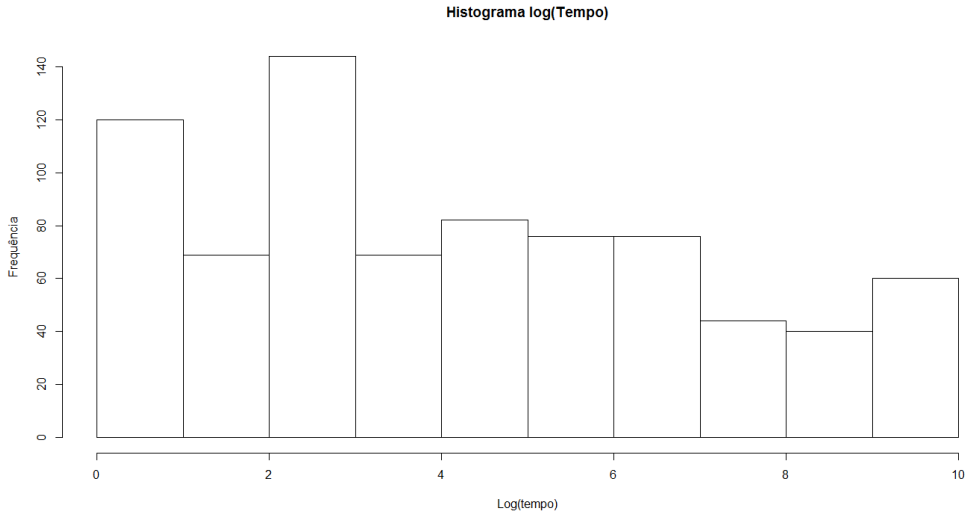


Figura 5.7: Histograma da Variável Tempo após a Transformação Logarítmica.

Após a escolha das variáveis, a próxima etapa foi definir o conjunto de dados para o projeto de experimento aplicado no modelo. Para isso, foram definidos os *workflows* de bioinformática utilizados neste contexto, o quais foram apresentados na Seção 5.4.

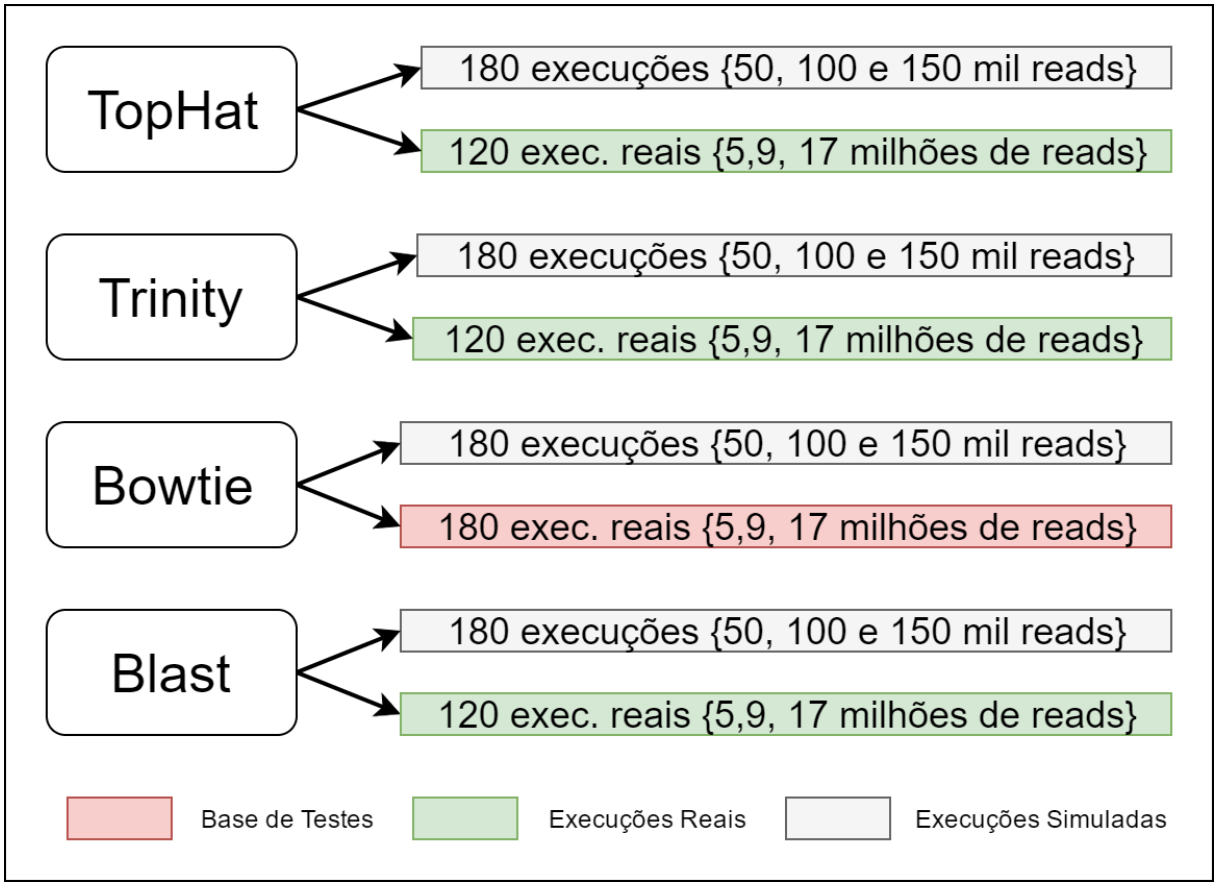


Figura 5.8: Projeto de Experimento dos Dados de Monitoramento.

Nesse contexto, a Figura 5.8 apresenta o projeto de experimento usado para preenchimento da base histórica. Esse projeto de experimento foi elaborado de acordo com os dados de monitoramento das execuções dos dois *workflows*. Nos experimentos realizados cada execução do programa é denominada como observação, ou seja, um item do conjunto de dados monitorados. Dessa forma, foram selecionados para a base de treinamento de maneira aleatória 180 execuções simuladas para cada programa ($TopHat$, $Trinity$, $Bowtie$ e $Blast$). É importante mostrar que as execuções simuladas são aquelas geradas em um primeiro momento, quando não existem dados de execução para o programa envolvido.

Além dessas, foram adicionadas observações de execuções reais, sendo 120 para cada um dos programas ($TopHat$, $Trinity$ e $Blast$). Além da base de treinamento, 120 observações reais de execuções do programa $Bowtie$ foram utilizadas para a validação do modelo preditivo. Com isso, é possível avaliar o desempenho do modelo preditivo, ao considerar que o programa $Bowtie$ não é conhecido a partir de execuções reais, uma vez que os dados de treinamento disponibilizam somente observações simuladas deste programa. O programa $Bowtie$ foi escolhido de forma empírica entre os programas utilizados para este projeto de experimento.

Em seguida, foi obtida a Equação 5.1 de regressão linear múltipla, na qual o *tempo* é definido como variável dependente ou preditando, e as demais ($qntReads$, $fatorCor$ e $programa$) como variáveis independentes ou preditoras.

$$tempo = \beta_0 + \beta_1 * log(qntReads) + \beta_2 * fatorCor + \beta_3 * programa \tag{5.1}$$

Os fatores $\beta_i s$, são coeficientes de regressão e β_0 é a constante. Esses são determinados para que a soma dos quadrados dos erros seja mínima, conforme indicada na Equação 5.2.

$$min\left(\sum_{j=1}^{n}(\hat{y}_j - y_j)^2\right) \tag{5.2}$$

Para avaliar a significância do modelo, um conjunto de técnicas estatísticas foi utilizado, como por exemplo, coeficiente de determinação ajustado $R_a^2 j$. Esse coeficiente é a medida de controle que leva em consideração tanto a variabilidade de y, que é explicada pelo modelo quanto o número de variáveis de controle utilizado [108]. Assim, tem-se que $0 \leq R_a^2 j \leq 1$, onde a proporção da variação de Y explicada pelo modelo é no máximo 1 e no mínimo 0.

Além da análise explicativa do modelo, é necessário realizar a Análise dos Resíduos [2], com objetivo de verificar o erro em Y não explicado pelas variáveis preditoras. Assim, quanto menor o resíduo melhor é a modelagem de Y a partir das variáveis preditoras. Dessa forma, supõe-se que os erros do modelo sejam normais e independentemente distribuídos, com desvio-padrão constante e média zero. Contudo, para validar o modelo é necessário verificar se estas condições são atendidas. Para isso, utiliza-se a análise residual, a qual é apresentada na Seção 5.6.

A próxima seção apresenta os resultados obtidos a partir da regressão linear múltipla.

[2] A Análise de Resíduos consiste em um conjunto de técnicas para investigar a adequabilidade do modelo com base nos resíduos.

5.6 Resultados da Regressão Linear Múltipla

Nesta seção são apresentados os resultados e a análise dos dados obtidos por meio do serviço de predição proposto. Para validar o modelo preditivo, foi verificada a ocorrência de multicolinearidade, que refere-se à independência entre as variáveis de controle do modelo de regressão, já que quando a correlação entre as variáveis é significativa, as inferências para o modelo de regressão são errôneas. O método utilizado foi o Fator de Inflação de Variância – FIV [67], que por meio Tabela 5.4 é possível analisar a inexistência da multicolinearidade, uma vez que os respectivos valores são menores do que cinco, indicando baixa correlação.

Tabela 5.4: FIV Indicando a Ausência de Correlação entre as Variáveis

Variáveis	FIV
log(qntReads)	1.15
fatorCor	1.00
programa	1.16

Além desses fatores, como dito anteriormente, também foram realizadas as análises dos resíduos. Inicialmente, os gráficos da Figura 5.9 apresentam técnicas de análise do erro, entre o predito e o real. A validação depende de quais erros são normais e independentemente distribuídos, com média zero e desvio-padrão constante, pois assim o modelo de regressão possui boa acurácia. Cada ponto dos gráficos é uma observação, ou seja, uma execução monitorada e registrada na base de dados histórica.

O gráfico da Figura 5.9 (a) não apresenta ter uma relação sistemática entre as observações, ou seja, estão aleatoriamente distribuídos, implicando em erros distribuídos de forma aleatória. A representação do eixo x é o número de observações na base de dados, e o eixo y traz os resíduos, ou o erro.

O gráfico da Figura 5.9 (b), mostra que a maioria das observações encontram-se sobre a linha da distribuição normal, tendo apenas uma pequena variação no começo e no final da cauda, ou seja, há indícios de que os resíduos estão normalmente distribuídos em torno da reta. O gráfico também mostra que as observações não tendem a seguir um padrão, que neste caso, é o ideal para o modelo, ou seja, as observações estão distribuídas aleatoriamente em torno da reta.

