UNIVERSIDADE FEDERAL DO TOCANTINS

MAYCON ANTONIO JUNQUEIRA COSTA

Abordagem para Classificação de Ofertas de Provedores de Computação em Nuvem para Implantação da Infraestrutura de Workflows Científicos

PALMAS - TOCANTINS

2014

UNIVERSIDADE FEDERAL DO TOCANTINS

MAYCON ANTONIO JUNQUEIRA COSTA

Abordagem para Classificação de Ofertas de Provedores de Computação em Nuvem para Implantação da Infraestrutura de Workflows Científicos

Trabalho de Conclusão de Curso apresentado na Universidade Federal do Tocantins (UFT), como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Ary Henrique Morais de Oliveira. M.Sc.

PALMAS - TOCANTINS

2014

Abordagem para Classificação de Ofertas de Provedores de Computação em Nuvem para Implantação da Infraestrutura de Workflows Científicos

Maycon Antonio Junqueira Costa

Trabalho de Conclusão de Curso apresentado na Universidade Federal do Tocantins (UFT), como parte das exigências do curso de Ciência da Computacão para obtenção do título de Bacharel em Ciência da Computação.

Aprovada por:

Prof. Ary Henrique Morais de Oliveira. M.Sc.
(Presidente)

Prof. Warley Gramacho, D.Sc.

Prof. Thiago Magalhães de Brito Rodrigues. M.Sc.

Palmas, 29 de Setembro de 2014.

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca da Universidade Federal do Tocantins
Campus Universitário de Palmas

C837a Costa, Maycon Antonio Junqueira
Abordagem para Classificação de Ofertas de Provedores de Computação em Nuvem para Implantação da Infraestrutura de Workflows Científico / Maycon Antonio Junqueira Costa. – Palmas, 2014.
67f.

Monografia (TCC) – Universidade Federal do Tocantins, Curso de Ciência da Computação, 2014.
Orientador: Prof. Msc. Ary Henrique Morais de Oliveira

1. *Cloud Computer*. 2. *Open Provenance Model*. 3. *W3C Prov*. 4. *Workflow* Científico. 6. Sistemas de Gerência de *Workflow* Científico. I. Título.
CDD 004

Bibliotecária: Emanuele Santos
CRB-2 / 1309

Aos meus filho, pais, avós, família e amigos.

Agradecimentos

Quero agradecer, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada.

Agradeço as duas mulheres mais importantes de minha vida, minhas mães, Hilda e Elaine, por todo amor, carinho, sabedoria e força por todos esses anos.

Agradeço ao meu avô e melhor amigo Antônio Junqueira, por todo amor incondicional e ensinamentos que levarei por toda minha vida. Ao meu pai que apesar de todas as dificuldades esteve ao meu lado e me fortaleceu durante essa jornada, que para mim foi muito importante.

Ao meu irmão Marcos Vinicius por toda a força durante esta jornada e os meus irmãos, Lailton Junior, João Lucas, Davi e Pedro.

Agradeço em especial a minha esposa, melhor amiga e companheira de todas as horas, Leiciane Oliveira, pelo carinho, compreensão e amor. Por toda a paciência em minhas ausências e incentivo nas etapas mais complicadas e sempre me fazendo acreditar que tudo era possível.

À minha grande família, pelo amor e apoio de sempre, em especial aos meus tios Antonio Junqueira Filho, Welligton, Leonina e Katia Junqueira.

Ao meu orientador e amigo Ary Henrique Oliveira, pela orientação, apoio, confiança e pelo empenho dedicado à elaboração deste trabalho.

Aos meus amigos Felipe Cavalca, Paulo Henrique, Marcelo Claudio, Itamar Junior e o Damito Lopes pelo incentivo e pelo apoio constante. A todos os funcionários da Superintendência do Banco do Brasil, que me permitiram um desenvolvimento profissional e ético. Especialmente, ao Elmar e o Cleuby, pelas orientações e ensinamentos.

Agradeço a todos os professores por me proporcionarem conhecimento não apenas racional, mas manifestação do caráter afetividade da educação no processo de formação profissional. Por fim, a todos que direta ou indiretamente fizeram parte da minha

formação, o meu muito obrigado.

Resumo

A tecnologia de *Workflows* Científicos surgiram há pouco tempo e vem se tornando um paradigma para representar e gerenciar processos científicos. Os mesmos conduzem o fluxo de dados ao qual são envolvidos nos experimentos, de forma que vem a colaborar, obtendo resultados eficientes.

Porém a maioria destes *workflows* são computacionalmente intensivos e demandam por ambientes de alto desempenho de execução, podendo demorar dias, meses até mesmo anos. Atualmente também, a Computação em Nuvem vem se firmando, e se mostrando atrativo para a comunidade científica, pois este ambiente já apresenta todas as suas configurações embutidas e quando há necessidade de novos recursos este já as atualiza automaticamente. Fazendo com que assim os cientistas optem por migrarem seus *workflows* para a nuvem.

Porém, diversos provedores em nuvem vem ofertando seus serviços, com preços e configurações diversas, tornando-se assim, um problema para a escolha do que melhor adeque a necessidade do que está sendo procurado, pois a grande variedade de ofertas, gera um longo tempo de análises até se encontrar o que melhor se ajusta aos requisitos buscados.

Dessa forma este projeto tem por objetivo a implementação de um *middleware*, que analisa as diversas ofertas de servidores em nuvem, analisando os requisitos da busca, retornando o que melhor se adeque ao que foi procurado.

Abstract

The technology *workflows* scientific, emerged recently and has becoming a paradigm for representing and managing scientific processes. The same driving the data stream which are involved in the experiments, so that it comes to cooperate, achieving efficient results.

However most of these *workflows* are computationally intensive and demand for high-performance environments for implementing, can take days, months, even years. Currently also, Cloud Computing has established itself, and proving attractive to scientific community, because this environment already has all your settings and when embedded no need to have the new features this update automatically. Thus causing the scientists choose to migrate their *workflows* to the cloud.

However, many cloud providers has been offering its services, prices and various configurations, thus becoming a problem to choose the one that best fits the need of being searched because the wide variety of offerings, generates a long time until analyzes find the one that best fits the Hot requirements.

Thus this project aims to implement a *middleware* analyzing various offers of cloud servers, analyzing the requirements search, returning what best fits to what was sought.

Palavras-chave

Cloud Computer;
Open Provenance Model;
W3C Prov;
Workflow Científico;
Sistemas de Gerência de Workflow Científico

Abreviações

WS	:	Workflows Científicos;
SGWC	:	*Sistema de Gerenciamento de Workflows Científicos*;
NIST	:	*National Institute of Standards and Technology*;
BW	:	Workflows de Negócios;
W3C	:	*World Wide Web Consortium*;
EC2	:	*Elastic Cloud Computing*;
S3	:	*Simples System Storage*;
IaaS	:	Infraestrutura como Serviço;
PaaS	:	Plataforma como Serviço;
SaaS	:	Software como Serviço;
DaaS	:	Dados como Serviço;
MAUT	:	Multi Atribute Theory Utility;
WfMc	:	Workflow Management Coalition;
SDB	:	Simple DB;
AMI	:	Amazon Machine Instance;
MER	:	Modelo de Evolução Filogética;
OPM	:	Open Provenance Model;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A tecnologia de *workflows* científicos (*Scientific Workflows WS*) surgiu há pouco tempo e vem se tornando um paradigma para representar e gerenciar processos científicos, conduzindo o fluxo de dados que são envolvidos nos experimentos, de uma forma que ajuda a obter os resultados com mais eficiência. Os *workflows* científicos são gerenciados pelos Sistemas de Gerenciamento de *Workflows* Científicos (SGWfC), os quais são responsáveis por gerenciar a execução de cada uma das etapas que compõe o *workflow* ajudando na automatização de tarefas repetidas.

Além disso, o SGWfC's são responsáveis por armazenar todo histórico de informação da execução do *workflow* gravando-as em uma base de dados denominada base de proveniência [Freire et al., 2008]. Apesar das abstrações que os SGWfC's fornecem, registrar os eventos e as características de cada etapa dos experimentos é uma tarefa importante, para garantir a reprodutibilidade. A captura dos dados de proveniência são informações uteis das características de cada execução, que podem ser guardadas e analisadas, para reproduções futuras ou em uma possível validação do processo, garantindo confiabilidade aos resultados [Freire et al., 2008].

A maioria desses *Workflows* Científicos são computacionalmente intensivos e demandam por ambientes de alto desempenho de execução para obterem resultados em tempo eficaz, pois determinados *workflows* podem demorar semanas ou até meses para concluir sua execução. Diante disso, é necessário utilizar ambientes de computação distribuída e de grande capacidade de processamento para executar os *workflows*. Segundo Klinginsmith [Klinginsmith et al., 2011], uma necessidade comum na e-Science é ter um infraestrutura disponível para ser capaz de armazenar e gerenciar os dados, bem como ter o poder computacional necessário para processá-los.

Em muitos casos o cientista não tem acesso a recursos de computação de alto desem-

penho para executar seus experimentos, modelados ou não como *workflows* científicos, em grande parte por falta de recursos de financeiros para financiar a aquisição de equipamentos. Além disso, os recursos adquiridos tornam-se rapidamente desfasados sendo necessária sua substituição em médio prazo. Devido a problemas dessa natureza é necessário buscar formas de fornecimentos de recursos e infraestrutura de computação para possibilitar a execução de experimentos científicos.

Este projeto apresenta a hipótese de que o tipo de fornecimento de recursos propostos pelos fornecedores de computação em nuvem pode amenizar o problema de aquisição de infraestrutura de computação de alto desempenho. Porém, existe atualmente no mercado um número considerável de fornecedores de computação em nuvem ofertando seus recursos nos mais diferentes configurações e valores. A computação em Nuvem é um novo modelo de computação que provém recursos e serviços computacionais através da Internet de forma clara e sob demanda, abstraindo toda a complexidade da configuração da infraestrutura.

Segundo a definição da Nist (*National Institute of Standards and Technology*), o modelo em nuvem tem cinco características essenciais, três modelos de serviços (Software como um Serviço - SaaS, Plataforma como um Serviço - PaaS, Infraestrutura como um Serviço-IaaS) e quatro modelos de implantação (nuvem privada, pública, baseada em comunidade e híbrida) [Mell and Grance, 2011].

Dentre essas características as que tornam esse modelo mais vantajoso não apenas para empresas, mas também para os cientistas são a escalabilidade e elasticidade dos recursos. Os provedores de serviços em nuvem trabalham como o modelo de pagamento pague pelo que usar (pay-as-you-go). Essa característica permite que os clientes paguem apenas pelos recursos utilizados gerando assim economia com gastos de recursos ociosos.

1.1 Justificativa

A computação em nuvem vem se firmando e mostrando ser um ótimo atrativo para a comunidade científica, pois antes para realizar um experimento era necessário adquirir e configurar vários computadores tornando assim o processo custoso e demorado. Além disso, atualmente os *hardwares* são rapidamente atualizados e em pouco tempo se tornavam obsoletos gerando gastos desnecessários, entretanto esses problemas são supridos na computação em nuvem.

Os *workflows* científicos (Scientific Workflows - WS) são fluxos de trabalhos que usam

muitos recursos de *hardwares* levando dias para terminarem seus processos, e atualmente os cientistas estão migrando seus *workflows* para a nuvem buscando reduzir gastos. Como exemplo o SciCumulus [de Oliveira et al., 2010] que paraleliza os *workflows* em nuvens de computadores dividindo os processos com objetivo de gerar os resultados mais rápidos.

Hoje em dia vários provedores em nuvem veem surgindo com diferentes ofertas e serviços em nuvem, como por exemplo, temos a AWS Amazon, Gogrid e Rackaspace. Entretanto, alguns problemas podem ser apontados como o fato de não haver um padrão definido, alguns cobram por hora, outros cobram por mês, e outros até por ano. Outro problema é o fato das configurações das instâncias serem diferentes, na AWS Amazon as instâncias são classificadas em micro, médio, grande e extragrande, já no Rackspace existem sete instancias com diferentes recursos.

Para comparar e verificar a melhor oferta em nuvem, os cientistas demandariam tempo pesquisando qual o melhor serviço contratar, pensado nisso a abordagem de desenvolver o *middleware* para suprir essa necessidade e auxiliar o cientista durante a escolha, tendo como o principal objetivo maximizar recursos e minimizar os gastos.

1.2 Objetivos

No objetivo geral será descrito detalhadamente o escopo em que o projeto está inserido, bem como os principais objetivos a serem alcançados. Os objetivos específicos descrevem de forma sucinta cada uma das etapas principais que são fundamentais para que o objetivo maior seja alcançado.

1.2.1 Objetivo Geral

Desenvolver um *middleware* que a partir das características de um ambiente computacional, utilizando algoritmos e métodos de classificação possa analisar as ofertas de recursos dos fornecedores de computação em nuvem, de forma a classificar as ofertas que mais se aproximam das características dos recursos desejados.

1.2.2 Objetivos Específicos

- Realizar uma pesquisa bibliográfica e estudar artigos e livros relacionados a computação em nuvem, *workflows* científicos, teoria da utilidade de Multiatributo (Multi Attribute Theory Utility - MAUT) [Almeida, 2011] e o algoritmo K-means [Jain, 2010].

- Implementar os algoritmos K-means e o modelo e decisão baseado no MAUT adaptados para a classificação de ofertas de recursos.
- Projetar e implementar o módulo que extrai as informações das instâncias ofertadas pelos fornecedores de nuvem via web tais como CPU, Memória RAM, Disco Rígido, Sistema Operacional, Plataforma e Informações de Rede, bem como o valor cobrado dos mesmos.
- Projetar um modelo de dados, para os dados de proveniência usados no projeto.
- Implementar um módulo para o SGWfC Vistrails com objetivo de capturar as informações de proveniência dos *workflows* executados.
- Executar o Workflow SciEvol [Ocaña et al., 2012] para gerar parâmetros para realização dos testes
- Desenvolver um módulo *web* para oferecer uma interface gráfica para tornar o uso dos recursos desenvolvidos mais simples e usual.
- Implementar e executar os testes no *middleware* usando os algoritmos de classificação e modelo decisão, gerando gráficos para uma melhor comparação dos resultados e ao final fazer uma analise dos melhores resultados.

