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

Especificação e Implantação de um ambiente de gerenciamento de Workflows Científicos baseado em Nuvens Computacionais Federadas

Jefferson Leandro da Silva

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientador
Prof.ª Dr.ª Genaína Nunes Rodrigues

Brasília
2013

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

Coordenador: Prof. Dr. Flávio de Barros Vidal

Banca examinadora composta por:

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientador) — CIC/UnB
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB

CIP — Catalogação Internacional na Publicação

da Silva, Jefferson Leandro.

Especificação e Implantação de um ambiente de gerenciamento de Workflows Científicos baseado em Nuvens Computacionais Federadas / Jefferson Leandro da Silva. Brasília : UnB, 2013.
133 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. workflow, 2. cloud computing, 3. federação, 4. bioinformatica,
5. Bioside, 6. Bionimbus

CDU 004.4

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

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

Especificação e Implantação de um ambiente de gerenciamento de Workflows Científicos baseado em Nuvens Computacionais Federadas

Jefferson Leandro da Silva

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientador)
CIC/UnB

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

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

Prof. Dr. Flávio de Barros Vidal
Coordenador do Curso de Computação — Licenciatura

Brasília, 07 de março de 2013

Dedicatória

Dedico este trabalho à construção de um mundo onde a igualdade e a cooperação agem como princípio norteadores de nossas maiores e mais importantes decisões.

Agradecimentos

Agradeço à minha família por todo o apoio e compreensão, sem vocês nada disso seria possível.

Agradeço aos meus amigos por toda a ajuda nos momentos difíceis, e momentos de descontração.

Agradeço aos meus professores, os bons e os ruins, pelas lições, boas e ruins que me ajudaram tanto na vida acadêmica, quanto pessoal.

À minha orientadora, Genaína Rodrigues, por toda a dedicação, orientação e iluminação.

E claro, aos meus inimigos, por me ajudarem a enxergar meus pontos fracos, ajudando-me a moldar o meu caminho para as minhas conquistas.

Resumo

A bioinformática é um ramo interdisciplinar que consiste na intersecção de conhecimentos dos domínios da Genética e Ciência da Computação. Este campo de pesquisa sofreu um aumento de popularidade nos últimos anos e tornou-se um campo no qual processos e fluxos de trabalho bem definidos podem ser facilmente abstraídos. Deste modo, este trabalho destina-se a especificar e implementar um ambiente de composição e execução de workflows científicos baseado em federação de nuvens computacionais.

Palavras-chave: workflow, cloud computing, federação, bioinformatica, Bioside, Bionimbus

Abstract

The Bioinformatics is an interdisciplinary field which leverages upon an intersection between Genetics and Computer Science. This research field has increased its popularity throughout the years, and became a domain where a plenty of well-defined scientific processes and workflows may be easily abstracted away. In these terms, this work aims to specify and implement the composition and execution of a scientific workflows environment based on Cloud federations.

Keywords: workflow, cloud computing, federation, bioinformatics, bioside, bionimbus

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Ao longo dos anos, a pesquisa científica em geral tem se fundado em vários processos bem definidos que podem ser apoiados e otimizados por recursos computacionais. Entretanto, o avanço da tecnologia computacional disponível trouxe consigo a complexidade do conhecimento necessário para se extrair todos os benefícios de tais recursos. Com isso, a necessidade de abstrair tais ferramentas de maneira a tornar os processos de trabalho da comunidade científica totalmente independentes da complexidade da configuração do ambiente computacional torna-se algo extremamente necessário, uma vez que o gasto de tempo pode tonar-se bastante custoso, seja no orçamento geral da pesquisa, ou mesmo, em termos de tempo disponível. Em resposta a tais desafios, o conceito de *workflows* emergiu a partir de estudos sobre processos organizacionais. O trabalho realizado em [48] define o conceito de *workflow* da seguinte maneira:

> É a automação computadorizada de um processo de negócio, parcialmente ou sua sua totalidade.

Desde então, a necessidade de automação de processos tornou-se uma realidade em vários domínios do conhecimento humano, dentre eles, a pesquisa científica. Baseando-se na crescente popularidade dos sistemas gerenciadadores de *workflows*, o trabalho realizado em [48] define o conceito de *workflow* da seguinte maneira:

> É um sistema com capacidades de definição, gerenciamento e execução de workflow agregadas ao seu corpo de funcionalidades através da execução de softwares nos quais a ordem de execução é determinada por uma representação computacional da lógica do fluxo de trabalho.

Alguns anos depois, a aplicação dos *workflows* extendeu-se ao domínio da pesquisa científica, e devido as peculiaridades inerentes à este domínio, os *workflows* científicos tornaram-se um campo de pesquisa com definição própria. Os *workflows* científicos, caracterizam-se por uma série de atributos que os distinguem dos *workflows* de outros domínios. Entre eles, destacam-se: A necessidade por processamento de alto desempenho

e armazenamento massivo, a escalabilidade, a comunicação orientada a dados entre os componentes e a reprodutibilidade. Portanto, a aplicação de tecnologias de computação de alto desempenho à este domínio tornou-se inevitável.