O gráfico da Figura 5.9 (c), apresenta em seu eixo x os valores preditos pelo modelo, e no eixo y os respectivos erros em relação ao valores reais, das observações de execuções contidas da base de dados. O mesmo mostra a distribuição do erro em relação ao conjunto de dados testes que foi utilizado. Pode-se observar, entre os valores preditos 6 e 8, uma espécie de agrupamento. Entretanto, esse agrupamento é justificado pelas observações destinadas ao teste, ou seja, observações que foram replicadas.

Os gráficos (d), (e) e (f) da Figura 5.9 indicam que os resíduos estão distribuídos de forma aleatória entre a *quantidadedereads*, as categorias do $fatorCor$ e os *programas*. No gráfico da Figura 5.9 (d) o eixo x mostra as observações da quantidade de *reads* categorizadas de acordo com seu valor na base, já o eixo y o erro calculado a partir da quantidade de *reads*. O gráfico da Figura 5.9 (e) apresenta os valores das observações no eixo x, sendo observados somente três categorias, uma vez que a base contém 3 diferentes

CPUs (2, 4 e 8). Por último, o gráfico da Figura 5.9 (f), apresenta no eixo x os programas observados em relação ao seu tempo de execução.

Além da análise gráfica para validação do modelo, também são utilizados índices de avaliação tais como o Bias, o Erro Absoluto Médio – EAM (MAE, em inglês), a Raiz do Erro Médio Quadrático – REMQ (RMSE, em inglês) e por último o Erro Percentual Médio Absoluto – EPMQ (MAPE, em inglês) [4].

O Bias, também conhecido como erro médio, mostra o desvio médio do modelo em relação à uma variável. Além disso, ele oferece informações de *performance* do modelo em um tempo longo e o erro sistemático. Ele também apresenta resultados positivos ou negativos, sendo que quanto mais próximo do zero, melhor o resultado. O Bias é calculado de acordo com a Equação 5.3, no qual N é o número de observações, $\hat{y}$ é o valor predito pelo modelo, e y é o valor real observador durante a execução.

$$Bias = \frac{1}{N}\sum_{n=1}^{N}(\hat{y} - y) \tag{5.3}$$

O EAM representa o erro médio absoluto da diferença entre a previsão e a observação. Quanto mais próximo de zero, melhor o acerto do modelo, além de medir o erro total. A Equação 5.4 apresenta a formulação do EAM, e nela o N é o número de observações, $\hat{y}$ é o valor predito pelo modelo e y é o valor real observado durante a execução.

$$EAM = \frac{1}{N}\sum_{n=1}^{N}|\hat{y} - y| \tag{5.4}$$

O REMQ, mostrado na Equação 5.5, consiste nas diferenças individuais entre a previsão do modelo e as observações. Ele mede os erros sistemáticos e randômicos, ou seja, é utilizado para medir a magnitude do erro. Nesse caso, N é o número de observações, $\hat{y}$ é o valor predito pelo modelo e y é o valor real observado durante a execução.

$$REMQ = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y} - y)^2} \tag{5.5}$$

E por último, o EPMQ que é mostrado na Equação 5.6 e expressa o erro absoluto médio em termos percentuais, para que se tenha uma visão do erro comparado com o valor previsto. Assim, N é o número de observações, $\hat{y}$ é o valor predito pelo modelo e y é o valor real observado durante a execução.

$$EPMQ = 100\frac{\sum_{n=1}^{N}|\frac{\hat{y}-y}{y}|}{N} \tag{5.6}$$

Como pode ser notado na Tabela 5.5, o modelo de predição proposto tem um desempenho satisfatório, pois os índices Bias, MAE, REMQ estão próximos de 0, ou seja, a acurácia dos índices dependem de quanto o erro está próximo de zero, uma vez que o mesmo é representado pelo valor predito menos o valor real. Já os cálculos do EPMQ

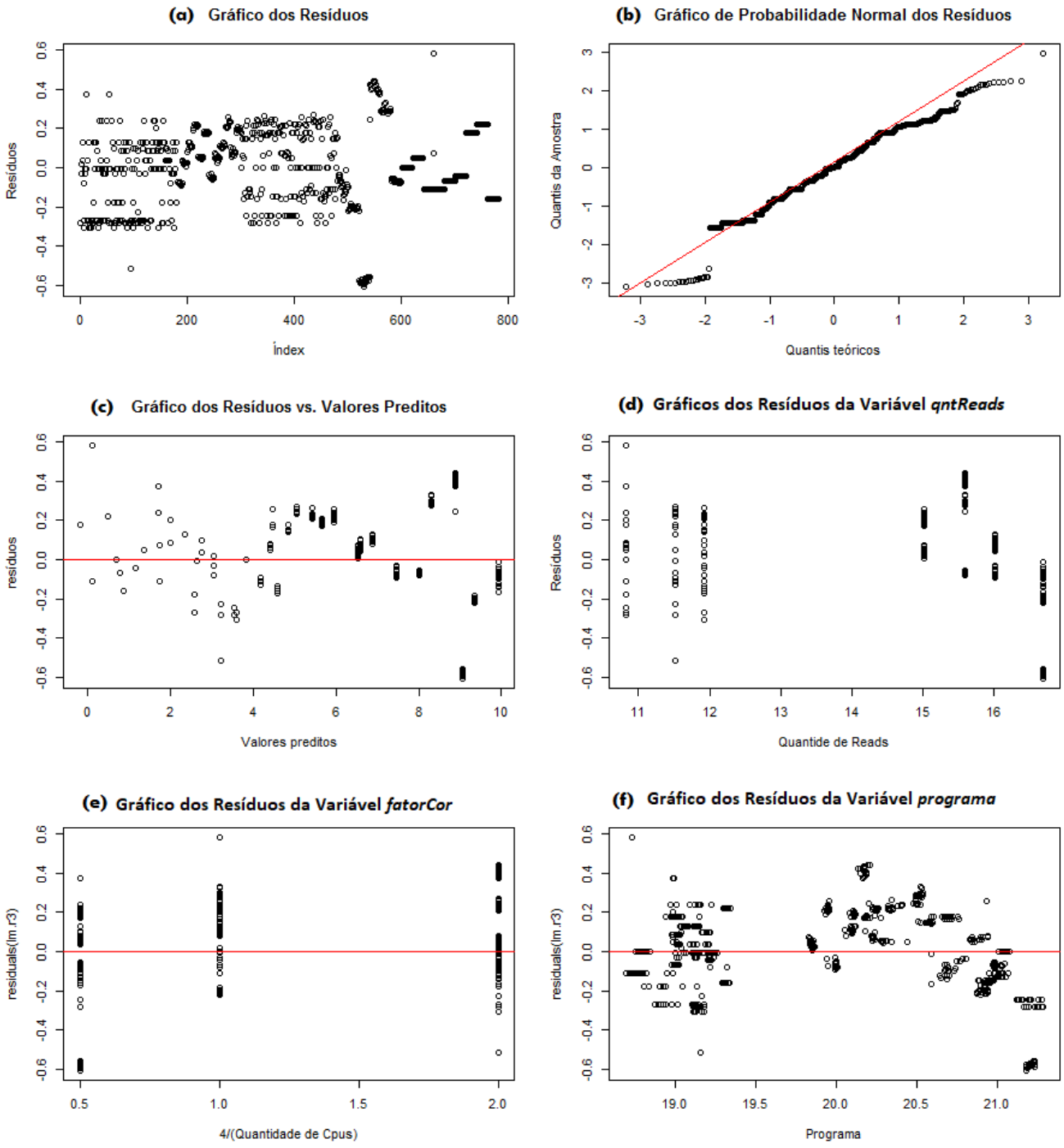


Figura 5.9: Gráficos dos Resíduos.

mostram que em média, as previsões do conjunto de testes estão incorretas em apenas $3,46\%$ dos casos.

Tabela 5.5: Resultado dos Índices de Avaliação.

Índices de Avaliação	Descrição
Bias	0.08
MAE	0.16
REMQ	0.20
EPMQ	3.46%
$R_a^2 j$	99.5%

Logo, tem-se informações suficientes para concluir que o modelo de regressão proposto neste trabalho pode ser utilizado como ferramenta para as estimativas de tempo de execução dos *workflows*. A próxima seção apresenta os resultados alcançados junto às estimativas.

5.7 Testes para Avaliar as Estimativas de Tempo

Nesta seção serão avaliados os resultados obtidos utilizando o modelo de Regressão Linear Múltipla para estimativas de tempo, a partir dos dados do projeto de experimento adotado.

As estimativas podem ser descritas em dois cenários. O primeiro ilustra o fluxo onde já existem informações de todos os programas envolvidos no *workflow* na base de dados histórica. Neste sentido, este cenário elimina a necessidade de obter infamações a partir de execuções simuladas. O segundo cenário ilustra a condição em que não existem informações de algum programa contido no *workflow* na base de dados histórica, sendo assim, é necessário instanciar o fluxo de execuções simuladas para preencher a base de dados com informações do programa a partir de pequenas execuções, auxiliando na acurácia das estimativas.

Vale ressaltar, nesse caso, que os testes foram distribuídos em três máquinas virtuais distintas do provedor Google, sendo que suas configurações foram:

- CPU: 2, 4 e 8;
- Memória: 7 *GigaBytes* em ambas;
- Disco: 60 *GigaBytes* em ambas;
- S.O. Ubuntu 14.04 x64;

As execuções foram replicadas vinte vezes, sendo que ao todo são 60 observações para cada CPU. Os testes realizados para avaliar as estimativas de tempo, são apresentados nas Seções 5.7.1 e 5.7.2.

5.7.1 Cenário 1 – Software Conhecido na Base

Neste cenário, a base de dados contém informações de execução do software, sendo possível estimar o tempo de execução instantaneamente. Diante disso, as Figuras 5.10, 5.11 e 5.12

exibem os resultados da regressão, junto às estimativas de tempo dos programas TopHat, Trinity e Blast, respectivamente.

O eixo x representa o tempo de execução em minutos, e o eixo y representa o número de observações de teste. Como pode ser observado, o tempo de execução real do programa é representado pela linha azul, e o tempo estimado pela linha laranja, onde as observações compreendidas entre 1 e 20, são execuções em uma máquina com 2 núcleos, as observações no intervalo 21 à 40 em máquina de 4 núcleos e por último, as observações de 41 à 60 são em uma máquina com 8 núcleos. Nesse sentido, tem-se uma aproximação com erro médio de 30 segundos para o TopHat, 7.07 minutos para o Trinity e 3.29 minutos para o Blast. Entretanto, ao observar os tempos totais, tem-se que o erro é razoavelmente bom. Note que a validação da acurácia do modelo é realizada na etapa de verificação dos resíduos, discutida na Seção 5.6.