1.3 Estrutura do Trabalho

Esta dissertação está organizada da seguinte forma: No capítulo 2 são apresentados os fundamentos da literatura que compõe a base do objeto de estudo (Workflows Científicos e os Sistemas de Gerencia de Workflows Científicos, Computação em Nuvem, Benchmarks para Avaliação de Recursos e Algoritmos para Comparação de Proximidade de Características de Sistemas de Computador). Já o capítulo 3 mostra a metodologia, demonstrando a utilização das ferramentas e conceitos desenvolvidos ao longo da pesquisa. O capítulo 4 mostra e discute os resultados de avaliações desenvolvidas sob a plataforma proposta no projeto. Os capítulos 5 apresenta a conclusão da pesquisa e propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Workflows Científicos e os Sistemas de Gerência de Workflows Científicos

O termo *workflow* surgiu na década de 1999 por uma das principais organizações mundiais relacionada a *workflow*, WfMC (*Workflow Management Coalition*), esta o definiu como automação de processo de negócio, por completo ou em parte, no qual, os documentos informações e tarefas são passados de um participante para o outro a fim de que uma ação seja tomada de acordo com uma série de regras procedimentais [Deelman et al., 2009], tal definição é direcionada para *workflows* de negócio, porém pode ser expandida para o escopo científico.

Workflows científicos (WS) são caracterizados por serem responsáveis em encadear o fluxo de dados das atividades envolvidas em um experimento, coordenando a troca de informações entre as saídas de um processo e a entrada de outro de forma automatizada e sincronizada. Antes de serem utilizados na comunidade científica, os *workflows* já possuíam papéis importantes nos negócios. *Workflows* de negócios (BW) são utilizados quando há tarefas que necessitam de serem repetidas diversas vezes, sem que haja algum tipo de mudança durante a maneira de ser executada. Mesmo sendo utilizadas em ambos os escopos (científico e negócios), os *workflows* possuem particularidades em cada uma das duas áreas. *Workflows* Científicos (WS) tendem a sofrer diversas variações no mesmo experimento, além do que manipula um grande volume de dados, já os *workflows* de negócios (BW) possuem um nível maior de especialização e poucos são dinâmicos [Mattoso et al., 2010]. Deve-se ressaltar que devido ao grande volume de dados que *workflows* científicos manipulam, um dos grandes desafios é a heterogeneidade dos ambientes de execução que são utilizados para realizar os experimentos.

Workflows podem ainda ser classificados como: fluxos de dados (*data-flows*), fluxo de controle (*control-flow*) ou como híbridos, ou seja, quando possuem características de ambas as classificações citadas anteriormente. [Ludascher et al., 2006] diz que *Workflows* de negócios são modelados com foco no controle, onde uma etapa deve ser executada, para que uma posterior possa ser iniciada, ou seja, o *workflow* estabelece o que pode ser comparado a um controle de fluxo. Já os (WS) são caracterizados por possuírem dados como principal ponto a ser observado, ou seja, diferente dos *workflows* de fluxo de controle, uma etapa precisa ser terminada tendo em vista que, há a necessidade de se obter as informações desta para iniciar a próxima.

Antigamente, a modelagem de *Worflows* científicos era realizada manualmente pelos cientistas, à medida que surgia uma maior necessidade de se encontrar uma forma de automatização, pesquisadores começaram a construi-los através das linguagens de *script*, obtendo uma forma mais eficiente, com capacidade de repetição sem necessidade de grandes esforços. Porém, com a iniciação do conceito de computação distribuída, a qual foi necessária para manipular um grande fluxo de dados, fez com que surgisse uma nova forma de modelagem de *workflows*, o qual foi denominado Sistemas de Gerência de *Workflows* Científicos (SGWfCs), que serão detalhados nas seções a seguir.

2.1.1 Sistemas de Gerência de Workflows Científicos (SGWfC)

Sistemas de Gerenciamento de *Workflows* Científicos (SGWfC) surgiram devido ao desuso dos *scripts ad-hoc* para fins de representatividade de *workflows* científicos. Segundo [Freire et al., 2012] SGWC fornecem uma forma simples e abstrata, através de interfaces gráficas, para modelagem e gerenciamento de *workflows* científicos, sem exigir que o pesquisar tenha conhecimento avançado em programação.

Atualmente existem diversos SGWC (Taverna [Oinn et al., 2007], VisTrails [Bavoil et al., 2005] entre outros), todos com diferentes características e particularidades. Alguns destes foram desenvolvidos com o objetivo de fornecer suporte a *workflows* executados em ambientes centralizados e outros em ambientes distribuídos, alguns possuem suporte nativo a coleta de proveniência e outros apenas em níveis abstratos. Geralmente, *workflows* são representados como um grafo, onde os nós representam as atividades a serem executadas e as arestas estabelecem as dependências ou seqüencias que devem ser obedecidas.

2.1.2 Proveniência de Dados

Os *workflows* científicos surgiram como alternativa mais eficiente no processo de encadeamento das atividades de um experimento, fornecendo uma maneira robusta para o gerenciamento dos mesmos. Pode ser observado como um ponto fundamental na utilização de WS, a capacidade que os experimentos ganham de reprodutibilidade, à qual segundo Moreau permite que usuários possam compreender e verificar a forma como os dados foram derivados, garantindo a credibilidade dos resultados obtidos pelos pesquisadores [Moreau, 2011]. Porém, para tornar possível a reprodução dos experimentos científicos, devem-se ser observadas e capturadas informações relacionadas as entradas, saídas, processos e ambientes computacionais utilizados durante a experimentação, estas informações são denominadas, dados de proveniência.

Segundo [Miles et al., 2007], a proveniência de um dado denota todos os processos e transformações que este sofreu para ser agregado. Altintas diz que um sistema de coleta de proveniência deve possuir capacidades de criar e manter associações entre entradas e saídas além de coletar parâmetros e configurações automaticamente no decorrer da execução do *workflow* [Altintas et al., 2006]. Os metadados de proveniência devem ser representados em uma arquitetura que permita acesso, armazenamentos e buscas, permitindo compartilhamento dos resultados de uma forma colaborativa no ambiente de pesquisa.

2.2 Computação em Nuvem

A Computação em Nuvem representa um novo paradigma [Taurion, 2009], em que os recursos computacionais como processamento, memória e armazenamento não estão fisicamente presentes no local do usuário. Sendo transferidos para um ambiente onde possa ser acessado com facilidade de qualquer dispositivo que tenha acesso a internet.

Existem muitas definições de computação em nuvem, para Taurion, uma definição simples seria, um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataforma, aplicações e serviços disponibilizados na Internet [Taurion, 2009].

O NIST (National Institute of Standards and Technology) [Mell and Grance, 2011], define a computação em nuvem como um modelo que permite acesso, de modo simples, rápido e sob demanda, a recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) possibilitando adquirir ou liberar recursos computacionais sem necessidade de muito esforço.

O ambiente de computação em nuvem possibilita o fornecimento de serviços, onde o objetivo é facilitar as tarefas dos usuários fornecendo infraestruturas de fácil acesso, se baseando na abstração que oculta à complexidade da infraestrutura [Hayes, 2008]. A nuvem é uma representação para a internet ou infraestrutura de comunicação entre componentes arquiteturais.

A computação em nuvem tem características, que definem a e fazem a distinção perante os outros paradigmas [Mell and Grance, 2011]:

- Serviço sob demanda: é um atributo da Computação em Nuvem em que o *hardware* e o *software* podem ser adquiridos sob demanda.
- Amplo acesso à rede: é uma característica em que a conexão com a nuvem está disponível através da rede em uma interface de acesso heterogênea podendo ser acessada de qualquer dispositivo conectado a rede. Exemplo: smartphone, notebook, PCs, PDAs.
- Medição de Serviço: possibilita-se avaliar, controlar e aperfeiçoar a utilização de recursos computacionais nos serviços de armazenamento, processamento e largura de banda.
- Elasticidade: é uma vantagem da Computação em Nuvem em que os recursos podem ser alcançados de forma rápida e elástica, tendo como virtudes recursos disponíveis a qualquer momento e em uma grande quantidade passando a sensação de ilimitados.
- Disponibilização de recursos: é uma característica da Computação em Nuvem em que os recursos de computação estão agrupados para atender vários clientes usando a mesma nuvem que são dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários.

Perante o que foi exposto anteriormente, podemos dizer que trabalhar com a Computação em Nuvem pode trazer ao mundo da tecnologia uma série de vantagens e benefícios. A disponibilização de serviços da Computação em Nuvem é dividida em modelos, possibilitando adquirir-se do serviço mais adequado e compatível com a sua necessidade.

2.2.1 Tipos de Nuvens

Os modelos de implantação da computação em nuvem podem ser divididos em: público, privado, híbrido e comunidade [Mell and Grance, 2011], Tais modelos são implantados

dependendo da sua necessidade para a aplicação do processo do negócio e o seu ambiente operacional.

O modelo de implantação de nuvem Privada, tem como característica principal a infraestrutura da nuvem ser exclusivamente de uma organização que pode ser gerenciada pela própria organização ou por terceiros. A nuvem privada é construída sobre um *Data Center* em que são delegadas tecnologias de autenticação e autorização para acesso aos serviços.

A implantação do modelo de nuvem Pública é caracterizado pela infraestrutura da nuvem ser disponibilizada para o público em geral, podendo ser acessada por qualquer pessoa que conheça o endereço do serviço. Ao contrário do modelo de implantação privada não pode ser aplicadas restrições ao acesso da nuvem e nem utilizar técnicas para autenticação e autorização.

No modelo de implantação da nuvem Comunitária, tem como objetivo o compartilhamento da nuvem por diversas empresas se tornando uma comunidade que compartilhou seus interesses e as preocupações (requisitos de segurança, política, missão) que pode ser administrada por uma empresa desta nuvem ou por uma empresa terceira.

A implantação da nuvem Híbrida, tem como característica uma composição de duas ou mais nuvens (privada, comunitária ou pública) que atuam como se fosse uma única nuvem que permite a portabilidade de dados e aplicações. Nas nuvens hibridas, alguns serviços que não são críticos, ou seja, não há exigência de privacidade das informações, são direcionados para parte pública da nuvem, enquanto os outros, são mantidos na parte privada.

Os modelos de implantação para Computação em Nuvem devem ser usados de acordo com o tipo de negócio, camada da nuvem a ser usada e as características das informações a serem armazenadas, visando sempre adotar o modelo mais apropriado.

2.2.2 Tipos de Fornecimento de Serviços

A disponibilização de serviços da Computação em Nuvem é dividida em camadas, possibilitando adquirir-se do serviço mais adequado e compatível com a sua necessidade. Atualmente, o fornecimento de Computação em Nuvem, está dividido em três principais modelos de serviços, que são eles IaaS (Infraestrutura como Serviço), PaaS (Plataforma como Serviço) e SaaS (Software como Serviço). Mas, outros modelos de serviços estão sendo propostos, e vem sendo utilizados como o DaaS (Dados como Serviço).

A Infraestrutura como Serviço - IaaS fornece para o cliente os recursos computacionais virtualizados, ou seja, sua tarefa é fornecer os recursos de servidores, armazenamento e redes, de forma virtual e sob demanda [Mell and Grance, 2011]. Ele oferece toda a infraestrutura que o cliente necessita para o desenvolvimento e disponibilização de suas aplicações em nuvem. Alguns exemplos de IaaS são: Amazon EC2 e GoGrid.

A Plataforma como Serviço - PaaS pode ser definido como o fornecimento de um ambiente de computação como um serviço, com função de manter, testar e escalar aplicação em um ambiente integrado de desenvolvimento e execução [Lenk et al., 2009]. Este serviço é direcionado para desenvolvedores que pretendem alocar suas aplicações na nuvem [Mell and Grance, 2011], sem se preocupar com instalação e configurações de sistemas ou até mesmo infraestrutura de computadores. Google AppEngine e Amazon S3 são exemplos de PaaS.

Software como Serviço - SaaS é um modelo de disponibilização de software em forma de um serviço em que a empresa ou produtor do *software* desenvolve, opera, mantém e o disponibiliza para clientes que o usam de acordo com suas necessidades ou seja sob demanda e acessível por vários dispositivos clientes através de uma interface leve com um navegador de internet [Taurion, 2009]. Alguns exemplos de aplicações SaaS são: Dropbox, Google Apps, Skydrive, entre outros.

Dados como um Serviço - DaaS se baseia no conceito de que os dados, podem ser fornecidos sob demanda para o usuário [Olson, 2009], ou seja, dados de diversos formatos e origens podem ser manipulados ou acessados de uma forma transparente pelos usuários, independentemente da separação geográfica ou organizacional do fornecedor. Dados como um serviço pode eliminar a redundância e reduzir custos ao armazenar dados críticos em um único local, além de centralizar as atualizações.

2.2.3 Provedores de Computação em Nuvem

Nos dias atuais, temos vários provedores de serviços de computação em nuvem, estes se diferem na forma de como oferece o serviço, dependendo do foco, são oferecidos recursos diferentes. Diversas empresas começaram a disponibilizar os serviços de nuvem, entre elas temos algumas que se destacam por serem utilizadas com mais frequência como a Amazon AWS, GoGrid e Rackspace.

A Amazon Web Services (AWS) pode ser considerada como uma das organizações com maior representatividade no fornecimento dos recursos de nuvem computacional,

disponibilizando diversos serviços, como: Elastic Computing Cloud (EC2), Simple Storage Service (S3), SimpleDB (SDB), entre outras [Cloud, 2013].

- *Elastic Computing Cloud* (EC2), o cliente pode criar uma *Amazon Machine Instance* (AMI).
- O *Simple Storage Service* (S3), é o serviço da Amazon que fornece armazenamento de dados, que pode ser usado para armazenar e recuperar qualquer quantidade de dados, de qualquer lugar na que tenha acesso a internet.
- O *SimpleDB* (SDB), é um *framework* que permite aos usuários armazenar e recuperar dados não relacional, com alta disponibilidade e flexibilidade.

A GoGrid é uma companhia que oferece soluções de hospedagem em nuvem, seus usuários podem criar e gerenciar aplicações novas ou existentes, bem como o desempenho. Ela tem o seu foco voltado para o desenvolvimento, construção e implantação de infraestrutura na nuvem [GoGrid, 2013].

Os servidores da Gogrid, são de alto desempenho que estão pré-configurados para o usuário, podendo executar quase que instantaneamente, tendo o controle completo sobre suas instâncias, com acesso *root* e permissão de adicionar dados e aplicações [GoGrid, 2013].

A Rackspace, é uma nuvem híbrida que utiliza o Openstack o sistema operacional de código aberto para a nuvem. Ela fornece a infraestrutura que mais se adeque para as necessidades do usuário, podendo tanto obter servidores dedicados como servidores virtuais, personalizando o ambiente de hospedagem de cliente de forma única para cada [Rackspace, 2013].

2.2.4 Algoritmo K-means

K-means é um algoritmo usado em tarefas de clusterização(classificação), sendo um dos algoritmos mais eficientes para realizar este trabalho. Classificação k-means é um método muito usado para o particionamento automático de um dado dentro de K grupos. Toma-se randomicamente K pontos de dados para representar os centróides dos *clusters*. Posteriormente, cada um desses pontos é associado ao *cluster*, de modo que a distância deste ponto até o centróide em cada *cluster* é a menor de todas as outras distâncias calculadas. Surge um novo centróide para cada *cluster*, este é obtido pela média dos pontos do *cluster*, este processo se repete até que os centróides de cada *cluster* pare de se modificar, ou após um número de iterações definidas pelo usuário.

Segundo Jain [Jain, 2010] algoritmo K-means é popular devido a sua facilidade de implementação e sua ordem de complexidade O(n), onde N é o número de padrões. Contudo, o fato de o usuário ter que necessariamente especificar K, o número de *clusters*, com antecedência, pode ser visto como um ponto negativo (Carlantonio [di Carlantonio, 2001]). Afirma ainda [Jain, 2010] que, um dos pontos negativos do algoritmo k-means é que ele, não é sensível à seleção da partição inicial e pode convergir a um mínimo local do valor da função de critério se a partição inicial não for devidamente escolhida.

Adverte ainda Carlantonio [di Carlantonio, 2001] que o k-means não é indicado para descobrir *clusters* com formas não convexas ou *clusters* com tamanhos muito diferentes. O algoritmo k-means pode ser descrito em 5 passos básicos:

1º Passar valores para o centróide, neste passo são atribuído valores aos K centróides. O usuário define no início, o número K de centróides. Vale ressaltar que, deve-se colocar todos os pontos em um centróide qualquer para que o algoritmo possa iniciar a sua ponderação.

2º Geração de uma matriz com valores de distância entre os pontos e os centróides, este passo é calculado os valores de distância entre os pontos e os centróides. Este passo demanda um alto custo computacional, pois se temos N pontos e K centróides o cálculo será de $N\ x\ K$.

3º Avalia valores de distância do centróide para colocar cada ponto nas suas classes. A classificação funciona da seguinte forma: O ponto é incorporado ao centróide mais próximo ao dele, ou seja o ponto irá pertencer à classe representada pelo centróide que está mais perto do ponto. Quando o ponto não é incorporado a alguma classe, e isso ocorrer para todos os pontos, o algoritmo é finalizado.

4ª Cálculo de novos centróides para cada uma das classes, este é o momento em que os valores dos centróides são normalizados. Para cada classe possuindo mais de um ponto, um novo valor de centróide é calculado através da média dos atributos dos pontos que pertencem a esta classe.

5ª Loop, repetição até a convergência, o algoritmo volta ao passo 2, repetindo os procedimentos anteriores.

2.2.5 Algoritmo MAUT

Multi Attribute Utility Theory denominada MAUT, foi idealizada pela Escola Americana e baseia-se nos conceitos de modelagem de preferência tradicional, admitindo duas situações: preferência estrita e indiferença, ambas transitivas. Segundo [Keeney,] foi derivada da teoria da utilidade e teoria da decisão. Consiste em agregar diferentes pontos de vistas em uma única função que deve ser posteriormente otimizada.

Tal método permite conseguir uma abordagem consistente para problemas multicritério de decisão sob incerteza, inserida dentro de uma estrutura axiomática [Almeida 2010]. Ainda segundo [Almeida, 2011] a teoria da utilidade multiatributo implica em obter uma função analítica (ou valores de utilidade), ou seja, consiste em utilizar uma função multiatributo como método de agregação dos múltiplos objetivos do problema analisado.

A idéia básica para medição da utilidade multiatributo é que cada conseqüência de uma ação apresente um valor para diferentes dimensões. O MAUT procura medir estes valores em uma só dimensão por vez, e segue pela agregação destes valores sobre as dimensões, através do processo de obtenção do peso. Na teoria da utilidade multiatributo, admita-se que cada alternativa decisória resulte em consequências que são avaliadas pelo decisor, de acordo com cada critério. Para um mesmo problema pode-se adotar diferentes decisões por pessoa, conforme sua disposição em assumir riscos. Essas pessoas poderiam ser classificadas em três categorias em função dessa disposição: avesso ao risco, propenso ao risco e neutro ao risco [Raiffa, 1970].

Quando há incertezas nos problemas de decisão, é possível associar a cada alternativa diferentes consequências e suas probabilidades, inserindo no modelo uma etapa de modelagem probabilística, em complemento à modelagem de preferências do decisor. Nesse caso, as alternativas são avaliadas pela utilidade esperada [Gomes et al., 2009].

A teoria da utilidade multiatributo envolve a junção de vários pontos de vista (critérios) considerados em uma única função de síntese. Assim, o objetivo é encontrar a forma da função utilidade multiatributo que represente as preferências do decisor de acordo com os pontos de vista considerados. A forma da função utilidade depende das condições de independência dos critérios.

Capítulo 3

Metodologia

Este capítulo apresenta as atividades seguidas para o desenvolvimento do *middleware* de classificação de ofertas, que envolve a análise e escolha dos provedores de serviços em nuvem computacional, execução de *workflows* científicos apoiados por dados de proveniência, construção de um módulo para o Vistrails com a função de capturar e armazenar dados de proveniência, modelagem de um banco de dados para proveniência e do desenvolvimento de um sistema web integrando todas as funcionalidades desta abordagem com intuito de disponibilizar uma interface gráfica com ferramentas para analisar e apoiar a reprodutibilidade de experimentos científicos de forma simples e usual. Este capítulo está estruturado com as seguintes seções:

- Contextualização
- Ferramentas
- Métodos
- Desenvolvimento do ReploCloudAnalysis

3.1 Contextualização

Este trabalho tem como objetivo prover uma abordagem para classificação e seleção, de ofertas de recursos da infra-estrutura de computação em nuvem através de algoritmos de aproximação. Esta abordagem recebe as características de um ambiente computacional como parâmetro tais como: velocidade de processamento, capacidade de memória, capacidade do disco de armazenamento, arquitetura da máquina (32 ou 64 *bits*), sistema operacional e as características de rede.

Em seguida, obtém os diferentes tipos de ofertas de recursos de determinados provedores em nuvem (Amazon, Gogrid e Rackspace) na forma de instâncias de máquinas virtuais. E ao final, em posse das informações sobre a arquitetura recebida como parâmetro e dos diferentes tipos de instâncias ofertadas, aplica-se uma função de aproximação para classificar e selecionar as ofertas que possuem as características mais próximas da estrutura desejada. Além disso, tal escolha é feita com base no menor custo para aquisição da infra-estrutura.

O desenvolvimento desta abordagem seguiu os procedimentos de acordo com o diagrama de atividades da Figura 3.1, logo a primeira etapa do desenvolvimento foi realizar um levantamento dos provedores de serviços em nuvem com intuito de construir uma base de ofertas para realizar os testes, em seguida foi implementado os algoritmos de classificação e aproximação k-means e MAUT. Além disso, foram estudados os modelos de dados de proveniências definidos pelos documentos W3C Prov [opm,] e o *Open Model Provenance* [Moreau et al., 2011] com o objetivo de modelar uma base de dados para guardar a proveniência, sendo também implementado um módulo para capturar os dados do ambiente computacional durante a execução do *workflow* científico. Ao final o sistema web ao qual foi denominado ReproeScience-web foi desenvolvido.

3.2 Ferramentas

Mediante as necessidades que surgiram no decorrer do desenvolvimento do projeto, diversas ferramentas precisaram ser analisadas e utilizadas para alcançar os objetivos da pesquisa. Estas serão citadas resumidamente logo abaixo.

3.2.1 Vistrails

Vistrails[1] é um sistema gerenciador de *workflows* científico e proveniência de dados suportando visualização e exploração de dados. Vistrails é uma ferramenta *open source* e fornece uma infra-estrutura geral que pode ser combinando com diferentes sistemas e bibliotecas, o sistema é escrito em Python e tem suporte a Mac, Unix e Windows. Vistrails foi usado para execução do *workflow* SciEvol [Ocaña et al., 2012].

[1] http://www.vistrails.org/

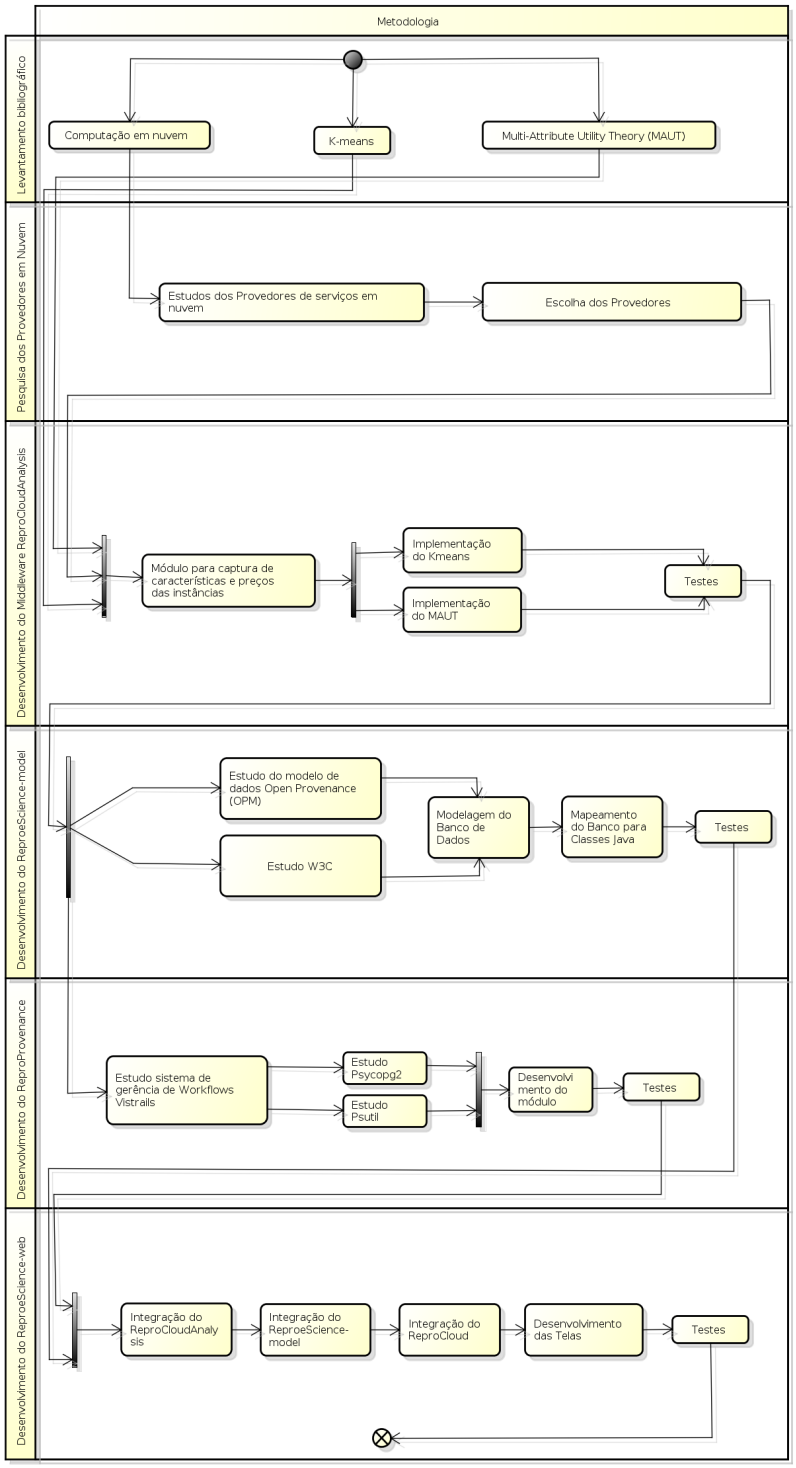

Figura 3.1: Metodologia do Projeto.

3.2.2 Java

Java[2] é uma linguagem de programação e uma plataforma de computação desenvolvida inicialmente pela empresa Sun Microsystem em 1990. Aplicativos desenvolvidos em Java

[2]http://java.com/

podem ser executados em qualquer dispositivo eletrônico independente de sua plataforma e sistema operacional, tais como computadores pessoais, televisores, *smartphones* e cartões inteligentes. Java, diferentemente das linguagens convencionais possuem código fonte compilado em *bytecode* para posteriormente ser executado por uma máquina virtual denominada JVM (*Java Virtual Machine*). Java foi utilizada como principal linguagem de programação no desenvolvimento dos módulos do projeto.

3.2.3 Google Gson

Gson[3] trata-se de uma biblioteca escrita em Java para trabalhar com o formato JSON (*JavaScript Object Notation*), Gson permite converter Objetos Java em sua representação equivalente em JSON, também é possível converter uma cadeia de caracteres JSON em objetos Java. Gson foi utilizada para converter os objetos JSON em todos os módulos do ReproeScience.

3.2.4 Java Server Faces

Java Server Faces[4] é um *framework web* escrito em Java que utiliza o modelo MVC(*model, view e controller*) que foi formalizada como padrão pela *Java Community Process* JSR 314. JSF é possível construir sistemas web com interfaces ricas baseadas em componentes, e foi projetado para facilitar significativamente a trabalhosa tarefa de escrever e manter as aplicações que são executadas em um servidor de aplicações Java. Por oferecer recursos para tornar a aplicação mais robusta e facilitar o desenvolvimento, este *framework* foi utilizado no desenvolvimento do módulo *web* do ReproeScience.

3.2.5 Apache Shiro

Apache Shiro[5] trata-se de um *framework* poderoso de segurança escrito em Java que executa autenticação, autorização, criptografia de dados e gerenciamento de sessão. Por oferecer esses poderosos recursos, este *framework* foi usado para autenticação e gerenciamento da sessão dos usuários no módulo web ReproeScience-web.

[3]https://code.google.com/p/google-gson/
[4]https://javaserverfaces.java.net/
[5]http://shiro.apache.org/

3.2.6 Hibernate ORM

Hibernate[6] é um *framework* ORM escrito em Java para mapeamento objeto-relacional que permite persistência de objetos Java em um banco de dados relacional. Esta ferramenta permite trabalhar com o banco de dados de forma simples orientada a objetos diminuindo a complexidade de trabalhar de forma estruturada. Por ter sido desenvolvido com base no JPA o hibernate foi usado como ferramenta ORM no módulo ReproeScience-web para mapear as classes desenvolvidas no módulo ReproeScience-model.

3.2.7 PostgreSQL

PostgreSQL[7] é um sistema gerenciador de banco de dados objeto relacional (SGBDOR), desenvolvido como projeto de código aberto e coordenando pelo PGDG (*PostgreSQL Global Development Group*). Tem compatibilidade com os principais sistemas operacionais (Windows, distribuições GNU/Linux e Unix) além de interfaces nativas de programação para diversas linguagens (C/C++, Java, .Net, Perl, Python, Ruby) e possui grande capacidade de gerência de dados e usuários concorrentes. PostgreSQL foi o gerenciador de banco de dados escolhido para armazenar os dados da proveniência do projeto.

3.2.8 Maven

Apache Maven[8] é uma ferramenta de gestão, automação de compilação de projetos desenvolvidos em Java, gerenciado pela *Apache Software Foundation*. É baseado no conceito de um modelo de objeto do projeto (POM), que descreve como um determinado projeto deve ser construído, suas dependências, componentes internos e externos, os armazenando em um sistema de repositório interno. Maven foi usado em topo o projeto para gerenciar as dependências e organizar os módulos do projeto.