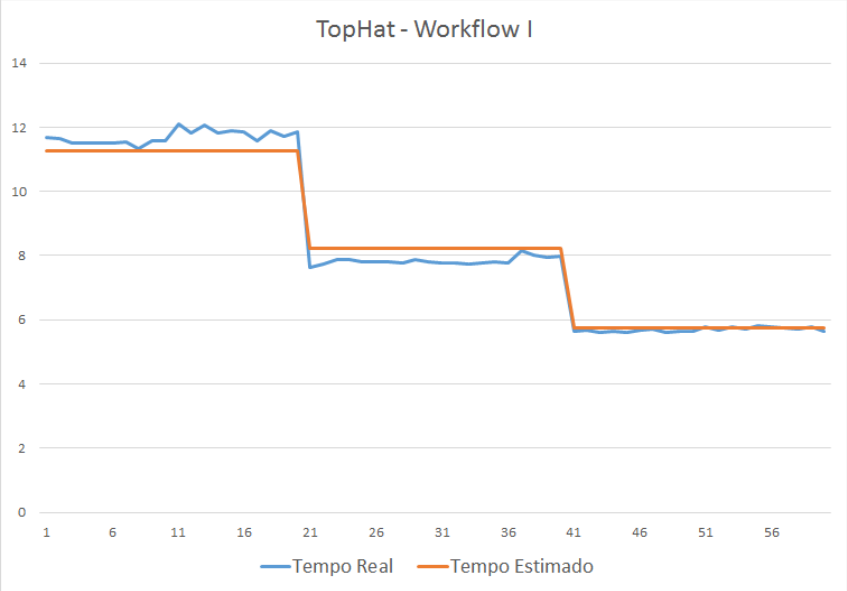


Figura 5.10: Tempo Real e Tempo Predito do Programa TopHat – *Workflow 1*.

Os resultados para o *workflow 2* são apresentados nas Figuras 5.13, 5.14 e 5.15. Nessas figuras tem-se que o eixo x representa o tempo de execução em minutos, e o eixo y representa o número de observações de teste. Como pode ser observado, o tempo de execução real do programa é representado pela linha azul, e o tempo estimado pela linha laranja, onde as observações compreendidas entre 1 e 20 são execuções em uma máquina com 2 núcleos, as observações 21 à 40 em máquina de 4 núcleos, e, por último, as observações 41 à 60 em uma máquina com 8 núcleos. Dessa forma, nota-se que há taxas de erros de 68 segundos para o TopHat, 12.40 minutos para o Trinity e 7.37 minutos para o Blast. Logo, assim como no *workflow 1*, tem-se um erro razoável ao comparar os tempos totais de execução, por exemplo, o programa Trinity, que possui tempo máximo de execução em aproximadamente 350 minutos em uma máquina de 2 núcleos.

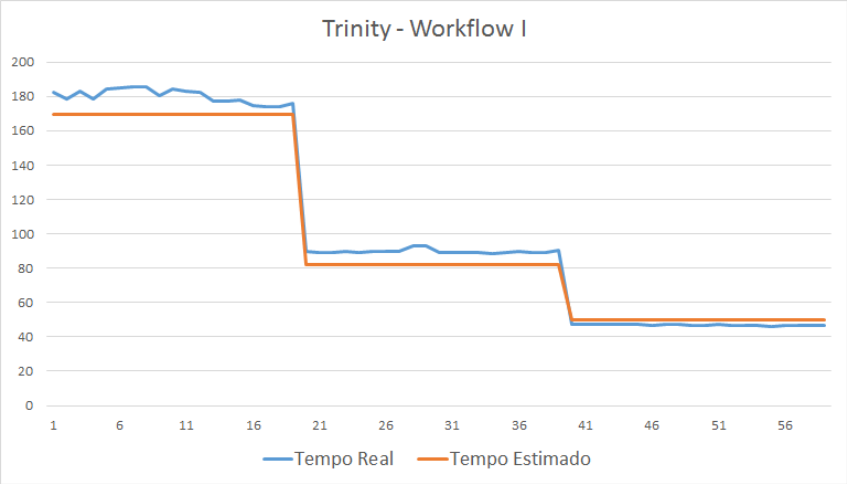


Figura 5.11: Tempo Real e Tempo Predito do programa Trinity – *Workflow 1.*

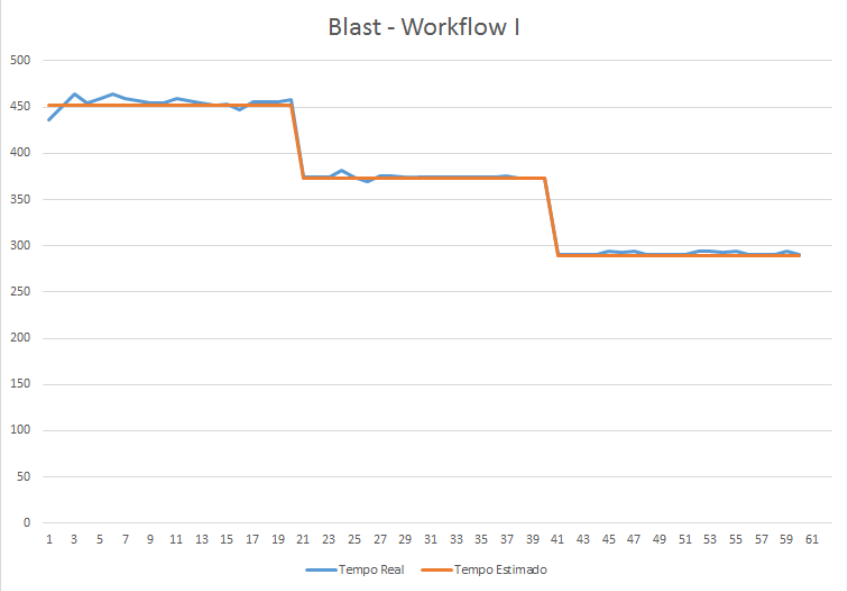


Figura 5.12: Tempo Real e Tempo Predito do Programa Blast – *Workflow 1.*

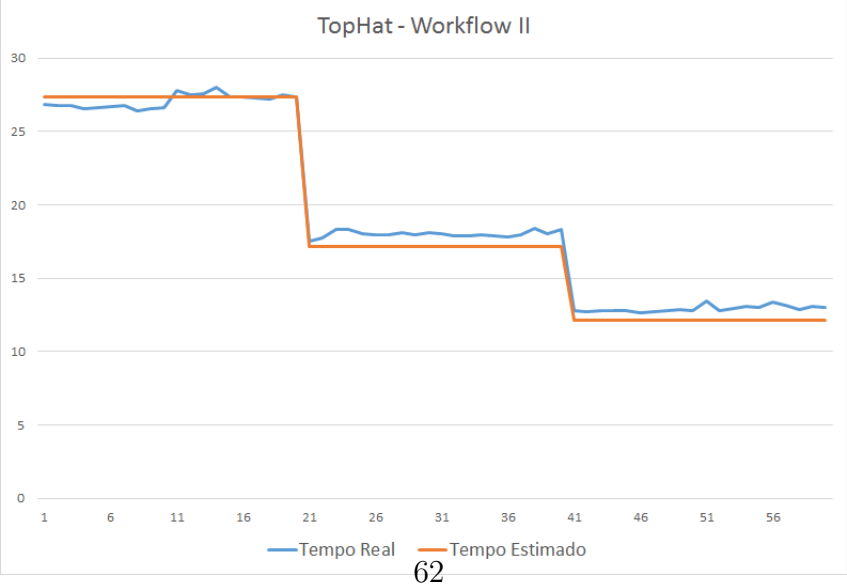


Figura 5.13: Tempo Real e Tempo Predito do Programa TopHat – *Workflow 2.*

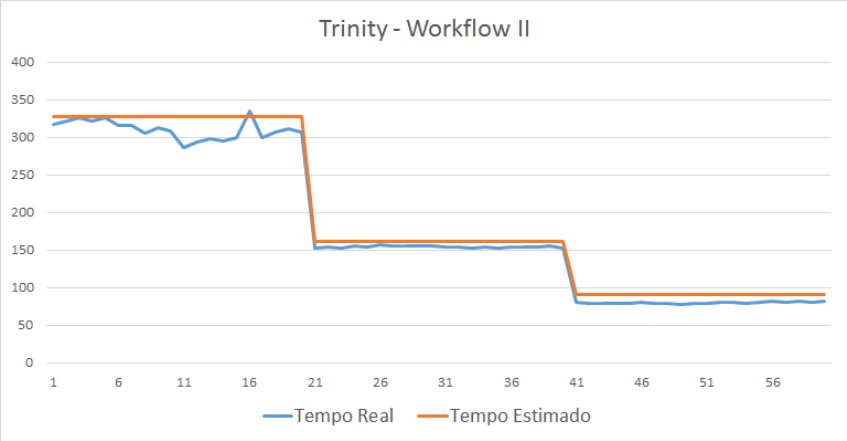


Figura 5.14: Tempo Real e Tempo Predito do Programa Trinity – *Workflow 2*.

.

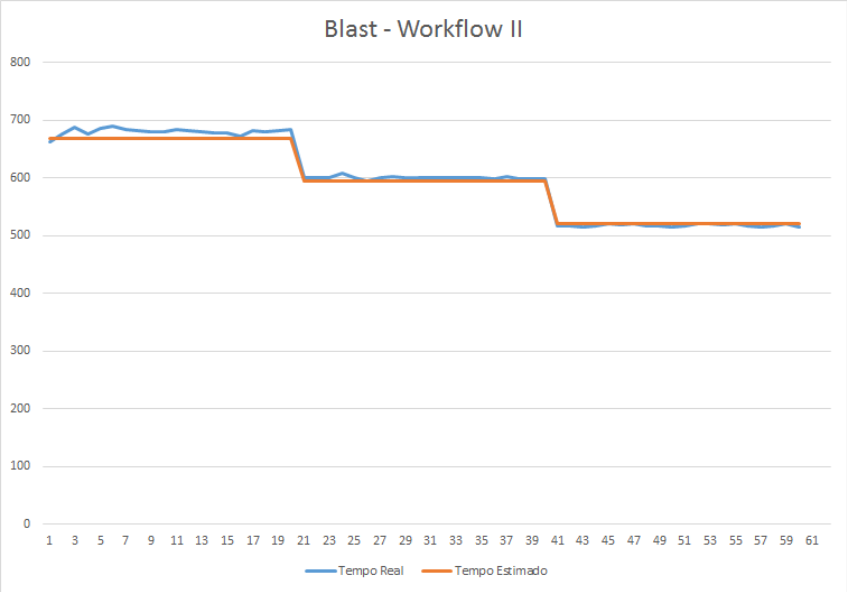


Figura 5.15: Tempo Real e Tempo Predito do Programa Blast – *Workflow 2*.