3.2.9 Prettyfaces

Prettyfaces[9] é uma ferramenta baseada em *servrlets* de código aberto para construção de Urls amigaveis para Java Server Faces. PrettyFaces resolve o problema *"URL RESTful"* elegantemente, incluindo características tais como: ações de carregamento da página,

[6] http://hibernate.org/orm/
[7] http://www.postgresql.org/
[8] http://maven.apache.org/
[9] http://ocpsoft.org/prettyfaces/

integração com a navegação do JSF, atribuição dinâmica de ids para a *view*, passagem de parâmetros e compatibilidade de configuração com diversos *frameworks*.Prettyfaces foi usado junto com o JSF para deixar as *urls* mais amigáveis para os usuários.

3.2.10 Primefaces

Primefaces[10] trata-se de uma api de código aberto que provêm poderosos e belos componentes gráficos para utilização na *view* do JSF. Suas principais características é a de não necessidade de configurações para utilizá-lo, possui suporte a Ajax(*Asynchronous Javascript and XML*), conta com mais de 100 componentes e possui suporte completo aos principais navegadores. Primefaces foi usado no módulo *web* para desenvolver uma interface mais rica e robusta para os usuários.

3.2.11 Netbeans IDE

O Netbeans IDE[11] trata-se de um ambiente de desenvolvimento integrado de código aberto e gratuito com suporte de desenvolvimento nas principais linguagens de programação utilizadas atualmente, tais como: Java, C, C++, PHP, Groovy, Ruby. O ambiente de desenvolvimento é suportado por diferentes plataformas (Windows, MacOS, e diversas distribuições Linux). Netbeans foi usado no desenvolvimento de todo o projeto por oferecer vários desde verificação de erros a compilação dos projetos.

3.2.12 Glassfish

Glassfish[12] é um servidor de aplicação disponibilizado pela Sun Microsystem para a plataforma J2EE. Usado para implantar o ReproeScience-web disponibilizando o serviço de cliente/servidor.

3.2.13 Jsoup

Jsoup[13] trata-se de uma biblioteca Java para manipular estruturas em HTML. Ela fornece uma API muito poderosa para extração e manipulação de dados, usando DOM, CSS e

[10] http://www.primefaces.org/
[11] https://netbeans.org/
[12] https://glassfish.java.net/
[13] http://jsoup.org/

métodos jQuery. Jsoup foi usado para minerar as ofertas na páginas dos provedores de serviços em nuvem.

3.2.14 Google Charts

Google Charts[14] é uma biblioteca escrita em JavaScript que fornece a geração de gráficos iterativos, inicialmente foi desenvolvida para uso interno da corporação Google, mas em 2007 foi disponibilizada para o uso por desenvolvedores web. Google Charts foi usado no módulo *web* para criar gráficos iterativos melhorando a visualização dos resultados obtidos pelo ReproCloudAnalysis.

3.2.15 JsPlumb

JsPlumb[15] é uma biblioteca para criação de grafos e diagramas animados interativos escrito em JavaScript. JsPlumb foi usado para gerar o gráfico para visualização dos dados de proveniência.

3.2.16 Python

Python[16] é uma linguagem de programação de alto nível interpretada, tendo como características a orientação a objeto e tipagem dinâmica e forte. Python foi desenvolvida inicialmente por Guido Van Rossum em 1991 e atualmente tem seu desenvolvimento comunitário gerenciado pela organização sem fins lucrativos *Python Software Foundation*. Python foi usado como linguagem para criação do módulo ReproProvenance.

3.2.17 Psycopg2

Psycopg2[17] é um adaptador de banco de dados PostgreSQL para a linguagem Python. Foi projetado para aplicações *multithreads* que abrem e fecham sessões diversas vezes inserindo, buscando e atualizando os dados diversas vezes. Psycopg2 foi usado no módulo ReproProvenance para conectar ao banco de dados PostgreSQL.

[14] https://developers.google.com/chart/
[15] http://jsplumbtoolkit.com/demo/home/jquery.html
[16] http://www.python.org/
[17] http://initd.org/psycopg/

3.2.18 PsUtil

PsUtil[18] trata-se de uma biblioteca escrita em Python que fornece funções para monitoramento de informações do sistema operacional, com o PsUtil é possível ter acesso as características de hardware da máquina executada, tais como, memória RAM, disco rígido, cpu, informações da rede, sistema operacional etc. Este pacote foi usado no módulo ReproProvenance para pegar as informações do ambiente computacional necessários para o ReproCloudAnalysis.

3.3 Métodos

3.3.1 Workflow SciEvol

SciEvol [Ocaña et al., 2012] é um *workflow* científico modelado para ser executado no SWFMS Vistrails com objetivo de suportar atividades de análise de modelos de evolução filogenética (MER). De acordo com [Ocaña] o SciEvol explora entradas, estimando parâmetros e hipóteses para explorar o processo de evolução, estimando taxas de substituição sinônimas e não sinônimas para detecção de purificação, ponto neutro ou evolução positiva em sequências de nucleotídeos que codificam as proteínas e identifica sítios selecionados.

O SciEvol faz uso de *softwares* de bioinformática: (i) Mafft [maf,] para produzir sequências alinhadas de nucleotídeos, (ii) Readseq [rea,] para converter entre o formato FASTA [fas,] alinhado e PHYLIP [PLOTREE and PLOTGRAM, 1989], (iii) RAxML [rax,] que produz árvores filogenéticas, e (iv) CodeML [cod,] responsável por produzir dados de análise evolucionaria a partir de um modelo e uma árvore filogenética. Deste modo, cada atividade do SciEvol é implementada usando o módulo *Python Source* do Vistrails, o código desenvolvido em Python responsável por executar os *softwares* e tratar as entradas necessárias.

Além disso, Ocaña explica que o *workflow* SciEvol tem um fluxo de execução composto por doze atividades:

- A primeira atividade é um código escrito em Python com a finalidade de formatar as entradas dos dados removendo codão de parada.
- A segunda atividade é a construção do alinhamento múltiplo de sequências (MSA) usando o aplicativo Mafft.

[18] https://code.google.com/p/psutil/

- A terceira atividade é a conversão do MSA para o formato usado pelo PHYLIP.
- A quarta atividade é composta por um script Python com a função de formatar o arquivo PHYLIP.
- A quinta atividade Construção da árvore filogenética no formato Newick [PLOTREE and PLOTGRAM, 1989] usando o RAxML.
- A sexta até a decima primeira atividade são responsáveis pela a execução da análise evolutiva composta por seis fases, executando os Modelos de Substituição de codão M0, M1, M2, M3, M7 e o M8.
- A Última atividade é análise dos dados evolutivos.

Portanto, ao final o ScilEvol gera saídas com informações para manipular e analisar os resultados dos dados genômicos em larga escala, de modo a inferir hipóteses evolutivas.

3.3.2 Open Provenance Model (OPM)

O OPM (*Open Provenance Model*) é um modelo para representação de proveniência classificada como retrospectiva segundo [Moreau et al., 2008], que foi desenvolvida pelas edições do *Provenance Challenges* [opm,] motivados pelos diversos desafios e necessidades de entender as diversas formas de representações para proveniência, bem como seus aspectos comuns, e suas diferenças.

O OPM foi projetado para atender o seguintes requisitos básicos:

- Permitir que informações de proveniência possam ser trocadas entre sistemas, por meio de uma camada de compatibilidade com base em um modelo de proveniência compartilhado.
- Permitir que os desenvolvedores criem e compartilhem ferramentas que operam em tal modelo de proveniência.
- Definir a origem do dado de forma precisa.
- Apoiar a representação digital de proveniência para qualquer "coisa" produzida por sistemas computacionais ou não.
- Permitir que vários níveis de descrição possam coexistir.

- Definir um conjunto básico de regras que identifiquem as inferências válidas que pode ser feita na representação proveniência.

Os objetos no OPM podem ser representados por um grafo dirigido e são definidos em três categorias básicas do modelo, estes são representados pelos nós do grafo:

- Artefato: representa um dado de estado imutável, que pode ter um corpo físico em um objeto físico, ou uma representação digital em um sistema de computador.
- Processo: representa ações realizadas ou causadas por artefatos, e resultam em novos artefatos.
- Agente: representa entidades contextuais agindo como um catalisador de um processo, permitindo, facilitando, controlando, ou afetando sua execução.

Já as dependências entre os elementos, são representados pelas ligações entre os nós, pertencendo a algum dos seguintes tipos: *wasGeneratedBy* (R), *used* (R), *wasDerivedFrom*, *wasControlledBy* (R), *wasTriggeredBy*. O sentido da seta caracteriza o relacionamento entre os objetos, origem representa o feito e o destino a causa. Alguns tipos de dependência (*wasGenereted, used, wasControlledBy*) podem incluir opcionalmente um texto entre parênteses, representando por "R", e indica o role que um agente exerce sob um processo. Um *role* é uma alternativa para representar diferentes funções entre artefatos ou agentes e processos.

Portanto, o modelo OPM fornece uma maneira bem estruturada para representar os dados de proveniência, garantindo então, uma forma de persistência das informações relativas a origem de um dado, processos que o utilizaram ou modificaram, e até mesmo quais agentes foram responsáveis pela situação atual do elemento.

3.3.3 W3C Provenance

W3C Prov é uma família de documentos desenvolvida pelo grupo de trabalho de proveniência denominado W3C (*The World Wide Web Consortium*). Estes documentos são responsáveis por estruturar a padronização das informações dos dados na web. Esta família de documentos foi desenvolvida na quarta edição do Provenance Challange [opm,]. O W3C Prov é composto pelos seguintes documentos:

- PROV-DM [pro, a] é o modelo de dados para a proveniência;

- PROV-CONSTRAINTS [pro, b]: é um conjunto de restrições aplicadas ao modelo de dados;
- PROV-O [pro, f]: a ontologia PROV, uma ontologia OWL2 permitindo o mapeamento de PROV para RDF;
- PROV-N [pro, e]: uma notação para proveniência destinado a consumo humano;
- PROV-AQ [pro, c]: os mecanismos de acesso e consulta de proveniência ;
- PROV-primer [pro, d]: *primer* para o modelo de dados PROV.

A família PROV tem como núcleo o documento PROV-DM, com a função de mapeamento dos dados de proveniência capturados, este documento representa um modelo de dados relacional customizável. Também é fornecida uma forma de validação dos dados de proveniência (PROV-CONSTR) por meio de restrições, e ainda um documento com especificações relativas ao acesso à proveniência na *Web* (PROV-AQ).

Para evitar redundância entre o PROV Model Primer e os conceitos de OPM citados anteriormente, a equipe do W3C definiu alguns termos básicos bem como possíveis tipos de relacionamentos, como pode ser visto na figura 3.2.

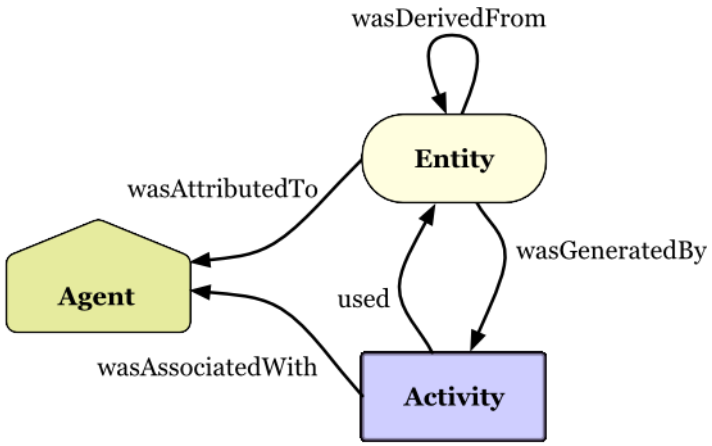

Figura 3.2: Estrutura dos tipos de dados e dos relacionamentos do W3C PROV.

Fonte: Página do W3C[19]

- Entidade: segundo as definições do modelo W3C PROV, qualquer coisa física, digital, conceitual ou outros tipos, são chamadas de entidades. Registros de proveniência podem descrever proveniência de entidades, e essas proveniências podem referenciar várias outras entidades.

[19]http://www.w3.org/TR/prov-primer/#intuitive-overview-of-prov

- Atividades: são como entidades passam a existir e como seus atributos podem mudar para ser tornar novas entidades, utilizando-se de entidades previamente existentes.

- Uso e Geração: entre entidades e atividades relacionam-se de duas principais formas: uso e geração. O uso é caracterizado com a utilização de uma entidade por uma atividade, da mesma forma, uma entidade é gerada a partir de uma atividade.

- Agentes e Responsabilidades: um agente assume um papel em determinada atividade, de forma que pode ser atribuído algum grau de responsabilidade em relação a atividade que está ocorrendo. Um agente pode ser uma pessoa, um software, uma organização, ou outras atividades que podem receber atribuição de responsabilidades. Quando o agente é responsável por alguma atividade, PROV define que existe uma associação com o agente, um agente por estar associado a várias atividades e vice e versa.

- Papel: especifica o relacionamento entre uma entidade e uma atividade. Papéis ainda descrevem como agentes estão envolvidos em uma atividade, qualificando sua participação na mesma ou especificando sua responsabilidade.

- Derivação e Revisão: quando a existência, conteúdo ou outras características de uma entidade são fortemente relacionadas com a de outra, pode-se dizer que a primeira foi derivada da segunda. Revisões e citações de um determinado documento, são um exemplo de especializações de derivação permitidos pela PROV .

- Planos: as atividades podem seguir procedimento pré-determinados, como por exemplo, *workflows*. PROV referencia a esses, em sua totalidade, como planos.

- Tempo: Este é geralmente um aspecto fundamental de proveniência. PROV permite a descrição da temporização dos acontecimentos, como por exemplo, quando uma entidade foi gerada ou utilizada, ou até mesmo quando foi iniciada e finalizada.

Deste modo, o W3C PROV pode ser considerado um modelo genérico para representação de dados de proveniência [Missier et al., 2013], e vem sendo amplamente utilizado para capturar proveniência de experimentos apoiados por SW. Diversos SGWfC's já utilizam o PROV como meio de representação, como por exemplo o Taverna e Vistrails.

3.4 Desenvolimento do ReproCloudAnalysis

O ReproCloudAnalysis tem por objetivo fornecer uma abordagem de classificação de instâncias ofertadas por provedores de serviço em nuvem, tornando-se uma ferramenta de auxílio ao cientista para escolha de qual tipo de instância melhor se adapta as suas necessidades computacionais. As instâncias, são servidores virtuais que consistem em várias combinações de memória, CPU, armazenamento em disco, capacidade de rede, plataforma e sistema operacional, e para obter um uso eficiente dos recursos, é aconselhável escolher a instância que mais se ajusta aos requisitos dos aplicativos.