É importante ressaltar que não foram exibidos os dados do *Bowtie*, mesmo ele pertencendo aos dois *workflows*. Isso é justificado pelo fato do mesmo ter sido escolhido empiricamente para ser incluído ao cenário em que o software ainda não é conhecido pelo modelo de regressão, que será detalhado na Seção 5.7.2.

5.7.2 Cenário 2 – Software Desconhecido na Base

Neste cenário o modelo de regressão desconhece valores reais das execuções dos programas contidos no *workflow*. Entretanto, como dito na Seção 5, são realizadas execuções simuladas com parte do arquivo de entrada, objetivando diminuir o erro da predição.

Para isso, inicialmente, foram adicionadas à base 180 observações de execuções simuladas, sendo essas divididas entre 50.000, 100.000 e 150.000 *reads*, para cada uma das

máquinas citada na Seção 5.7.1. Além das observações simuladas, foram replicadas mais 180 observações de execuções reais para a validação da proposta deste cenário.

Diante dos dados apresentados nas Figuras 5.16 e 5.17, o tempo predito implicou em erro médio de 6 segundos para o *workflow 1*, e de 9 segundos para o *workflow 2*. Nesse contexto, é possível dizer que a criação de uma base simulada é relevante para o modelo preditivo, uma vez que as execuções simuladas expressam o comportamento do programa, mesmo com execuções de menor tempo. É importante ressaltar que este procedimento ocasiona *overhead* de resposta, ou seja, para a geração das 180 observações das execuções simuladas, foram demandados 442 segundos. Entretanto, o valor é irrisório quando observado o tempo total de execução de alguns *workflows* de bioinformática que, geralmente, apresentam tempo de execução em torno de horas.

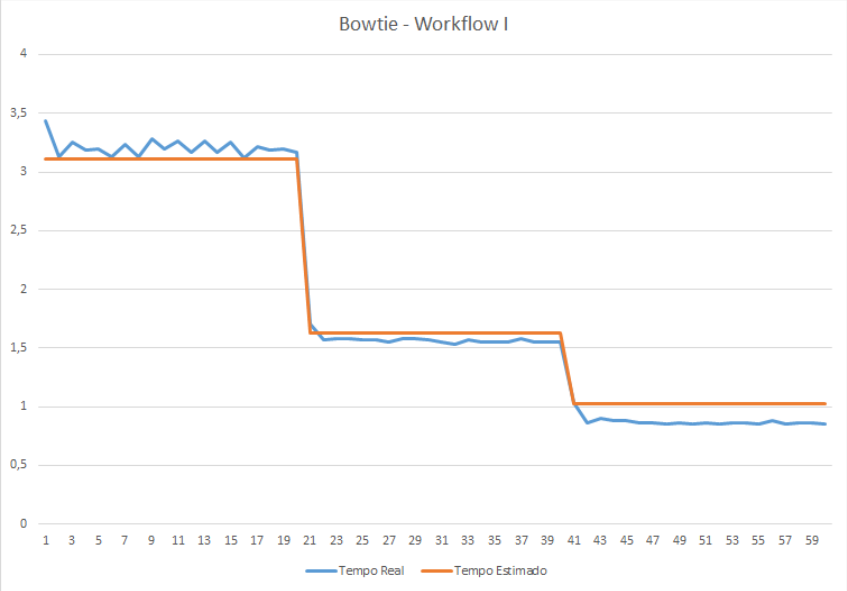


Figura 5.16: Tempo Real e Tempo Predito do Programa Bowtie – *Workflow 1*.

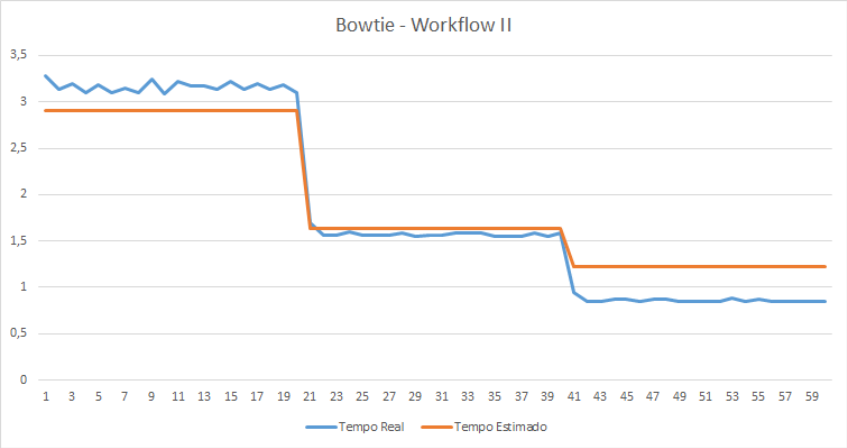


Figura 5.17: Tempo Real e Tempo Predito do Programa Bowtie – *Workflow 2*.

A próxima Seção apresenta a implementação da metaheurística GRASP utilizada para predição dos recursos.

5.8 Metaheurística GRASP

Como visto na Seção 4, este trabalho adotou o uso da metaheurística GRASP para a predição do dimensionamento dos recursos, no Serviço de Predição proposto. Neste sentido, para descrever o método, são necessárias algumas notações para a alocação das máquinas virtuais.

A notação para atribuição de máquinas virtuais utilizada é uma adaptação do trabalho de Coutinho *et al.* [19]. Assim, considere que P é o conjunto de todas as máquinas virtuais de um determinado provedor j. Deve-se considerar também os requisitos de usuários, tais como o custo financeiro máximo C_m e o tempo máximo de execução T_m. Além dos requisitos relacionados ao *workflow*, tais como a capacidade necessária de armazenamento em disco D, a capacidade de memória M, e o poder de processamento necessário N_c.

Da mesma forma, cada tipo de máquina virtual $p \in P$ possui um custo financeiro associado c_p, ou seja, o custo de alocação de uma máquina virtual por um determinado período de tempo pode variar entre 1min ou 1h, dependendo do provedor de nuvem, e tem recursos de computação como armazenamento em disco d_p, capacidade de memória m_p, quantidade de núcleos n_p.

Deve ser considerado também o custo do *upload* uc_p, *download* dc_p de uma máquina virtual, o custo do armazenamento dos dados transmitidos cd_p e o limite máximo de máquinas virtuais N_h que podem ser alocadas por usuários, variando de acordo com o provedor de nuvem. Dessa forma, a notação usada neste trabalho é apresentada na Tabela 5.6. Essas notações são utilizadas para definir o método e a função de custo, também chamada de função objetivo, que será apresentada por meio da Equação 5.7.

Tabela 5.6: Notação para Ambiente de Nuvem Federada.

Notação	Descrição
P_j	Conjunto de tipos de máquians virtuais oferecidas pelo provedor $_j$
C_m	Custo máximo financeiro
T_m	Tempo máximo de execução
D	Capacidade de armazenamento em disco
M	Capacidade de memória
N_c	Poder de processamento da máquina virtual
c_p	Custo da máquina virtual $_p$
d_p	Capacidade de armazenamento em disco da máquina virtual p
m_p	Capacidade de memória da máquina virtual $_p$
n_p	Quantidade de núcleos da máquina virtual $_p$
N_h	Limite de máquinas virtuais alocadas por usuário
uc_p	Custo do upload para uma máquina virtual $_p$
dc_p	Custo do dowload a partir de uma máquina virtual $_p$
cd_p	Custo do armazenamento dos dados transmitidos
L_s	Lista de softwares utilizados no *workflow*

O método GRASP, conforme indicado no Algoritmo 2, é composto por duas fases: a fase de construção – *construction* (veja Algoritmo 3) e a fase de busca local (veja o Algoritmo 4). A fase de construção encontra uma solução inicial que pode ser melhorada na fase de busca local.

Para descrever o método é necessário uma notação adicional. Para isso, definiu-se uma solução $\{(p_1, i_1), (p_2, i_2), ...\}$ como um conjunto de 2 tuplas (p, i), representando uma máquina virtual i do tipo p. Ditemos que uma solução $s \in S$ é factível se s não viola os requisitos de usuário. Também foi definido o tempo máximo $tp(s)$, como o tempo total de execução estimado da regressão para s e c_{udd} como o custo de *upload*, *download* e de armazenamento dos dados de uma máquina virtual p.

Além disso, foi definida uma função de custo $F : S \to R$ (veja a Equação 5.7), que calcula a qualidade da solução. A função F tenta minimizar os custos financeiros e o tempo total de execução, enquanto relaxa as desigualdades, penalizando a inviabilidade em relação ao tempo máximo e o custo financeiro estipulados pelo usuário. Os parâmetros λ_1 e λ_2, são coeficientes de penalidade associados à violação dos requisitos de tempo e de custo, respectivamente.

$$\begin{aligned} F(s) = \Bigg(\alpha_1 \Bigg(\sum_{(p,i)\in s} c_p + \sum_{(p,i)\in s} c_{udd} \Bigg) + \alpha_2 tp(s) \Bigg) \\ + \lambda_1 (max\{0, tp(s) - T_m\}) \\ + \lambda_2 \left(max \left\{ 0, \sum_{(p,i)\in s} c_p - C_m \right\} \right) \end{aligned} \tag{5.7}$$

Assim, o Algoritmo 2 possui como entrada o conjunto de máquinas virtuais P, o custo total da execução C_m, o tempo máximo de execução T_m, a lista de softwares que compõem o *workflow*, os parâmetros que definem o custo, o tempo e as penalidades de cada um, sendo eles $\alpha_1, \alpha_2, \lambda_1$ e λ_2, respectivamente. O parâmetro $maxIter$ indica o número máximo de iterações sem melhoria, na melhor solução encontrada. A primeira tarefa de cada iteração do Algoritmo 2 é construir uma solução a partir de um conjunto vazio, de forma randomizada e gulosa, por meio do Algoritmo 3.

Algoritmo 2: GRASP.

Entrada: $P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Saída: s^*;
1 $s^* := \emptyset$;
2 $F(s^*) := \infty$;
3 $i := 0$;
4 **enquanto** $i \leqslant maxIter$ **faça**
5 $\quad s := construcao(P, C_m, T_m, \alpha_1, \alpha_2, \lambda_1, \lambda_2)$;
6 $\quad s := buscaLocal(s, P, C_m, T_m, \alpha_1, \alpha_2, \lambda_1, \lambda_2)$;
7 $\quad$ **se** $(F(s) < F(s^*))$ **então**
8 $\quad\quad s^* := s$;
9 $\quad\quad i := 0$;
10 $\quad$ **fim**
11 $\quad i := i + 1$;
12 **fim**
13 *retorna* s^*;

Na fase de construção do GRASP (Algoritmo 3) é definida uma lista ordenada L_p (veja linha 2 do Algoritmo 3), com todas as máquinas virtuais $p \in P$, de tal forma que os elementos aparecem em ordem descendente do custo financeiro e do poder de processamento, calculado a partir da fórmula $(\alpha_1 c_p + \alpha_2 n_p)$.

A cada iteração das linhas (3)-(6) do Algoritmo 3, é feita a escolha de uma máquina virtual $p*$, entre as primeiras β máquinas virtuais a serem adicionadas. O parâmetro β define o grau de aleatoriedade que a fase de construção terá. O processo termina somente quando houver uma solução. E nessa solução, cada máquina virtual escolhida tem o mínimo de memória necessária para a execução do software, sendo esse valor obtido por meio do monitoramento das execuções.

Algoritmo 3: Construção.

Entrada: $s, P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Saída: s;
1 $s := \emptyset$;
2 $L_p := ordenar(P)$;
3 **enquanto** $(qntItens(s) \leqslant qntItens(L_s))$ *e* $(m_p^* \leqslant M)$ **faça**
4 $\quad$ escolha máquina virtual p^* (index i) randomicamente entre os primeiros β elementos de L_p ;
5 $\quad s := s \cup \{p^*, i^*\}$;
6 **fim**
7 *retorna* s;

Todavia, não há garantia de que o método de construção retorne uma solução viável, ou localmente ótima em relação à sua vizinhança. Como consequência, a solução pode ser melhorada através do procedimento de busca local, mostrado no Algoritmo 4.

Assim, no Algoritmo 4 a estrutura de vizinhança $Nr(s)$ é definida como a família de todas as soluções obtidas pelas trocas de *rtuplas* em s, com outras *rtuplas* não contidas em s. As trocas são feitas extensivamente com a estratégia da primeira melhoria, método no qual é adquirido o contexto de uma nova solução, assim que forem encontradas. O método de busca local inicia com a solução fornecida pela fase de construção, e iterativamente, substitui a solução atual, quando s melhora, ou seja, quando a solução da vizinhança encontrada possui menor custo em relação à solução atual, avaliada pela função de custo (veja a Equação 5.7).

Neste momento, a busca é reiniciada até que não haja mais melhorias na vizinhança da nova solução. Para este trabalho, especificamente, foi empregada uma vizinhança com $r \leqslant 2$, uma vez que valores superiores a dois condicionam computação intensiva para realizar uma busca exaustiva. De modo que a vizinhança se estenderia para um subconjunto cada vez maior. A estrutura das vizinhanças pode ser definida como $N1(s) = \{(s_0, ..., s_{i+1}, s_i, ..., s_n) : i = 0; ..., n-1\}$ e $N2(s) = \{(s_0, ..., s_j, s_i, ..., s_n) : i = 0; ..., n-1; j = i+1, ..., n\}$. A primeira indica uma permutação de $i+1$ itens que formam o subconjunto de soluções a partir da solução i. A segunda, indica uma permutação de $i+1$ se estendendo para cada item a partir de $j = i+1$, onde é possível obter um subconjunto ainda maior de soluções a partir de uma solução i.

Algoritmo 4: Busca Local.

Entrada: $P, C_m, T_m, L_s, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
Saída: s;
enquanto *s melhora* **faça**
  para todo $s^* \in (N1(s) \cup N2(s))$ **faça**
    se $(F(s^*) < F(s))$ **então**
      $s := s^*$;
    fim
  fim
fim
retorna s;

Vale ressaltar que a função implementada neste trabalho, não leva em consideração o tempo de alocação das máquinas virtuais, uma vez que a alocação é feita para o programa, e não para um conjunto de ativações como proposto por Coutinho *et al.* [20]. Desse modo, a solução leva em consideração o software como uma caixa preta sendo possível estimar o *workflow* através do seu conjunto de programas antes mesmo que ele seja executado. Além de considerar também todos os custos financeiros de execução do *workflow*, inclusive de transferência (*upload/download*).

A próxima seção apresenta o uso do modelo preditivo juntamente com a metaheuristica GRASP. O objetivo é indicar eficientemente, detre as várias opções de uma nuvem federada, um conjunto de recursos para os *workflows*, além do custo e do tempo demandado.

5.9 Resultados da Metaheurística GRASP

Nesta seção são apresentados os resultados obtidos a partir da metaheurística GRASP implementada neste trabalho por meio das execuções em ambiente de nuvens federadas. Para isso, foi utilizada a plataforma BioNimBuZ como estudo de caso. O principal objetivo destes testes é analisar o conjunto de recursos, juntamente com o custo financeiro e o tempo, sendo que a escolha desses dois últimos é resultado de uma interação com o usuário. Para esses testes, foram utilizados dois provedores de nuvem, o *Google Cloud Platform* [42] e o *Amazon Web Services* [100], uma vez que ambos disponibilizam recursos de forma gratuita.

Cada um dos provedores citados possui uma gama de opções de máquinas virtuais, desde máquinas preemptivas até mesmo máquinas robustas com 128 núcleos. Entretanto, para este trabalho, as máquinas preemptivas foram removidas do domínio, a fim de eliminar possíveis interferências no monitoramento, caso elas fossem usadas.

Os resultados experimentais são a verificação, em termos de qualidade da solução, e do tempo de execução predito pelo modelo de regressão. Os testes foram feitos com os programas que compõem os dois *workflows* de bioinformática adotados neste trabalho. Assim, para esses *workflows* devem ser encontrados um conjunto de recursos com o tempo total de execução e custo financeiro dentro do limite definido pelo usuário. Os parâmetros utilizados no algoritmo foram $\lambda_1 = 500$, $\lambda_2 = 500$, $\beta = soma$ (número máximo de máquinas virtuais que o usuário pode alocar), $maxIter = 1000$, e α_1 e α_2 representam os casos em que as variáveis alteram seus pesos na função objetivo. Os parâmetros λ_1, λ_2 e $maxIter$ foram definidos empiricamente, após vários experimentos realizados. Além disso, também foram definidas as variáveis de usuário como o custo máximo a ser gasto e o tempo máximo desejado. Para os testes foram considerados 4 cenários:

- O primeiro cenário caracteriza uma execução de baixo orçamento;
- O segundo cenário é definido como uma execução de alto desempenho e alto orçamento;
- O terceiro cenário considera que existe um custo financeiro máximo definido pelo usuário;
- O último cenário limita o tempo máximo de execução, também configurado pelo usuário.

No primeiro cenário, definiu-se os parâmetros, $T_m = C_m = \infty$, uma vez que não seriam aplicados limites de custo e tempo; $\alpha_1 = 1, \alpha_2 = 0$. Assim, a performance da execução será totalmente desconsiderada. Com isso, ao executar o algoritmo foram selecionadas as instâncias de acordo com os parâmetros de usuário.

A Tabela 5.7 detalha as instâncias usadas. Assim, na primeira coluna é informado o nome da instância escolhida. Na segunda coluna, é apresentado o número de núcleos, a terceira coluna informa a memória RAM, e a quarta coluna especifica o Sistema Operacional.

Além da tabela de instâncias, é apresentada na Tabela 5.8 a combinação de testes realizados a partir do sPCR. Assim, a tabela indica qual a instância escolhida, representada pela Tabela 5.7. A segunda coluna apresenta qual o *workflow* envolvido na execução das

Tabela 5.7: Tabela de Instâncias Utilizadas pelo sPCR.

	Instância	Núcleos.CPU	Memória
1	m3.large	2	7.5
2	t2.large	2	8
3	t2.large	2	8
4	t2.large	2	8
5	c4.4xlarge	16	30
6	c4.4xlarge	16	30
7	N1-ST-16	16	60
8	N1-HIG-16	16	14.4
9	c4.2xlarge	8	15
10	m3.large	2	7.5
11	c4.xlarge	4	7.5
12	t2.large	2	8
13	c4.4xlarge	16	30
14	c4.4xlarge	16	30
15	c4.4xlarge	16	30
16	c4.4xlarge	16	30

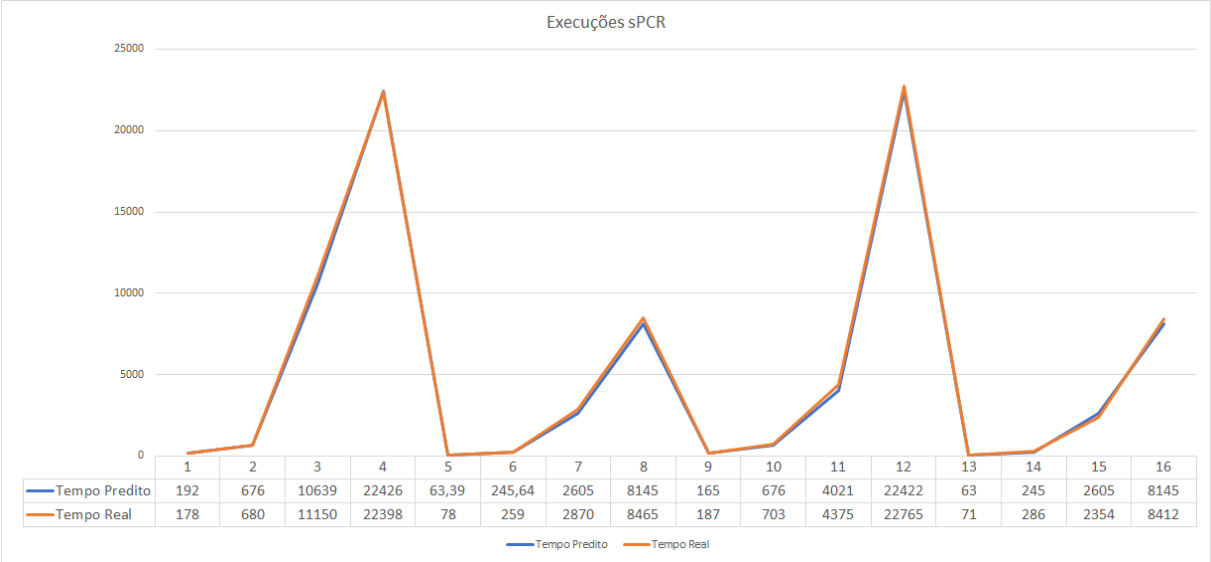
estimativas. A terceira coluna mostra o programa envolvido. A quarta coluna representa o tempo predito pelo modelo em segundos, dado um conjunto finito de recursos escolhidos. A quinta coluna informa o tempo de execução real em segundos, no recurso selecionado pela metaheuristica, sendo que o custo total, apresentado na sexta coluna, representa o valor financeiro, o qual é calculado a partir do tempo previsto. E por último, as colunas sete, oito, nove e dez, apresentam os quatro parâmetros de usuário, no qual o usuário pode optar por custo x benefício. As colunas sete e oito representam pesos da função objetivo custo e tempo, respectivamente. A coluna nove representa o tempo máximo de execução definido pelo usuário, e por último a coluna dez representa o custo máximo que o usuário quer pagar pela execução.