Para que o ReproCloudAnalysis alcançasse sua finalidade foi necessário antes de tudo, definir quais provedores em nuvem seriam utilizados para formar uma base de ofertas de instâncias, sendo definidos dois parâmetros para tal escolha, aos quais serão descritos a seguir:

- Disponibilização de informações das instâncias ofertadas em sua página na internet tais como as configurações de hardware e preços por hora de consumo.
- O Fornecimento de uma API para comunicação com os serviços oferecidos.

Dos provedores pesquisados três se encaixaram nos parâmetros definidos, sendo eles: AWS Amazon, Gogrid e Rackspace. Estes provedores oferecem uma página na internet com as configurações e as APIs para comunicação, estas, no caso do Gogrid e o Rackspace oferecem *webservices* RESTFul que usa o paradigma *Representational State Transfer* (REST) [Introducing,], enquanto que a AWS Amazon oferece um API escrita em JAVA.

Para capturar as informações das instâncias nos casos dos provedores Gogrid e RackSpace, foi realizado uma análise no DOM HTML por algoritmos feitos especificamente para cada provedor, isto se deve pela peculiaridades de suas páginas e pela forma nos quais os dados são disponibilizados. Porém, no caso do provedor AWS Amazon, os dados são fornecidos em formato JSON (*JavaScript Object Notation* - Notação de Objetos JavaScript) [Ihrig, 2013], desta maneira os dados são recuperados por uma simples requisição HTTP GET. Os objetos no formato JSON são estruturados por coleção de dados chave e valor em uma *string*, sendo assim necessário realizar uma conversão para objetos Java, para isto foi usado a ferramenta Gson. A Tabela 3.1 lista as ferramentas utilizadas para o desenvolvimento deste middleware.

O vetor na figura 3.3 representa as configurações das instâncias em máquinas virtuais ofertas sendo que cada índice representa uma determina característica capturada na

Ferramenta	Aplicação
Java	Linguagem de Programação
NetBeans	Ambiente de Desenvolvimento
Maven	Automação de Compilação
Gson	Parser JSON
Jsoup	Parser HTML

Tabela 3.1: Ferramentas utilizadas no desenvolvimento do ReproCloudAnalysis.

página:

Vetor									
0	1	2	3	4	5	6	7	8	9
Memoria RAM	Disco Rígido	Cores	vCPU	Rede Interna	Rede Externa	Plataform a	Sistema Operacional	Região	Preço

Figura 3.3: Vetor de características das ofertas.

Além disso, as classes responsáveis pela análise e recuperação destas instâncias implementam interfaces com métodos utilizados pelos algoritmos de classificação que definem os parâmetros de entrada e o tipo de dado a ser retornado. As interfaces e as classes podem ser vistos no diagrama abaixo (figura 3.4):

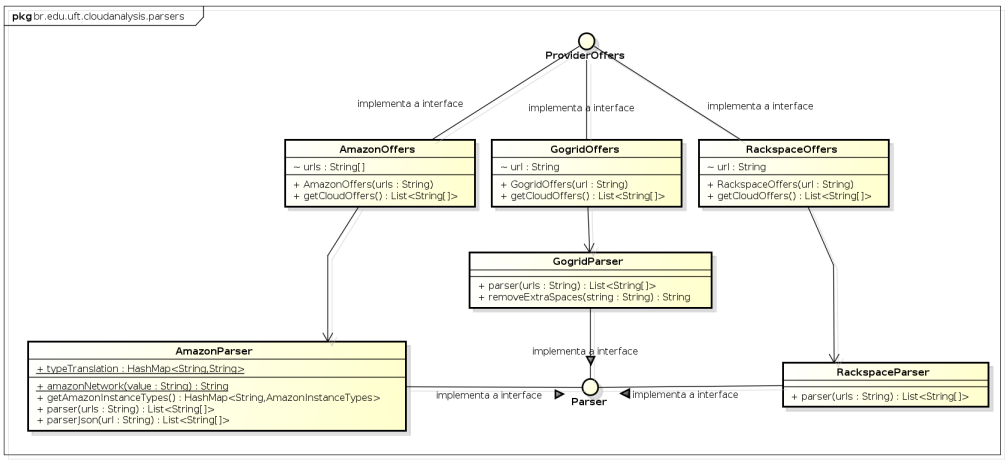

Figura 3.4: Diagramas de classes dos provedores.

As classes AmazonOffers, GogridOffers e RackspaceOffers implementam a interface ProviderParses que assina um método chamado de "getCloudOffers"que retorna um lista de vetores do tipo "String"que seguem o padrão do vetor mostrado na figura 3.3. Cada provedor tem seu parser responsável pela capturar os dados nas paginas e organizá-los no formato definido pela interface.Logo, esta abordagem oferece uma forma de adicionar posteriormente novos provedores de serviço em nuvem.

O middleware ReproCloudAnalysis foi desenvolvido de um modo no qual os métodos

de classificação seguissem um determinado fluxo de eventos para geração dos resultados. A Figura 3.5 mostra o diagrama de atividades deste fluxo.

Inicialmente os métodos de classificação recebem como parâmetros um vetor contendo as características da máquina e a lista dos provedores de serviço em nuvem. Depois é solicitado a estes provedores as ofertas e em seguida são convertidas para um formato no qual o algoritmo trabalha. Uma vez que estes dados estão no formato desejado é aplicado o método de classificação e ordenação das ofertas classificadas.

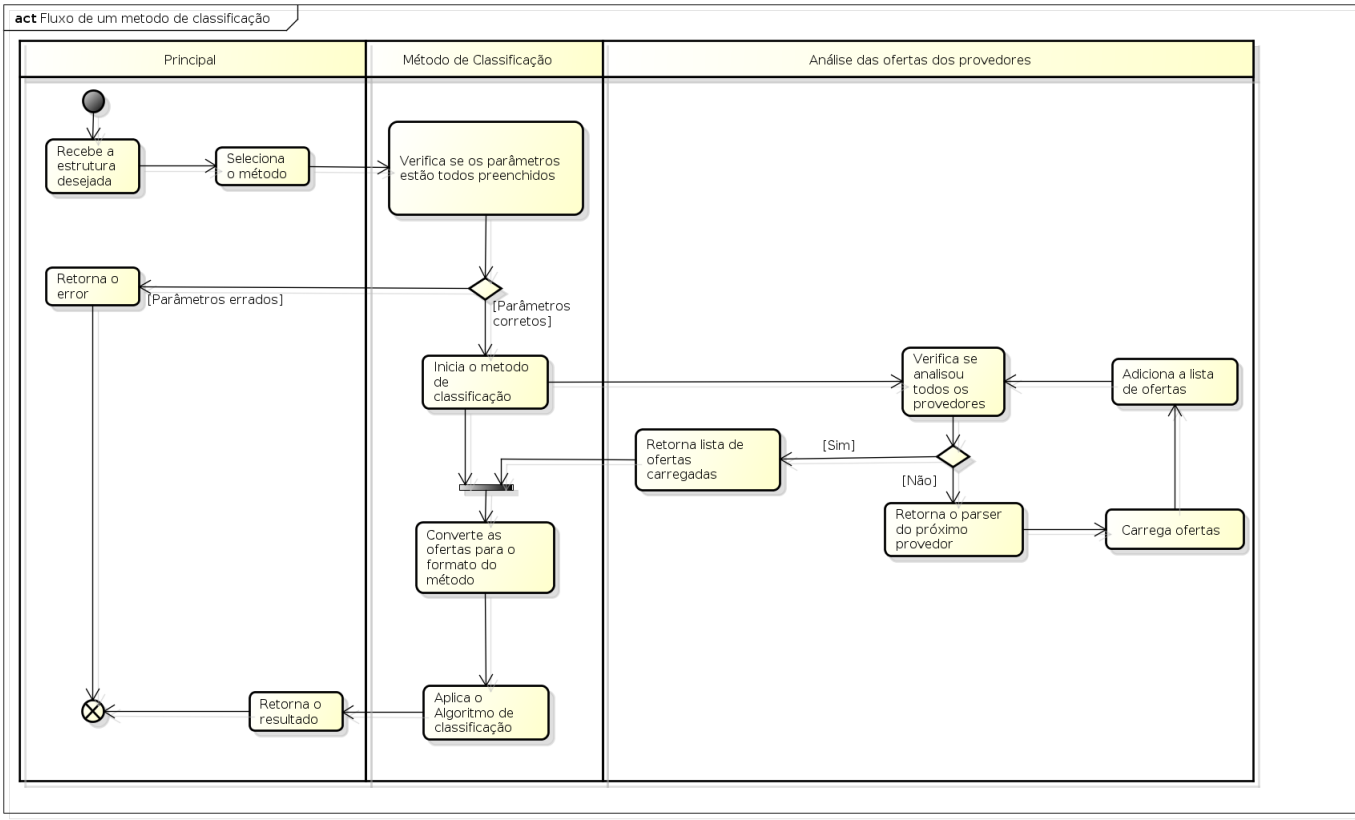

Figura 3.5: Diagrama de atividades dos métodos de classificação.

Os métodos K-means e MAUT foram escolhidos para serem implementados neste projeto, eles seguem os mesmos fluxos apresentados na figura, entretanto os algoritmos tem abordagens diferentes para a classificação. As próximas subseções irão detalhar a implementação dos algoritmos.

3.4.1 Kmeans

O algoritmo K-means recebe como parâmetros um vetor contendo as configurações de um ambiente computacional no qual deseja aproximar e a quantidade de interações que serão realizadas. Antes de iniciar as interações é realizada a escolha dos centróides, este proce-

dimento cria três centróides sendo que o primeiro é o ambiente computacional desejado e os outros são com valores 50% a menos e 50% a mais do valor desejado. Desta maneira são criados três clusters correspondentes a um centróide para classificar as ofertas, além disso, os clusters são identificados com valores entre 0 e 3.

Em seguida são convertidas as ofertas solicitadas aos provedores para uma estrutura de dados que o algoritmo trabalha, a subseção explica detalhadamente as classes usadas pelo algoritmo. O algoritmo define uma variável para armazenar a distancia mínima encontrada, responsável para definir a cluster a oferta pertence. Após a conversão é realizado a interação com as ofertas calculando para cada centróide a somatória da distância dos atributos, abaixo a fórmula usada:

$$dist = \sqrt{\sum_{i=1}^{n}(atrDesejado_i - atrOferta_i)^2}$$

Onde i é a posição do atributo no vetor semente e n a quantidade de atributos. Depois de realizada a somatória para o cluster é verificado se a distancia encontrada é menor do que a distância mínima, se for menor a oferta é definida ao cluster, se não a iteração continua.

Após realizar esses passos para todas as ofertas os centróides são atualizados a fim de encontrar o novo centro para os clusters. Ao fim das iterações o algoritmo ordena as ofertas por seus preços do primeiro cluster no qual definido com os valores desejados.

Algoritmo 1: Algoritmo K-means

Entrada:

Um conjunto finito $s = \{s_0, s_1, s_2, \ldots, s_8\}$ de características desejadas, sendo esses valores numéricos.

Quantidade de interações it

Saída: Conjunto de ofertas classificadas.

```
inicio
    ofertas ← carregarOfertas();
    centroides ← gerarCentroides();
    interacao ← 1;
    while interacao < it do
        foreach oferta ∈ ofertas do
            distanciaMin ← ∞;
            foreach centroide ∈ centroides do
                distanciaOferta ← 0;
                foreach atributo ∈ atributos do
                    distanciaOferta =
                    distanciaOferta + (atrDesejado − atrOfertado)²;
                end
                distanciaOferta ← √distanciaOferta if
                distanciaOferta < distanciaMin then
                    distanciaMin ← distanciaOferta;
                    adicionarOferta(centroide, oferta);
                end
            end
        end
        atualizarCentroide();
    end
    ofertasClassificadas ← classificarOfertas(centroide);
    return ofertasClassificadas;
fin
```

3.4.2 Diagrama de Classes K-means

Está subseção tem como objetivo apresentar o diagrama de classes do algoritmo K-means, a figura 3.6 exibe o diagrama de classes usadas na programação.

A classe KmeansMethod implementa a interface ProviderOffers que define o método *execute* responsável por executar o algoritmo K-means, este método recebe como parâmetro o vetor semente com as características do ambiente computacional para classificação das ofertas. Esta classe tem também uma lista de provedores do tipo ProviderOffers usados para retornar as ofertas para serem usadas pelo algoritmo.

Já a classe KmeansData representa as ofertas no formato utilizado por esta abordagem e suas características são representadas pela classe KmeansAtribute que define qual o tipo do valor e o valor do atributo.

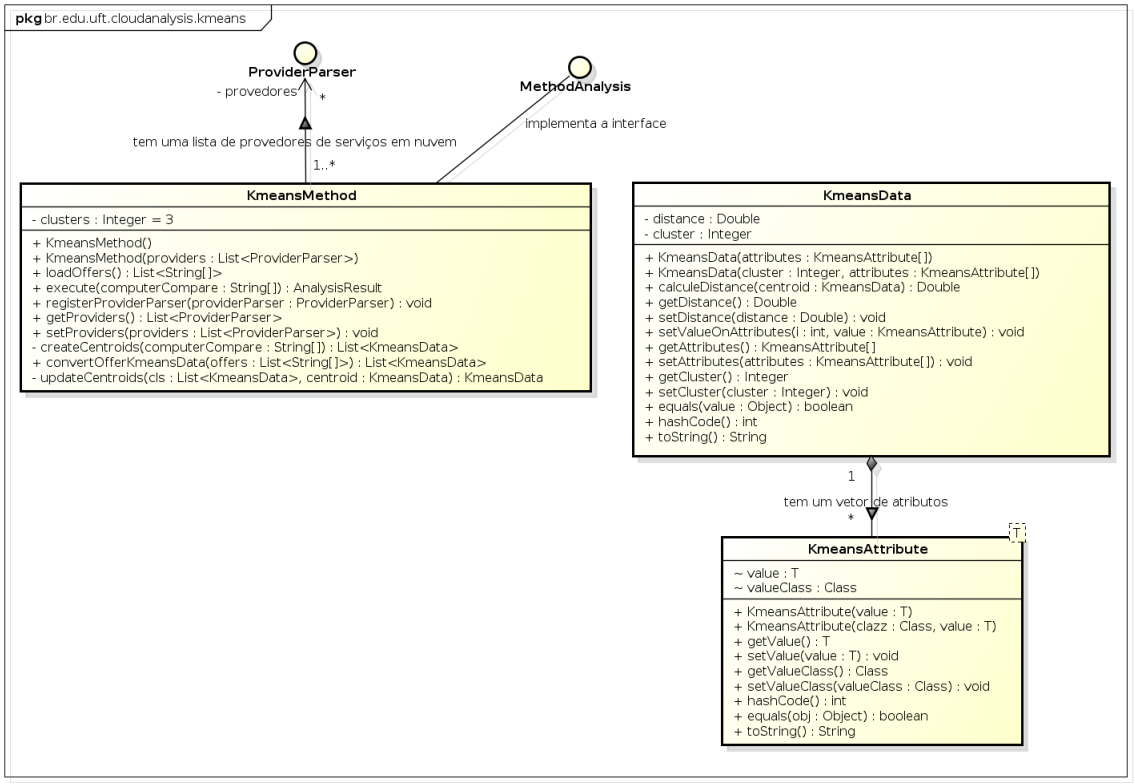

Figura 3.6: Diagrama de classes do algoritmo K-means.