Além disso, é importante ressaltar que este trabalho foca na obtenção da lista dos recursos de acordo com o tempo e o custo financeiro especificado pelo usuário. Assim sendo, é possível notar na Tabela 5.8 que para o primeiro cenário (Instâncias 1 a 4), o erro médio foi de 139.25 segundos (3.14%). Para o segundo cenário (Instâncias de 5 a 8) ocorreu um erro médio de 153.24 segundos (9.22%). No terceiro cenário (Instâncias 9 a 12) o erro médio foi de 186.5 segundos (6.30%). Por último, o quarto cenário (Instâncias 13 a 16) obteve um erro médio de 141.75 segundos (9.60%). Com isso, é possível observar que o maior erro percentual apurado foi do quarto cenário, significando que ao limitar o tempo a $10k$, o serviço busca ajustar o tempo máximo, dentro de um conjunto de máquinas que não extrapole o tempo de execução indicado, sendo que o *workflow* possui uma execução rápida e de alto custo.

A Figura 5.18 apresenta o comparativo entre o tempo de execução real e o tempo predito das execuções dos quatro cenários apresentados na Tabela 5.8. O eixo x representa as execuções de cada instâncias/programas especificados nos *workflows*. O eixo y apresenta o tempo em segundos de cada programa estimado a partir do sPCR. Além disso, a legenda apresenta, respectivamente, os tempos de execução preditos e reais.

Tabela 5.8: Tabela com Resultados das Execuções do sPCR.

Instâncias	*Workflow*	Programa	Tempo de Exec. Predito	Tempo de Exec. Real	Custo Total	α_1	α_2	T_m	C_m
1	1	Bowtie	178	192	0.53	1	0	∞	∞
2	1	TopHat	676	680	1.47	1	0	∞	∞
3	1	Trinity	10639	11150	14.35	1	0	∞	∞
4	1	Blast	22426	22398	42.24	1	0	∞	∞
5	2	Bowtie	63.39	78	1.04	0	1	∞	∞
6	2	TopHat	245.64	259	3.55	0	1	∞	∞
7	2	Trinity	2605	2870	43.41	0	1	∞	∞
8	2	Blast	8145	8465	82.53	0	1	∞	∞
9	1	Bowtie	165	187	1.15	0.9	0.1	∞	50
10	1	TopHat	676	703	1.62	0.9	0.1	∞	50
11	1	Trinity	4021	4375	14.13	0.9	0.1	∞	50
12	1	Blast	22422	22765	39.25	0.9	0.1	∞	50
13	2	Bowtie	63	71	0.92	0.1	0.9	10k	∞
14	2	TopHat	245	286	3.55	0.1	0.9	10k	∞
15	2	Trinity	2605	2354	36.51	0.1	0.9	10k	∞
16	2	Blast	8145	8412	114	0.1	0.9	10k	∞



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Tempo Predito	192	676	10639	22426	63,39	245,64	2605	8145	165	676	4021	22422	63	245	2605	8145
Tempo Real	178	680	11150	22398	78	259	2870	8465	187	703	4375	22765	71	286	2354	8412

Figura 5.18: Comparação dos Resultados das Execuções do sPCR.

Além disso, este trabalho também foi implementada a técnica de força bruta, que consiste em enumerar todos os possíveis candidatos da solução iterativamente utilizando a mesma função objetivo da metaheurística GRASP. O algoritmo é capaz de obter a melhor solução para os quatro cenários apresentados. Desta forma, é possível comparar a eficiência da metaheurística proposta.

Assim, são apresentadas na Tabela 5.9 as comparações entre a técnica de força bruta e a metaheurística proposta neste trabalho. A primeira coluna da tabela indica o método em questão, a segunda coluna o custo financeiro para cada cenário, a terceira coluna indica o tempo de execução do *workflow*, as colunas quatro a sete indicam as restrições do usuário e a última coluna indica o tempo de duração para encontrar a solução. Nesse sentido, é possível observar que para todos os cenários a metaheurística GRASP é na média 99.9% mais rápida do que a técnica de força bruta. Além de obter boas soluções, visto que o maior erro percentual em relação ao custo financeiro é do cenário 2 com 1.47% entre GRASP e força bruta. Já para a estimativa de tempo, o maior erro percentual está relacionado com os cenários 1 e 3, com 1.22% de erro entre GRASP e força bruta.

As Figuras 5.19 e 5.20 apresentam as estimativas de tempo e de custo financeiros para as execuções entre GRASP e força bruta, respectivamente. A Figura 5.19 , apresenta em seu eixo x o tempo em segundos, no eixo y os cenários envolvidos. Já na Figura 5.20, o eixo x representa o custo financeiro para cada um dos cenários representados no eixo y. Diante do exposto, é possível observar que o sPCR foi capaz de estimar o conjunto de máquinas a partir de um ambiente de nuvem federada, respeitando as restrições impostas pelo usuário com eficiência e rapidez. Dessa forma, ele é adequado para ser usado na predição de recursos e nas estimativas de tempo e custo em ambientes de nuvem federada, uma vez que foi possível, a partir da lista de recursos de ambos os provedores utilizados para estes testes, estimar os recursos adequados para os cenários apresentados.

Tabela 5.9: sPCR *versus* Força Bruta.

Método	Custo Financeiro	Tempo	α_1	α_2	λ_1	λ_2	Tempo de Execução
GRASP	53.85	30071	1	0	∞	∞	0.117
Força Bruta	53.43	30444	1	0	∞	∞	9388
GRASP	157.45	11060	0	1	∞	∞	0.198
Força Bruta	155.13	11060	0	1	∞	∞	9792
GRASP	53.81	30071	0.9	0.1	50	∞	0.174
Força Bruta	53.43	30444	0.9	0.1	50	∞	8689
GRASP	155.93	11060	0.1	0.9	∞	10k	0.229
Força Bruta	155.13	11060	0.1	0.9	∞	10k	8702

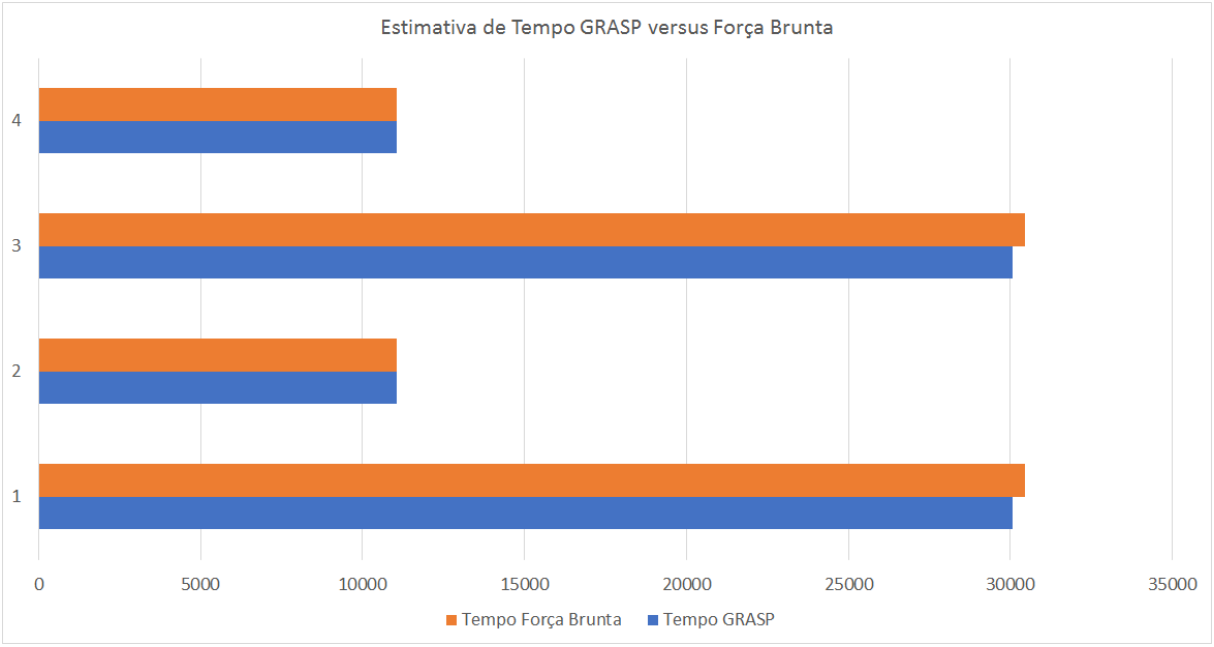


Figura 5.19: Comparação das Estimativas de Tempo: GRASP *versus* Força Bruta.

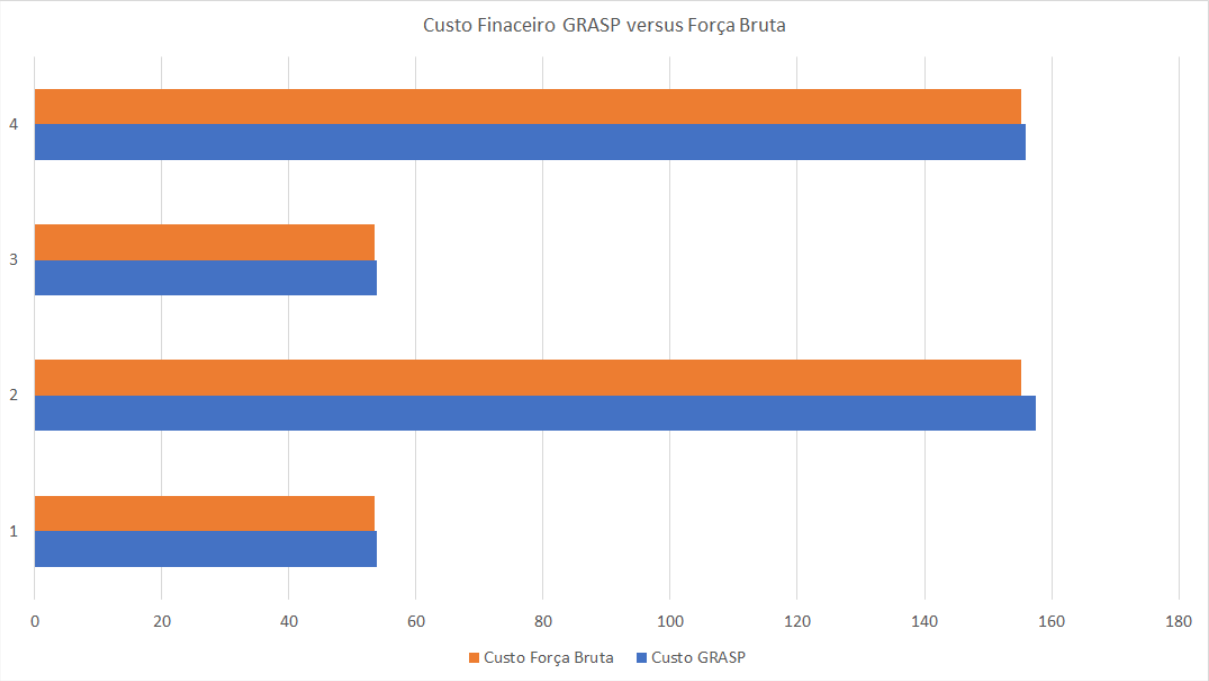


Figura 5.20: Comparação das Estimativas de Custo: GRASP *versus* Força Bruta.

Neste capítulo, foram apresentados os *workflows* utilizados neste trabalho, além de mostrar os resultados obtidos com o modelo preditivo de regressão múltipla, e também os testes com o GRASP para as estimativas de recursos em um ambiente real de federação. Assim sendo, o próximo capítulo apresenta as conclusões e os trabalhos futuros deste trabalho.

Capítulo 6

Conclusão e Trabalhos Futuros

O uso da computação em nuvem federada, para suprir as demandas por recursos dos *workflows* de bioinformática traz consigo uma gama de recursos, e a escolha desses não é trivial para os usuários que trabalham com esses *workflows*. Dessa forma, este trabalho apresentou um serviço de predição de recursos que possibilitou ao usuário escolher uma execução de baixo custo financeiro ou de alto desempenho. Isso foi possível por meio da utilização do método estatístico de Regressão Linear e também por meio da utilização da metaheurística GRASP para a escolha eficiente de recursos em um tempo hábil.

Nesse contexto, o modelo de regressão linear múltipla implementado foi capaz de prever o tempo e o custo de um determinado programa, a partir dos dados de monitoramento desse programa disponíveis na base de dados histórica. Além disso, neste trabalho foi considerada uma base de dados histórica sem dados reais do programa *Bowtie*. Para isso, foi implementado um fluxo no qual é criada uma base de dados simulada de forma a disponibilizar as informações na base de dados histórica.

A metaheurística GRASP foi usada com o intuito de disponibilizar uma solução viável de recursos para execução dos *workflows*. Nesse caso, o método busca, iterativamente e de forma aleatorizada e gulosa, uma solução que seja compatível com as decisões dos usuários. Esse método mostrou ser eficiente, a partir da análise dos resultados, pois as soluções não excederam um tempo de resposta superior a 45 segundos para obtenção da lista de recursos.

Diante do exposto, co o Serviço de Predição proposto neste trabalho, foi possível estimar um conjunto de máquinas para os *workflows* utilizados como estudo de caso, sendo que as estimativas respeitaram os cenários definidos pelo usuário. Dessa forma, foi possível comprovar que a premissa básica, que era evitar o desperdício de recursos e ainda facilitar, de forma transparente, as escolhas de recursos para os usuários, foi atingida com este trabalho.

Assim, os trabalhos futuros incluirão análises de outros modelos preditivos, incluindo aprendizado de máquina, uma vez que no decorrer do tempo, pode-se obter uma base de dados histórica suficiente para a utilização desses modelos preditivos.

Outros trabalhos futuros a serem desenvolvidos são: a implementação do modelo de predição para ambientes de programação paralela, considerando o custo e o tempo de execução; e implementar um banco de dados distribuído a fim de aprimorar o acesso das informações da base de dados histórica.

Para finalizar, é importante registrar que este trabalho teve como resultado imediato a publicação do artigo M. Rosa *et al.*, BioNimbuZ: A federated cloud platform for bioinformatics applications, 2016 *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Shenzhen, 2016, pp. 548-555. doi: 10.1109/BIBM.2016.7822580 e o convite para uma versão estendida no periódico *International Journal of Data Mining and Bioinformatics (IJDMB)*.

Referências

[1] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology 215*, 3 (1990), 403–410. 9

[2] Araújo, A. P. F. *Paralelização Autonômica de Metaheurísticas em Ambientes de Grid*. PhD thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio, Brasil, 2008. 39, 41

[3] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. Above the clouds: A Berkeley view of cloud computing. Tech. rep., Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 2009. 11

[4] Armstrong, J. S., and Overton, T. S. Estimating nonresponse bias in mail surveys. *Journal of marketing research* (1977), 396–402. 58

[5] 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/10483/13199`, 2015. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 2, 20, 21, 31

[6] 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. 2, 20, 21, 31

[7] Bendel, R. B., and Afifi, A. A. Comparison of stopping rules in forward "stepwise" regression. *Journal of the American Statistical Association 72*, 357 (1977), 46–53. 53

[8] Bendel, R. B., and Afifi, A. A. Comparison of stopping rules in forward "stepwise" regression. *Journal of the American Statistical Association 72*, 357 (1977), 46–53. 53

[9] Bittman, T. The evolution of the cloud computing market. *Gartner Blog Network, http://blogs. gartner. com/thomas bittman/2008/11/03/theevolution-of-the-cloud-computing-market* (2008). Acessado online em 20 de abril de 2015. 16

[10] Boussaïd, I., Lepagnot, J., and Siarry, P. A survey on optimization metaheuristics. *Information Sciences 237* (2013), 82 – 117. Prediction, Control and Diagnosis using Advanced Neural Computations. 39

[11] Buyya, R., Abramson, D., and Giddy, J. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on* (2000), vol. 1, IEEE, pp. 283–289. 36

[12] Buyya, R., Ranjan, R., and Calheiros, R. N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I* (Berlin, Heidelberg, 2010), ICA3PP'10, Springer-Verlag, pp. 13–31. x, 1, 16, 17, 19, 20

[13] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst. 25*, 6 (jun. 2009), 599–616. 1, 11, 12

[14] Camacho, C., Madden, T., Ma, N., et al. Blast command line applications user manual. 2013. *Reference Source*. 10, 50, 51

[15] 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* (jul. 2010), pp. 337 –345. x, 16, 17, 18

[16] Chaisiri, S., Lee, B. S., and Niyato, D. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing 5*, 2 (April 2012), 164–177. 2, 3

[17] Chen, Z., Zhu, Y., Di, Y., and Feng, S. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. *Intell. Neuroscience 2015* (Jan. 2015), 17:17–17:17. 35

[18] Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Research 38*, 6 (2010), 1767–1771. 6

[19] Coutinho, R. d. C., Drummond, L. M., and Frota, Y. Optimization of a cloud resource management problem from a consumer perspective. In *Euro-Par 2013: Parallel Processing Workshops* (2013), Springer, pp. 218–227. 36, 45, 65

[20] Coutinho, R. d. C., Drummond, L. M., Frota, Y., and de Oliveira, D. Optimizing virtual machine allocation for parallel scientific workflows in federated clouds. *Future Generation Computer Systems 46* (2015), 51–68. 35, 36, 42, 43, 45, 46, 68

[21] Cristianini, N., and Shawe-Taylor, J. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. 37

[22] Crockford, D. Json. `http://json.org/`, 2012. 23

[23] de Oliveira, D., Ogasawara, E., Baião, F., and Mattoso, M. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (2010), IEEE, pp. 378–385. 40

[24] de Oliveira, D., Viana, V., Ogasawara, E., Ocana, K., and Mattoso, M. Dimensioning the virtual cluster for parallel scientific workflows in clouds. In *Proceedings of the 4th ACM Workshop on Scientific Cloud Computing* (New York, NY, USA, 2013), Science Cloud '13, ACM, pp. 5–12. 36, 40

[25] 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

[26] Emeakaroha, V. C., Łabaj, P. P., Maurer, M., Brandic, I., and Kreil, D. P. Optimizing bioinformatics workflows for data analysis using cloud management techniques. In *Proceedings of the 6th workshop on Workflows in support of large-scale science* (2011), ACM, pp. 37–46. 2

[27] 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. 32

[28] Feo, T. A., and Resende, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization 6*, 2 (1995), 109–133. 40, 41, 46

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

[30] 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`, 1999. Acessado online em 05 de junho de 2015. 23

[31] Fonseca, N. A., Rung, J., Brazma, A., and Marioni, J. C. Tools for mapping high-throughput sequencing data. *Bioinformatics* (2012), bts605. 9

[32] 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* (nov. 2008), pp. 1 –10. x, 11, 13, 14

[33] Foundation, A. S. Apache avro). `http://avro.apache.org/`, 2012. 21, 23

[34] Foundation, T. A. S. Apache zookeeper. `http://zookeeper.apache.org/`, 2010. 21, 23

[35] Fundation, T. A. S. Apache avro documentation. http://avro.apache.org/docs/1.7.7/, 2015. Acessado online em 05 de junho de 2015. 23

[36] Fundation, T. A. S. Apache fundation. http://apache.org/, 2015. Acessado online em 30 de março de 2015. 23

[37] Gallon, R. F. Política de armazenamento de dados em nuvens federadas para dados biológicos. Master's thesis, School, 2014. 31

[38] Goldberg, D. E., et al. *Genetic algorithms in search optimization and machine learning*, vol. 412. Addison-wesley Reading Menlo Park, 1989. 39, 40

[39] Google. Cloud storage. https://cloud.google.com/storage/. Acessado online em 10 de janeiro de 2017. 32

[40] Google. Google docs. https://docs.google.com/?hl=pt-BR, 2012. Acessado online em 27 de abril de 2015. 15

[41] Google. Google app engine. https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=pt-br, 2013. 14