3.4.3 Multi Attributte Utility

O algoritmo recebe dois parâmetros para realizar a classificação, o primeiro tem como função determinar quais os valores dos pesos para cada atributo e grupo, sendo este parâmetro uma lista de vetores, onde cada vetor corresponde a um grupo e sua posição

determina sua prioridade. O segundo parâmetro recebe um vetor contento os atributos das características do ambiente computacional no qual deseja comparar. O pseudo Algoritmo 2 exibe os passos realizados pelo algoritmo.

Um exemplo de representação dos grupos em formato JSON pode ser visto abaixo:

$$\textbf{grupos} = \textbf{[[0, 1, 2, 3, 4, 5], [6, 7, 8]]}$$

Esse mesmo exemplo na interface gráfica desenvolvida no módulo ReproeScience-web pode ser visto na figura 3.7.

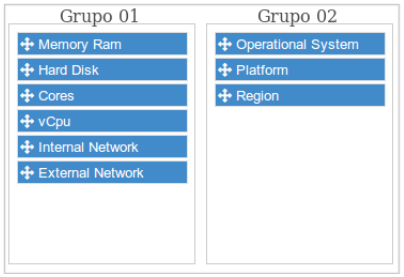

Figura 3.7: Grupos com atributos do ambiente computacional.

Os pesos dos grupos são calculados de acordo com suas ordens na lista e do mesmo jeito é realizado o calculo dos pesos dos atributos.

Em seguida são solicitadas as ofertas ao analisador dos provedores de serviços em nuvem e depois sendo iteradas a fim de calcular o valor da avaliação da oferta (ofertaAvalicao). Para cada oferta é realizada outra iteração com os grupos selecionados e para cada grupo é realizado uma iteração com seus atributos realizando o calculo da distancia euclidiana com os valores do atributo e do atributo desejado. Um exemplo da fórmula sendo aplicada com os atributos pode ser vista abaixo:

$$dist = \sqrt{(atr_1 - atr_2)^2}$$

Depois é calculada a razão da distância ao atributo da oferta e verificando para que a razão não ultrapasse o valor de 1.0, esta condição garante que os valores fiquem normalizados. Após esse passo uma subtração é realizada na razão com um valor 1.0 a fim de deixar a visualização das ofertas em ordem crescente pela taxa, logo, as taxas mais próximas de zero são as que mais se aproximam do ambiente computacional desejado. Por fim é somado a variável atrSomatoria o resultado da multiplicação da razão ao peso do atributo corresponde ao grupo.

Após o término da somatória a taxa do grupo é multiplicada com o peso corresponde ao grupo, este procedimento é realizado até terminar a iteração de todos os grupos, ao final obtendo a taxa de classificação da oferta. Após iterar com todas as ofertas é realizada uma ordenação analisando as taxas com seus preços, a lista resultante é retorna pela função que implementa o algoritmo.

Algoritmo 2: Algoritmo MAUT

Entrada:

Um conjunto finito $s = \{s_0, s_1, s_2, \ldots, s_8\}$ de características desejadas, sendo esses valores numéricos.

Uma lista finita $g = \{g_0, g_1, g_2, \ldots, g_m\}$ contendo grupos para geração de pesos.

Saída: Conjunto de ofertas classificadas.

inicio

$ofertas \leftarrow carregarOfertas()$;
$ofertasClassificadas \leftarrow \{\}$;
foreach $oferta \in ofertas$ **do**
$ofertaAvaliacao \leftarrow 0$;
for $i \leftarrow 0$ ***to*** m **do**
$atrSomatoria \leftarrow 0$;
$atrPesos \leftarrow pesosAtributos(g_i)$;
$atributos \leftarrow grupoAtributos(g_i)$;
foreach $atributo \in atributos$ **do**
$atrDesejado \leftarrow s_{atributo}$;
$atrOfertado \leftarrow oferta_{atributo}$;
$distancia = \sqrt{(atrDesejado - atrOfertado)^2}$;
$razao \leftarrow 0$;
if $distancia > 0$ **then**
$razao \leftarrow \frac{distancia}{atrOfertado}$;
end
if $razao > 1.0$ **then**
$razao \leftarrow 1.0$;
end
$razao \leftarrow 1.0 - razao$;
$atrSomatoria \leftarrow atrSomatoria + (razao * atrPesos_{atributo})$;
end
$ofertaAvalicao \leftarrow ofertaAvaliacao + (atrSomatoria * pesoGrupo(g_i))$;
end
$adicionarAvaliacao(oferta, ofertaAvaliacao)$;
end
return $ofertasClassificadas$;

fin

3.4.4 Diagrama de Classes MAUT

Esta seção tem como objetivo explicar as classes programadas para o algoritmo MAUT, a fim de auxiliar na compreensão das classes a figura 3.8 exibe o diagrama de classes.

A classe MautMethod contém a implementação do pseudo algoritmo MAUT (Algoritmo 2, esta classe tem uma lista de provedores de nuvens do tipo ProviderOffers, além disso ela implementa a interface MethodAnalysis que assina o método execute responsável por executar o algoritmo, este método recebe como parâmetro um vetor semente com os valores do ambiente computacional desejado. A classe MautGroup representa os grupos e os atributos para definir os pesos necessários usando no calculo da distância.

Já a classe MautData é responsável por representar as ofertas, deste modo definindo o tipo de dados que o MAUT trabalha, e esta classe tem um vetor de características das ofertas que são do tipo MautAttribute que define o tipo e o valor do atributo da ofe

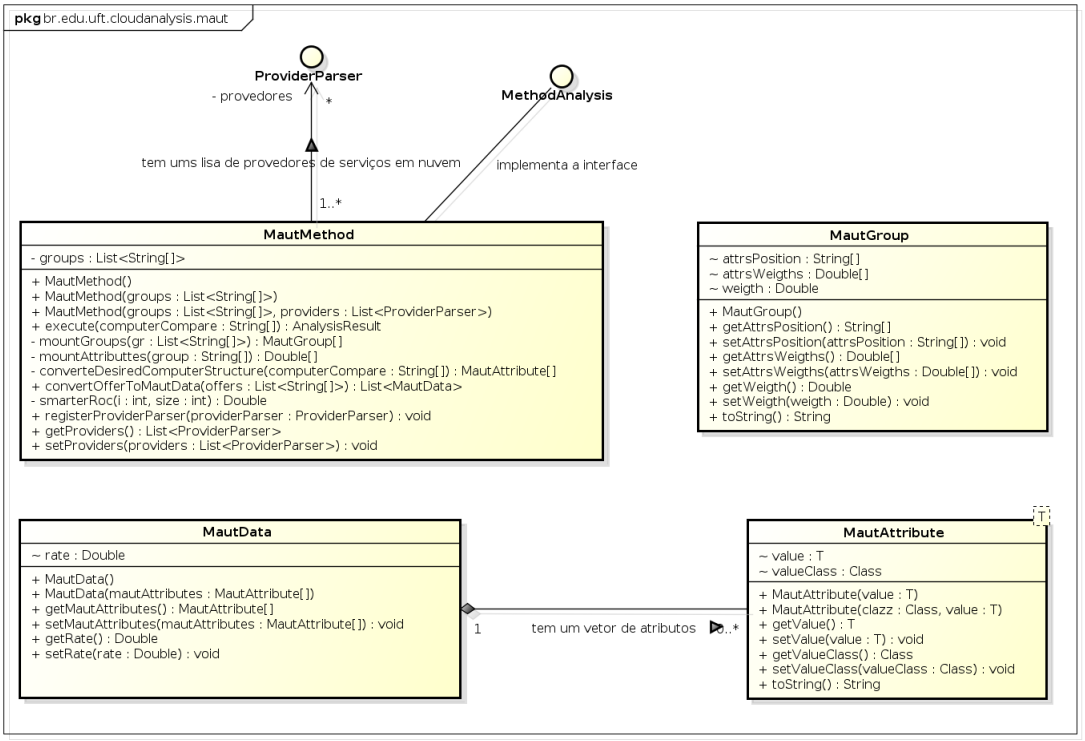

Figura 3.8: Diagramas de classes do algoritmo MAUT.

3.4.5 Definição do ReproeScience-model

ReproProvenance-model foi desenvolvido como uma biblioteca para armazenar dados da Base de Proveniência do ReproeScience. As classes contidas nesta biblioteca são mapeadas usando a especificação *Java Persistence API* (JPA), para que posteriormente possa

ser utilizadas por *frameworks* de mapeamento objeto/relacional(ORM) que implementam este padrão. JPA é uma especificação padrão da linguagem Java para persistência de dados que define as regras ORM via anotações definindo um Modelo Entidade- Relacionamento (MER). Esta seção tem como objetivo descrever as classes implementadas no ReproProvenance-model, a fim de auxiliar na compreensão dos relacionamentos entre as entidades, a figura 3.9 exibe o diagrama de classes desta biblioteca.

A classe Workflow é responsável por armazenar informações de um *workflow* científico como, por exemplo, título, descrição, diretório, a hora de início e de término da execução. Um Workflow tem uma relação de zero ou muitos para um ScientificPaper que contém informações de artigos científicos, sendo que um artigo pode ser gerado por um determinando Workflow. Da mesma forma, existe uma relação para si mesmo na classe Workflow, neste caso definindo uma derivação entre *workflows* científicos.

As atividades que compõem um *workflow* científico são representadas pela classe Activity, esta que armazena os dados necessários da execução como os comandos, parâmetros, hora de inicio e de término do processo, além disso, as informações de entradas e saídas produzidas são definidas pela classe Artifacts por um relacionamento entre estas classes. Uma atividade tem um relacionamento com a classe ComputerSystem, sendo responsável por guardar as informações do ambiente computacional em que a atividade foi executada , este que por sua vez usa um sistema operacional definido pela classe OperationSystem. Os programas usados na execução da atividade são salvos na classe Software, por outro lado as dependências destes programas são armazenadas na classe Library.

A classe Author representa um cientista ou desenvolvedor, um autor tem um ou vários *workflows* definidos e podendo ter várias contas de diferentes provedores de serviços em nuvens, estas contas são salvas na Classe Conta, que por sua vez tem recursos em nuvens alocados representados pela classe CloudResource. Um recurso tem valores e uma localização representados respectivamente pelas Classes Preço, Localization e Country.

Já a classe CloudStorageSystem defini a capacidade de armazenamento em disco de uma imagem de maquina virtual, esta determinada pela classe VirtualMachineImage. Estas máquinas virtuais são baseadas por um ComputerSystem através de um relacionamento e definida como base para um tipo de máquina virtual por um VirtualMachineType. Por fim um máquina virtual é uma instância de um recurso na nuvem representado por um VirtualMahineCloud.

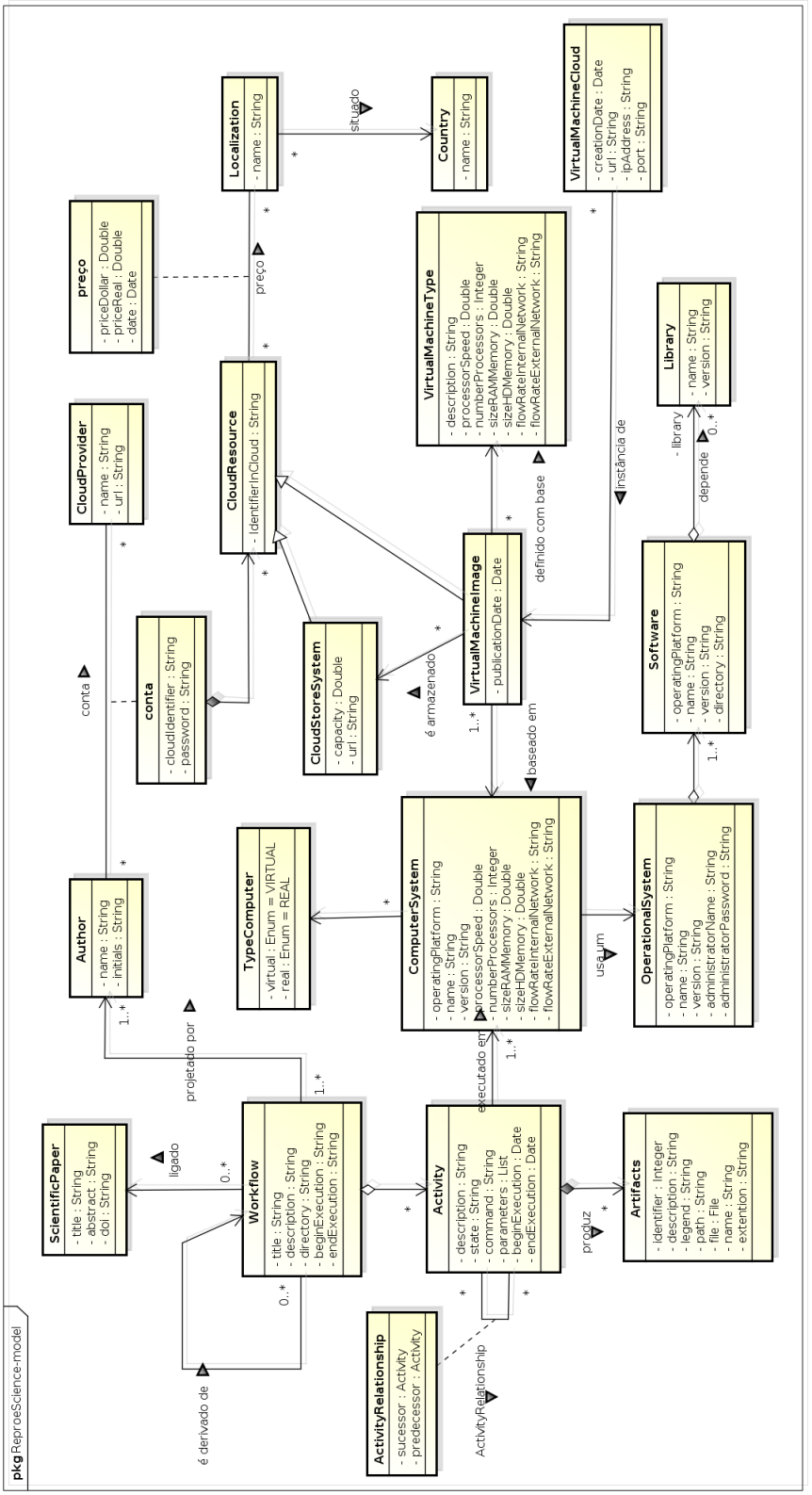

Figura 3.9: Diagrama de classes de proveniência do ReproeScience-web.