[42] Google. Google cloud plataform. https://cloud.google.com/, 2013. 16, 69

[43] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., Chen, Z., Mauceli, E., Hacohen, N., Gnirke, A., Rhind, N., di Palma, F., Birren, B. W., Nusbaum, C., Lindblad-Toh, K., Friedman, N., and Regev, A. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nat Biotech 29*, 7 (Jul 2011), 644–652. 50

[44] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., et al. Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data. *Nature biotechnology 29*, 7 (2011), 644. 9, 51

[45] 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. 5, 9

[46] Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., Couger, M. B., Eccles, D., Li, B., Lieber, M., MacManes, M. D., Ott, M., Orvis, J., Pochet, N., Strozzi, F., Weeks, N., Westerman, R., William, T., Dewey, C. N., Henschel, R., LeDuc, R. D., Friedman, N., and Regev, A. De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. *Nat. Protocols 8*, 8 (Aug 2013), 1494–1512. Protocol. 9, 50

[47] Hastie, T., Tibshirani, R., and Friedman, J. *Unsupervised Learning*. Springer New York, New York, NY, 2009, pp. 485–585. 37

[48] Higgins, D. G., and Sharp, P. M. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene 73*, 1 (1988), 237–244. 9

[49] Hollingsworth, D., et al. The workflow reference model: 10 years on. In *Fujitsu Services, UK; Technical Committee Chair of WfMC* (2004), Citeseer. 5

[50] Islam, S., Keung, J., Lee, K., and Liu, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems 28*, 1 (2012), 155 – 162. 35, 36, 38, 41, 42, 43

[51] JCraft. Pure implementation of sftp for java. `http://www.jcraft.com/jsch/examples/Sftp.java.html`, 2012. 21

[52] Junqueira, F., and Reed, B. *ZooKeeper: distributed process coordination.* "O'Reilly Media, Inc.", 2013. 24

[53] Kamburugamuve, S., Fox, G., Leake, D., and Qiu, J. Survey of apache big data stack. *Indiana University, Tech. Rep.* (2013). 24

[54] Khatua, S., Manna, M. M., and Mukherjee, N. Prediction-based instant resource provisioning for cloud applications. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing* (Dec 2014), pp. 597–602. 36

[55] Kianpisheh, S., and Charkari, N. M. A grid workflow quality-of-service estimation based on resource availability prediction. *The Journal of Supercomputing 67*, 2 (2014), 496–527. 35, 42, 43

[56] Kiran, M., Hashim, A.-H. A., Kuan, L. M., and Jiun, Y. Y. Execution time prediction of imperative paradigm tasks for grid scheduling optimization. *Int J Comput Sci Netw Secur 9*, 2 (2009), 155–163. 42, 43

[57] Kwei-Jay, L., and Gannon, J. Atomic remote procedure call. *Software Engineering, IEEE Transactions on SE-11*, 10 (1985), 1126–1135. 21

[58] Lab, H. Fastx-toolkit. `http://hannonlab.cshl.edu/fastx_toolkit/index.htm`, 2015. Último acesso em 12/11/1015. 6

[59] Lama, P., and Zhou, X. Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In *Proceedings of the 9th international conference on Autonomic computing* (2012), ACM, pp. 63–72. 36, 37

[60] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. Ultrafast and memory-eficient alignment of short dna sequences to the human genome, 2009. 50, 51

[61] Langmead, B., Trapnell, C., Pop, M., Salzberg, S. L., et al. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biol 10*, 3 (2009), R25. 8, 9, 10

[62] Latgé, J.-P. Aspergillus fumigatus and aspergillosis. *Clin Microbiol Rev 12*, 2 (Apr 1999), 310–350. 0016[PII]. 50

[63] 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. 12

[64] Levene, P., and London, E. The structure of thymonucleic acid. *Journal of Biological Chemistry 83*, 3 (1929), 793–802. 6

[65] 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). 2, 20, 21

[66] Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., and Fotouhi, F. Service-oriented architecture for view: a visual scientific workflow management system. In *Services Computing, 2008. SCC'08. IEEE International Conference on* (2008), vol. 1, IEEE, pp. 335–342. 42

[67] Lin, F.-J. Solving multicollinearity in the process of fitting regression model using the nested estimate procedure. *Quality & Quantity 42*, 3 (2008), 417–426. 57

[68] LLC, A. W. S. Amazon elastic compute cloud (EC2). `http://aws.amazon.com/pt/ec2/`, 2012. 14, 16

[69] LLC, A. W. S. Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3/`, 2015. Acessado online em 29 de maio de 2015. 14, 32

[70] Matsubara, E. T., Monard, M. C., and Batista, G. E. Multi-view semi-supervised learning: An approach to obtain different views from text datasets. In *LAPTEC* (2005), pp. 97–104. 36, 37

[71] Mattoso, M., Dias, J., Costa, F., de Oliveira, D., and Ogasawara, E. Experiences in using provenance to optimize the parallel execution of scientific workflows steered by users. In *Workshop of Provenance Analytics* (2014). 5

[72] Mell, P., and Grance, T. The NIST definition of cloud computing. *National Institute of Standards and Technology 53*, 6 (2009), 50. 12

[73] Mendenhall, W., Sincich, T., and Boudreau, N. S. *A second course in statistics: regression analysis*, vol. 5. Prentice Hall Upper Saddle River^ eNew Jersey New Jersey, 1996. 38, 39

[74] Microsoft. Windows azure. `http://www.windowsazure.com/pt-br/`, 2013. 16

[75] Montgomery, D. C., Peck, E. A., and Vining, G. G. *Introduction to linear regression analysis*. John Wiley & Sons, 2015. 38, 39

[76] Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods 5*, 7 (2008), 621–628. 8

[77] Mossucca, L., Terzo, O., Goga, K., Acquaviva, A., Abate, F., and Provenzano, R. Ngs workflow optimization using a hybrid cloud infrastructure. *International Journal on Advances in Networks and Services Volume 5, Number 3 & 4, 2012* (2012). 9

[78] 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. 31

[79] 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. 1, 42

[80] 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. 31

[81] Pandey, S., Wu, L., Guru, S. M., and Buyya, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on* (2010), IEEE, pp. 400–407. 36

[82] Paula, R. d. Proveniência de dados em workflows de bioinformática. Master's thesis, Departamento de Computação, Universidade de Brasília Brasília, DF, Brasil, 2013. x, 5, 6, 7, 8, 9

[83] Pearson, W. R., and Lipman, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences 85*, 8 (1988), 2444–2448. 6

[84] Pereira, G. C., de Oliveira, M. M., and Ebecken, N. F. Genetic optimization of artificial neural networks to forecast virioplankton abundance from cytometric data. x, 40

[85] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0. 53

[86] Raicu, I., Foster, I. T., and Zhao, Y. Many-task computing for grids and supercomputers. In *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on* (2008), IEEE, pp. 1–11. 5

[87] 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, 2, 20, 21, 25, 26, 27, 28, 29, 34

[88] RedHat, J. Resteasy. http://resteasy.jboss.org. Acessado online em 07 de janeiro de 2017. 29

[89] Resende, M. G., and de Sousa, J. P. *Metaheuristics: computer decision-making*, vol. 86. Springer Science & Business Media, 2013. 39

[90] Ribeiro, C. C., Martins, S. L., and Rosseti, I. Metaheuristics for optimization problems in computer communications. *Computer Communications 30*, 4 (2007), 656–669. 39

[91] Rimal, B. P., Eunmi, C., and Lumb, I. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on* (aug. 2009), pp. 44–51. 14, 15

[92] Rodrigues, G., Bauman, G., Lagerwaard, F., et al. Classification of brain metastases prognostic groups utilizing artificial neural network approaches. *Cureus 5*, 5 (2013). x, 38

[93] Rosa, M., Moura, B. R., Vergara, G., Santos, L., Ribeiro, E., Holanda, M., Walter, M. E., and Araújo, A. Bionimbuz: A federated cloud platform for bioinformatics applications. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (Dec 2016), pp. 548–555. 2, 20

[94] Saldanha, H., Ribeiro, E., Borges, C., Araújo, A., Gallon, R., Holanda, M., Walter, M. E., Togawa, R., and Setubal, J. C. *BioInformatics*. In Tech, 2012. 2, 19, 20

[95] 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, 16, 21

[96] SanMiguel, P. J., Ramakrishna, W., Bennetzen, J. L., Busso, C. S., and Dubcovsky, J. Transposable elements, genes and recombination in a 215-kb contig from wheat chromosome 5am. *Functional & Integrative Genomics 2*, 1 (2002), 70–80. 7, 8

[97] 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. 2, 20, 21, 32

[98] Sardenberg, R. S. Integridade e confidencialidade dos arquivos na plataforma de nuvem federada BioNimbuZ. http://bdm.unb.br/handle/10483/13502, 2016. Monografia de graduação, Departamento de Ciência de Computação, Universidade de Brasília. 34

[99] Schmieder, R., and Edwards, R. Quality control and preprocessing of metagenomic datasets. *Bioinformatics 27*, 6 (2011), 863–864. 9

[100] Services, A. W. Amazon web services. http://aws.amazon.com/pt/, 2013. 69

[101] 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. 20

[102] Team, R. C. R: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria, 2012, 2014. 9

[103] Trapnell, C., Pachter, L., and Salzberg, S. L. Tophat: discovering splice junctions with rna-seq. *Bioinformatics 25*, 9 (2009), 1105–1111. 8, 9, 10, 50, 51

[104] Van Steen, M., and Tanenbaum, A. *Distributed Systems: Principles and Paradigms*. Prentice Hall; 2 edition, 2006. 21

[105] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev. 39*, 1 (Dec. 2008), 50–55. x, 1, 12, 13

[106] Vergara, G. F. Arquitetura de um controlador de elasticidade para nuvens federadas. um estudo de caso na plataforma BioNimbuZ. Master's thesis, Universidade de Brasília - UnB, 2017. x, 29, 30

[107] Wang, S.-C. *Artificial Neural Network*. Springer US, Boston, MA, 2003, pp. 81–100. 37

[108] Weisberg, S. *Applied linear regression*, vol. 528. John Wiley & Sons, 2005. 56

[109] Widodo, A., and Yang, B.-S. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing 21*, 6 (2007), 2560 – 2574. x, 37

[110] Wieers, D. Dstat: Versatile resource statistics tool. *online] http://dag. wiee. rs/home-made/dstat/(accessed December 2016)* (2016). 49

[111] Yin, J., Lu, X., Chen, H., Zhao, X., and Xiong, N. N. System resource utilization analysis and prediction for cloud based applications under bursty workloads. *Information Sciences 279* (2014), 338–357. 42

[112] 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. 12

[113] Zookeeper, A. Curator. `http://curator.apache.org/`. Acessado online em 10 de janeiro de 2017. 24