3.4.6 Desenvolvimento do ReproProvenace

O módulo ReproProvenance é responsável por coletar e armazenar os dados de proveniência no banco de dados do ReproeScience durante a execução do *workflow*, sendo desenvolvido como módulo para ser utilizado no gerenciador de *workflows* científico Vistrails.

Inicialmente foi realizado um levantamento das informações nas quais seriam coletadas pelo ReproProvenance, gerando então a modelagem do ReproeScience-model visto anteriormente na seção 3.4.5, entretanto, surgiu a necessidade de gerar um XML baseado no modelo de dados de proveniência OPM (*Open Model Provenance*), para posteriormente ser utilizado na geração da visualização da proveniência em forma de grafos no ReproeScience-web.

O ReproProvenance salva os dados do início e o fim da execução durante o processo de execução do *workflow* científico e de seus processos, tais como nome do processo, id, informações de entrada e saída, além disto, salva as características do ambiente computacional onde o *workflow* foi executado, sendo estas informações necessárias para que o ReproCloudAnalysis possa realizar a análise e classificação das melhores ofertas do provedores em nuvem.

No Vistrails cada processo tem entradas e saídas de acordo com o tipo do módulo usado pelo processo, temos como exemplo o módulo *python source* disponibilizado pelo Vistrails que oferece a opção de programar a função do processo em Python, este módulo tem como entrada o código fonte dentre outras entradas que podem ser adicionadas pelo cientista. O ReproProvenance após a captura da proveniência inicia inserindo no banco de dados as atividades que são gerados a partir das informações de cada processo, os artefatos são gerados das entradas, saídas e do código fonte quando há nos processos.

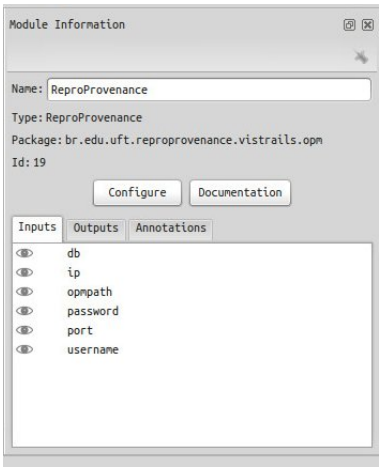

Figura 3.10: Propriedades do módulo ReproProvenance.

A conexão com o banco de dados utiliza entradas obrigatórias que devem ser inseridas pelo cientista e são o *host* de conexão, nome do banco de dados, usuário com privilégios de inserção, senha do usuário, número da porta e o local juntamente com o nome do arquivo do XML. As entradas são de extrema importância para o sucesso da captura do ambiente computacional juntamente com os dados de proveniência. A figura 3.10 mostra uma tela com as entradas obrigatórias do módulo ReproProvenance.

Para a utilização do módulo o cientista precisa realizar chamadas de duas funções para informar ao ReproProvenance o início e o término da execução de cada processo que compõe o *workflow* científico, sendo possível escolher qual o processo há a necessidade de coletar as informações. Para a coleta dos dados do *workflow* não é necessário chamar nenhum método, pois, o módulo identifica e realiza automaticamente a captura das informações necessárias, a listagem 3.1 mostra um exemplo de chamada das funções.

Listing 3.1: Exemplo de utilização do ReproProvenance

```
1 reproprovenance = self.registry.get_module_by_name("br.edu.uft.reproprovenance.vistrails.opm", "
      ReproProvenance")
2 reproprovenance.saveModuleProvenance(self)
3 reproprovenance.informEndModule(self)
```

Ao final o módulo gera o XML com os dados capturados tanto pelo Vistrails quanto pelo ReproProvenance, seguindo a estrutura do OPM. Além disso, é inserido o id e o nome que identifica o *workflow* inserido no banco de dados, essas informações serão úteis para identificar posteriormente a proveniência e o *workflow* pertencente ao XML durante a geração do grafo para análise da proveniência.

Listing 3.2: Exemplo do arquivo gerado pelo ReproProvenance

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <opmGraph version="1.0.3" workflowid="35" workflowname="SciEvol.vt" xmlns="http://openprovenance.org/model/
      v1.01.a">
3    <artifact id="a341">
4       <value>
5          <computer moduleId="10">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
6       </value>
7    </artifact>
8 </opmGraph>
```

Para o desenvolvimento do módulo foram utilizadas as ferramentas contidas na tabela 3.2, abaixo:

Python é a linguagem de programação no qual o Vistrails foi desenvolvida, por isso obrigatoriamente os módulos escritos para serem executados na ferramenta devem utilizar esta linguagem, desta forma a implementação do módulo foi escrita em Python apoiadas

Ferramenta	Aplicação
Python	Linguagem de Programação
NetBeans	Ambiente de Desenvolvimento
Maven	Automação de Compilação
Psycopg2	Persistência com o banco de dados
PsUtil	Biblioteca para captura do ambiente computacional
PostgreSQL	Banco de Dados

Tabela 3.2: Ferramentas utilizadas no desenvolvimento do ReproProvenance.

pelas bibliotecas Psycopg2 utilizada para conectar ao banco de dados em PostgreSQL e PsUtil para capturar as características do ambiente computacional.

3.4.7 Desenvolvimento do ReproeScience-web

ReproeScience-web tem como objetivo oferecer uma interface simples e usual utilizando tecnologias html5 e jQuery, a fim de tornar o uso do ReproeScience mais fácil aumentando o interesse por novos usuário. Além disso, sistemas web oferecem uma maior iteração com os componentes melhorando a experiência com os módulos desenvolvidos neste projeto e tendo como requisitos para os usuários apenas um navegador web.

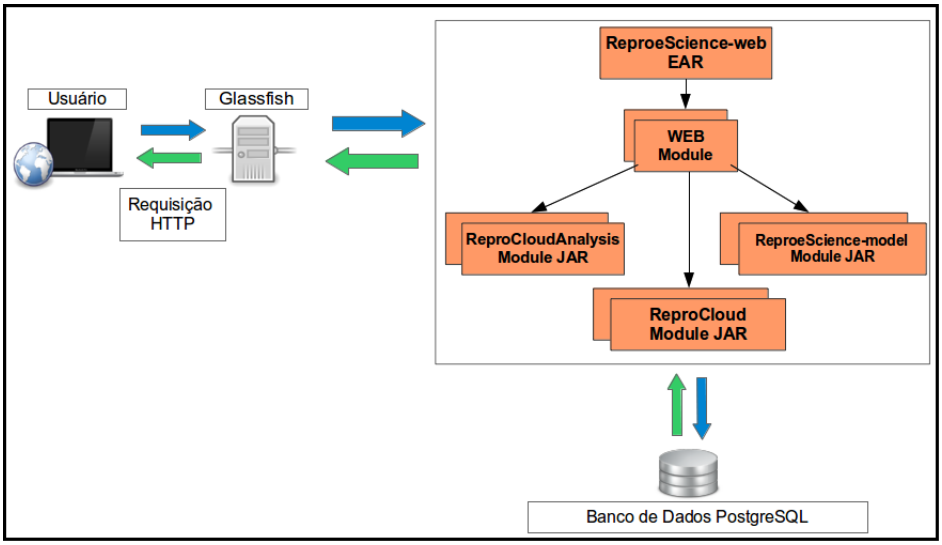

Figura 3.11: Diagrama da estrutura do ReproeScience-web.

Como o ReproCloudAnalysis, ReproeScience-model e o ReproCloud[] foram desenvolvidos usando o conceito de modularidade a integração destes módulos a um sistema web se tornou possível usando a tecnologia JavaEE, deste modo foi criado uma aplicação no formato Enterprise Archive(EAR) acoplando estes *middlewares*, tornando possível adicionar outros tipos de aplicações, como por exemplo o desenvolvimento de um sistema para

dispositivos móveis como o Android [Developers, 2011]. A figura 3.11 mostra o diagrama da aplicação EAR.

O sistema é executado no servidor JavaEE GlassFish 3.2, como o ReproeScience-model foi mapeado usando a API JPA, foi utilizada a ferramenta ORM Hibernate, além disso para cuidar da segurança do sistema foi utilizado o *framework* Apache Shiro. As ferramentas que contribuíram para o desenvolvimento deste módulo podem ser vista na tabela abaixo:

Ferramenta	**Aplicação**
Java	Linguagem de Programação
NetBeans	Ambiente de Desenvolvimento
Maven	Automação de Compilação
JSF	*Framework MVC*
GlassFish	Servidor de aplicação JavaEE
PostgreSQL	Banco de Dados
ApacheShiro	*Framework* de segurança
Prettyfaces	*Framework* para url amigáveis
PrimeFaces	Biblioteca com componentes gráficos
JsPlumb	Biblioteca para geração de grafos
Google Charts	Bibliteca para geração de gráficos
Gson	Conversor de objetos JSON para objetos Java

Tabela 3.3: Ferramentas utilizadas no desenvolvimento do ReproeScience-web.

Ao todo seis telas foram desenvolvidas: (i) Tela de autenticação, (ii) tela para gerenciar as contas de provedores de serviços em nuvem, (iii) tala para analisar dados de proveniência em formato OPM, (iv) Tela para realizar análises de classificação das ofertas, (v) tela para realizar a implantação de *workflows* na nuvem e (vi) tala para reprodução dos experimentos científicos implantados na nuvem.

Capítulo 4

Resultados

Neste capítulo é apresentado uma execução de um experimento usando os métodos K-means e MAUT. Foi realizado uma análise comparativa dos resultados obtidos, para identificar qual método retornaria os melhores resultados.

Primeiramente foi realizado a execução do *Workflow* SciEvol para retornar as configurações de hardware da máquina, essas configurações eram submetidas na execução dos métodos. Os casos de teste foram executados em uma máquina, com sistema operacional Linux, 4 gb de Memória RAM, 200 gb de Disco Rígido, Arquitetura de 64 bits com conexão de 10,0 Mpbs.

4.1 Execução do SciEvol

Para a aquisição dos dados para executar os experimentos nos métodos k-means e MAUT, foi necessário a execução do *Workflow* SciEvol. Esse teste tem como objetivo pegar a configuração do hardware através do modulo ReproProvenance. O resultado será um XML no formato OPM contendo essas informações. Posteriormente essa configuração será usada como entrada para a execução dos métodos.

A Figura 4.1 exibe os parâmetros de configuração do ReproProvenance adicionado no SciEvol. O teste foi finalizado em um tempo 16 minutos e 23 segundos, onde foi obtido a seguinte saída:

Listing 4.1: Resultado gerado pelo ReproProvenance

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <opmGraph version="1.0.3" workflowid="35" workflowname="SciEvol.vt" xmlns="http://openprovenance.org/model/
      v1.01.a">
3    <artifact id="a37">
4       <value>
5           <computer moduleId="19">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
```

```
6         </value>
7     </artifact>
8     <artifact id="a38">
9         <value>
10            <computer moduleId="20">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
11        </value>
12    </artifact>
13    <artifact id="a39">
14        <value>
15            <computer moduleId="21">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
16        </value>
17    </artifact>
18    <artifact id="a40">
19        <value>
20            <computer moduleId="22">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
21        </value>
22    </artifact>
23    <artifact id="a41">
24        <value>
25            <computer moduleId="23">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
26        </value>
27    </artifact>
28    <artifact id="a42">
29        <value>
30            <computer moduleId="24">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
31        </value>
32    </artifact>
33    <artifact id="a43">
34        <value>
35            <computer moduleId="25">["3.75", "133.85", "4", "1", "10", "10", "x86_64", "Linux"]</computer>
36        </value>
37    </artifact>
38</opmGraph>
```

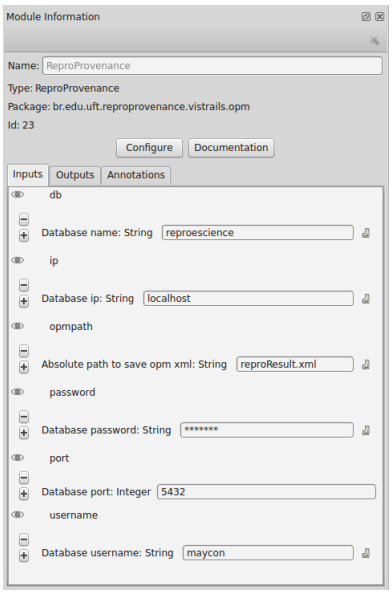

Figura 4.1: Configurações do ReproProvenance.

4.2 Teste de execução do K-means

Após a captura das informações pelo ReproProvenance foi realizado o teste do K-means. A Figura 4.3 mostra os dados de entrada usado no teste. O método realizou uma busca nos três provedores de nuvem: Amazon, Gogrid e Rackspace, selecionando as ofertas que mais se aproximavam aos dados de entrada.

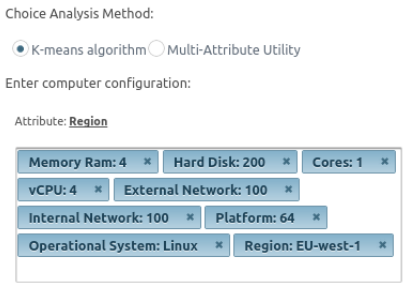

Figura 4.2: Configuração de entrada para o método K-means.

O resultado do teste pode ser visto na Figura 4.3 e na Tabela 4.1. Onde o texto no centro de cada bolinha exibida no gráfico significa o id das ofertas.

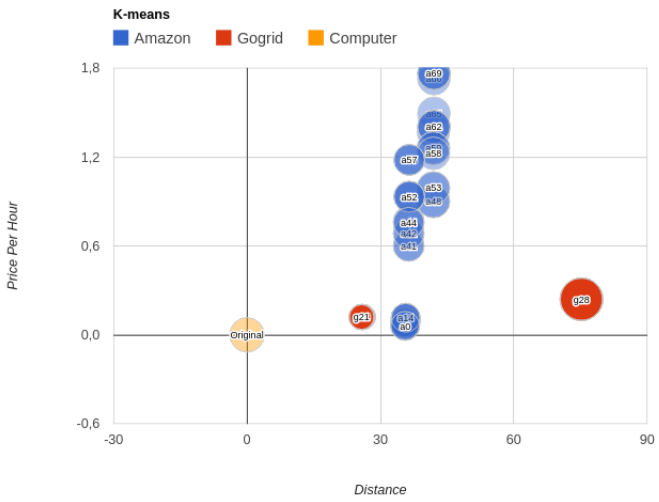

Figura 4.3: Gráfico com o resultado do método K-means.

4.3 Teste de execução do Maut

Para o teste do método MAUT foi preciso criar grupos para determinar quais os atributos tem mais pesos, pois esta configuração é de suma importância para o resultado final. Deste modo foi criado três grupos, onde o primeiro grupo tem com maior peso o atributo de

memória seguido dos atributos de vCpu, cores, disco rígido. A Figura 4.4 abaixo mostra os parâmetros preenchidos no aplicativo Reproescience-web.

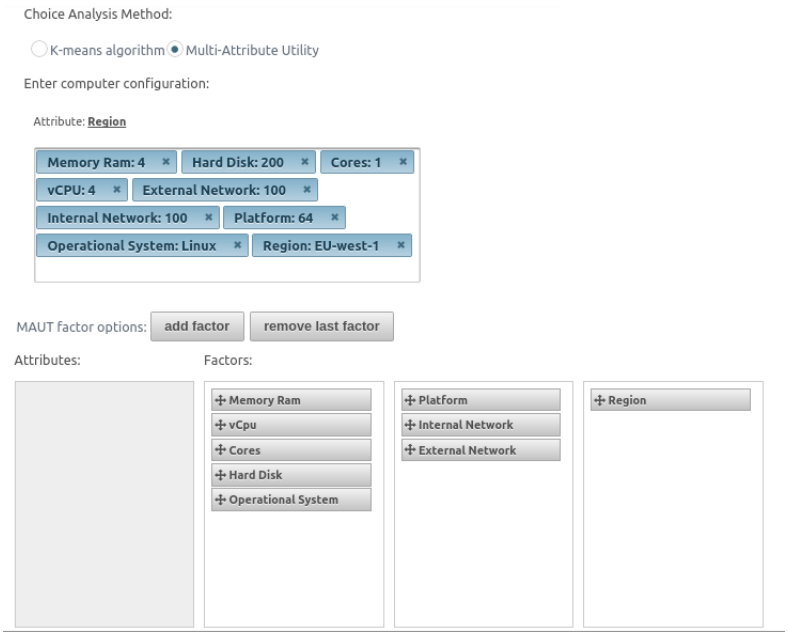

Figura 4.4: Configurações do ReproProvenance.

A Figura 4.5 mostrá em forma de gráfico o resultado do teste do método MAUT. Enquanto a Tabela 4.2 mostra o resultado do teste em forma de tabela.

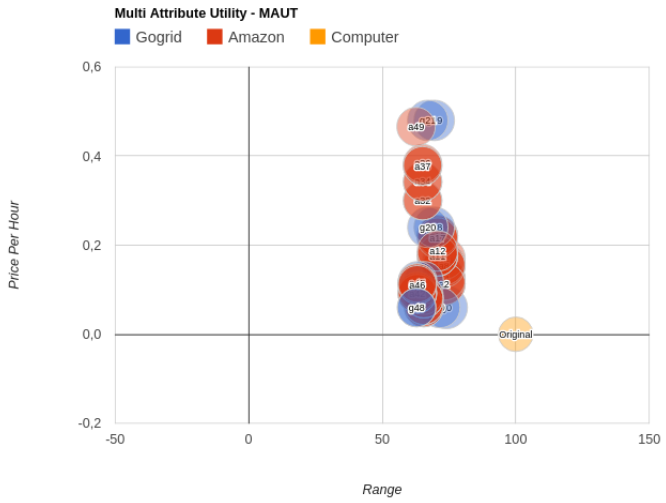

Figura 4.5: Configurações do ReproProvenance.

4.4 Análise dos Resultados Experimentais

Os métodos demostraram eficiência na classificação das melhores ofertas, como pode ser visto tanto nas figuras e tabelas elas retornaram as ofertas que mais se aproximavam as

configurações desejadas. Porém, o K-means se demonstrou mais rápido para iniciar o processo de classificação por não ser necessário nenhum tipo de configuração.

Por outro lado os resultados podem não ser tão exatos como o método MAUT, pois este por sua vez é necessário a realização de algumas configurações para iniciar o processo de classificação. Entretanto o MAUT retorna resultados mais precisos de acordo com os atributos mais importantes.

Distância	Tipo	Memória RAM	Disco Rígido	Cores	vCPU	Rede Interna	Rede Externa	Plat-form	SO	Preço	Região	Provedor
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	us-east	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	us-east	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	us-west-2	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	us-west-2	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	eu-ireland	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	eu-ireland	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	apac-sin	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	apac-sin	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	apac-syd	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	apac-syd	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.02	us-east	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.02	us-east	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.02	us-west-2	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.02	us-west-2	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.02	apac-sin	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.02	apac-sin	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.02	apac-syd	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.02	apac-syd	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.025	us-west	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.025	us-west	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.027	apac-tokyo	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.027	apac-tokyo	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.027	sa-east-1	Amazon
20.20	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.027	sa-east-1	Amazon
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Linux	0.03	US-west-1	Gogrid
25.16	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Linux	0.03	US-west-1	Gogrid
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Linux	0.03	US-east-1	Gogrid
25.16	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Linux	0.03	US-east-1	Gogrid
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Linux	0.03	EU-west-1	Gogrid
25.16	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Linux	0.03	EU-west-1	Gogrid
25.52	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Windows	0.03	US-west-1	Gogrid
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Windows	0.03	US-west-1	Gogrid
25.52	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Windows	0.03	US-east-1	Gogrid
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Windows	0.03	US-east-1	Gogrid
25.52	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	32	Windows	0.03	EU-west-1	Gogrid
25.34	X-Small	0.5	25.0	1.0	0.5	100.0	100.0	64	Windows	0.03	EU-west-1	Gogrid
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.035	us-west	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.035	us-west	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.035	eu-ireland	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.035	eu-ireland	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.035	apac-tokyo	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.035	apac-tokyo	Amazon
20.64	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Windows	0.037	sa-east-1	Amazon
20.42	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Windows	0.037	sa-east-1	Amazon
4.123	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	US-west-1	Gogrid
2.828	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	US-west-1	Gogrid
4.123	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	US-east-1	Gogrid
2.828	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	US-east-1	Gogrid
4.123	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	EU-west-1	Gogrid
2.828	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	EU-west-1	Gogrid

Tabela 4.1: Resultados do teste K-means.

Rank	Tipo	Memória RAM	Disco Rígido	Cores	vCPU	Rede Interna	Rede Externa	Plat-form	SO	Preço	Região	Provedor
97.03	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	US-east-1	Gogrid
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	Dallas Texas	Rackspace
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	Chicargo Illinois	Rackspace
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	Reston Virginia	Rackspace
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	London England	Rackspace
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	Hong Kong	Rackspace
94.55	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Linux	0.06	Sydney Australia	Rackspace
94.25	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	US-west-1	Gogrid
94.25	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Linux	0.06	EU-west-1	Gogrid
90.086	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	US-east-1	Gogrid
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	Dallas Texas	Rackspace
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	Chicargo Illinois	Rackspace
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	Reston Virginia	Rackspace
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	London England	Rackspace
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	Hong Kong	Rackspace
87.61	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	32	Linux	0.06	Sydney Australia	Rackspace
87.30	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	US-west-1	Gogrid
87.30	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Linux	0.06	EU-west-1	Gogrid
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	us-east	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	us-west-2	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.025	us-west	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	eu-ireland	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	apac-sin	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.027	apac-tokyo	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.02	apac-syd	Amazon
87.27	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	64	Linux	0.027	sa-east-1	Amazon
81.75	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Windows	0.06	US-east-1	Gogrid
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	us-east	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	us-west-2	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.025	us-west	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	eu-ireland	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	apac-sin	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.027	apac-tokyo	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.02	apac-syd	Amazon
80.33	t1.micro	0.615	30.0	1.0	1.0	100.0	100.0	32	Linux	0.027	sa-east-1	Amazon
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	Dallas Texas	Rackspace
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	Chicargo Illinois	Rackspace
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	Reston Virginia	Rackspace
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	London England	Rackspace
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	Hong Kong	Rackspace
79.27	RAM: 1GB	1.0	40.0	1.0	1.0	60.0	60.0	64	Windows	0.08	Sydney Australia	Rackspace
78.97	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Windows	0.06	US-west-1	Gogrid
78.97	Small	1.0	50.0	1.0	1.0	100.0	100.0	64	Windows	0.06	EU-west-1	Gogrid
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	Dallas Texas	Rackspace
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	Chicargo Illinois	Rackspace
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	Reston Virginia	Rackspace
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	London England	Rackspace
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	Hong Kong	Rackspace
78.43	RAM: 512MB	0.512	20.0	1.0	1.0	40.0	40.0	64	Linux	0.022	Sydney Australia	Rackspace
74.80	Small	1.0	50.0	1.0	1.0	100.0	100.0	32	Windows	0.06	US-east-1	Gogrid

Tabela 4.2: Resultados do teste MAUT.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foi estudado e analisado métodos de classificação, para análise de instâncias ofertadas pelos provedores de serviços em nuvem. Puderam-se ainda apresentar, as dificuldades que foram encontradas para obter a base de dados de ofertas disponibilizadas pelos provedores de serviços de nuvem, devido as constantes alterações de seus sites. A utilização de serviços em nuvem vem aumentando gradativamente, tornando uma grande ferramenta para cientistas executarem workflows científicos, além disso, seus preços se tornam cada vez mais atrativos. Através dos resultados obtidos o ReproAnalysis oferece um recurso para analisar automaticamente as ofertas tornando viável, pois diminui o trabalho da análise humana nos sites.

As ferramentas como foi visto, retorna uma lista de recursos através da estrutura desejada classificado pelos preços, filtrando assim entre os provedores as ofertas mais interessantes. Ainda assim a ferramenta pode ser otimizada para tornar essa lista melhor, através de testes mais precisos usando as técnicas de benchmarks para medir os desempenhos das instâncias ofertadas.

5.1 Trabalhos Futuros

- Criar uma versão mobile do ReproCloudAnalysis.
- Disponibilizar outras formas de visualização dos resultados na web.
- Melhorar os parâmetros para as análises através de benchmarks.

Referências

[pro, a] Constraints of the prov data model. http://www.w3.org/TR/prov-constraints/.

[pro, b] Constraints of the prov data model. http://www.w3.org/TR/prov-constraints/.

[fas,] Fasta format is a text-based format for representing either nucleotide sequences or peptide sequences. http://zhanglab.ccmb.med.umich.edu/FASTA/.

[maf,] Multiple alignment program for amino acid or nucleotide sequences. http://mafft.cbrc.jp/alignment/server/.

[cod,] Phylogenetic analysis by maximum likelihood. http://abacus.gene.ucl.ac.uk/software/paml.html.

[pro, c] Prov-aq: Provenance access and query. http://www.w3.org/TR/prov-aq/.

[pro, d] Prov model primer. http://www.w3.org/TR/prov-primer/.

[pro, e] Prov-n: The provenance notation. http://www.w3.org/TR/2013/REC-prov-n-20130430/.

[pro, f] Prov-o: The prov ontology. http://www.w3.org/TR/prov-o/.

[opm,] Provenance challenge wiki. http://twiki.ipaw.info/bin/view/Challenge/.

[rax,] Randomized axelerated maximum likelihood. http://www.exelixis-lab.org/.

[rea,] Sequence format conversion. http://www.ebi.ac.uk/Tools/sfc/readseq/.

[Almeida, 2011] Almeida, A. d. (2011). O conhecimento eo uso de métodos multicritério de apoio a decisão. *Recife: Editora Universitária UFPE.[Links]*.

[Altintas et al., 2006] Altintas, I., Barney, O., and Jaeger-Frank, E. (2006). Provenance collection support in the kepler scientific workflow system. In *Provenance and annotation of data*, pages 118–132. Springer.

[Cloud, 2013] Cloud, A. E. C. (2013). Disponível em http://aws. amazon. com/.

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

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

[Developers, 2011] Developers, A. (2011). What is android. *ht tp://developer. android. com/guide/basics/what-is-android. html*, 2.

[di Carlantonio, 2001] di Carlantonio, L. M. (2001). *Novas metodologias para clusterizaçao de dados*. PhD thesis, UNIVERSIDADE FEDERAL DO RIO DE JANEIRO.

[Freire et al., 2008] Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21.

[GoGrid, 2013] GoGrid, L. (2013). Cloud hosting, cloud servers, hybrid hosting, cloud infrastructure from gogrid.

[Gomes et al., 2009] Gomes, L. F. A. M., Gomes, C. F. S., and de Almeida, A. T. (2009). *Tomada de decisão gerencial: enfoque multicritério*. Atlas.

[Hayes, 2008] Hayes, B. (2008). Cloud computing. *Communications of the ACM*.

[Ihrig, 2013] Ihrig, C. J. (2013). Javascript object notation. In *Pro Node. js for Developers*, pages 263–270. Springer.

[Introducing,] Introducing, R. Representational state transfer (rest).

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666.

[Keeney,] Keeney, R. en h. raiffa, 1976, decisions with multiple objectives: Preferences and value tradeoffs.

[Klinginsmith et al., 2011] Klinginsmith, J., Mahoui, M., and Wu, Y. M. (2011). Towards reproducible escience in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 582–586. IEEE.

[Lenk et al., 2009] Lenk, A., Klems, M., Nimis, J., Tai, S., and Sandholm, T. (2009). 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*, pages 23–31. IEEE Computer Society.

[Mattoso et al., 2010] Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E., Oliveira, D., Cruz, S., Martinho, W., and Murta, L. (2010). Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management*, 5(1):79–92.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The nist definition of cloud computing (draft). *NIST special publication*, 800(145):7.

[Missier et al., 2013] Missier, P., Belhajjame, K., and Cheney, J. (2013). The w3c prov family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 773–776. ACM.

[Moreau, 2011] Moreau, L. (2011). Provenance-based reproducibility in the semantic web. *Web semantics: science, services and agents on the world wide web*, 9(2):202–221.

[Moreau et al., 2011] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., et al. (2011). The open provenance model core specification (v1. 1). *Future Generation Computer Systems*, 27(6):743–756.

[Moreau et al., 2008] Moreau, L., Ludäscher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., Chin, G., Clifford, B., Cohen, S., Cohen-Boulakia, S., et al. (2008). Special issue: The first provenance challenge. *Concurrency and computation: practice and experience*, 20(5):409–418.

[Ocaña et al., 2012] Ocaña, K. A., de Oliveira, D., Horta, F., Dias, J., Ogasawara, E., and Mattoso, M. (2012). Exploring molecular evolution reconstruction using a parallel cloud based scientific workflow. *Advances in Bioinformatics and Computational Biology*, page 179.

[Olson, 2009] Olson, J. A. (2009). Data as a service: Are we in the clouds? *Journal of Map & Geography Libraries*, 6(1):76–78.

[PLOTREE and PLOTGRAM, 1989] PLOTREE, D. and PLOTGRAM, D. (1989). Phylip-phylogeny inference package (version 3.2).

[Rackspace, 2013] Rackspace, U. (2013). The rackspace cloud inc. disponível em http://www.rackspace.com/.

[Raiffa, 1970] Raiffa, H. (1970). *Decision Analysis: Introductory Lectures on Choices Under Uncertainty:(2. Printing)*. Addison-Wesley.

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