Desde os primórdios das redes de computadores, o seu uso intenso em pesquisas científicas foi concebido como ferramenta de grande valia [47]. As nuvens computacionais (do inglês, *Cloud Computing*) são o estado-da-arte da computação distribuída [39]. No entanto, as nuvens computacionais não são um novo tipo de infraestrutura de computação, mas sim um novo modelo de oferta de serviços [39]. O usuário final comum, ou até mesmo outras organizações, utilizam o serviço das nuvens computacionais de acordo com a demanda de suas aplicações, liberando-os da aquisição de infraestruturas físicas de computação, e ao mesmo tempo, oferecendo-os a ilusão de infinitude dos recursos de armazenamento e processamento.

Devido ao potencial de oferta sob demanda de recursos, as nuvens computacionais tornaram-se uma fonte de recursos para aplicações científicas, inclusive, os *workflows* científicos. A execução dos *workflows* científicos em ambientes de nuvens computacionais torna evidente vários desafios, devido à questões pertinentes a este tipo de ambiente tais como: a Segurança, a Escalabilidade, a Eficiência, a Confiabilidade, etc [52].

No presente trabalho, foram tomados processos inerentes às atividades de pesquisa em Bioinformática, como principal estudo de caso. Embora este trabalho não possua o desenvolvimento da Bioinformática como objetivo principal, pode-se utilizar os resultados obtidos bem como as ferramentas desenvolvidas como instrumentos de auxílio aos processos que poderão surgir através de experimentos realizados futuramente. Assim, a automatização de fluxos de trabalho extraidos de processos específicos do domínio da pesquisa científica será o objetivo central deste trabalho.

1.2 Objetivo

O presente trabalho possui como principal objetivo, a especificação e desenvolvimento de um ambiente de definição e execução de *workflows* científicos em estruturas de nuvens computacionais. O ambiente deverá disponibilizar estas funcionalidades de forma que usuários sem conhecimento técnico sobre tais infraestruturas sejam capazes de executar *workflows* científicos, dispensando a necessidade de conhecimentos avançados em informática, liberando, assim, recursos para serem dispendidos apenas com o domínio de aplicação do sistema.

1.3 Estrutura do trabalho

O documento consiste de 5 capítulos, os quais são descritos a seguir:

O Capítulo 2 consiste na definição de conceitos fundamentais necessários ao entendimento deste trabalho. Serão abordados os seguintes temas:

Capítulo 2

Conceitos Fundamentais

Neste capítulo, serão expostos alguns conceitos fundamentais para o entendimento deste trabalho. Na seção 2.1 serão expostos conceitos sobre *workflows*, e na seção 2.2 seguinte, serão definidos os *workflows* científicos. Uma breve explicação sobre Nuvens Computacionais (*Cloud Computing*) e Nuvens Federadas será exposta na seção 2.3. E por fim, conceitos básicos sobre biologia molecular serão expostos na seção 2.5, com exemplos de *workflows* reais explicitados na seção 2.6.

2.1 Workflows

2.1.1 Conceitos Básicos

À medida em que os processos organizacionais, em geral, tornaram-se um objeto de estudo com escopo e teoria própria, a representação deste conceito em sistemas computacionais tornou-se algo emergente [48]. E deste então, a modelagem de fluxos de trabalho (do inglês, *workflows*) tonou-se um ramo da Modelagem de Processos Organizacionais, como parte da Re-egenharia de Processos Organizacionais [48]. Os *workflows* são um conjunto de tarefas inerente a algum processo pertencente a algum domínio específico onde os dados de saída de uma tarefa servem como entrada à tarefa seguinte, sendo que as tarefas podem ou não depender da interação com o usuário final para o términio de sua execução. O grupo *workflow management coalition*, em sua primeira especificação documentada sobre o conceito [48] estabeleceu a definição da seguinte maneira:

> A automação computatorizada de um processo de negócio, parcialmente ou em sua totalidade.

Os Sistemas de Gerenciamento de Workflows (do inglês *Workflow Management Systems*, ou ainda, *WfMS*) conceituam-se como sistemas de fluxos de trabalho apoiados por sistemas de computador, como é explicitado por *Hollingsworth et al.* [48] na definição a seguir:

É um sistema com capacidades de definição, gerenciamento e execução de *workflow* agregadas ao seu corpo de funcionalidades por meio da execução de softwares nos quais a ordem de execução é determinada por uma representação computacional da lógica do fluxo de trabalho.

2.1.2 Modelo de Referência

A *Workflow Management Coalition* (doravante WfMC) [25], fundada no ano de 1993, é uma organização internacional formada por empresas privadas, consultores autônomos, universidades e grupos de pesquisa, para fins de padronização da comunicação entre os sistemas gerenciadores de *workflow*. O objetivo principal é garantir maior capacidade de comunicação entre as aplicações e os sistemas de *workflow* existentes bem como uma maior interoperabilidade entre diferentes tecnologias. Foi proposto ainda, pela mesma organização, um modelo de referência para que novos Sistemas de Gerenciamento de *Workflow* fossem implementados futuramente. As relações entre os principais componentes propostos por tal modelo podem ser explicitadas na figura 2.4 e podem ainda serem definidos de acordo com os conceitos abaixo:

1. **Processo de Negócio:** segundo a WfMC [48], é um conjunto de um ou mais procedimentos ou atividades os quais em sua coletividade tornam-se intrínsecos a um conjunto de ações necessárias para a realização de um objetivo de negócio, podendo o mesmo ser utilizado ainda, para a verificação do cumprimento de uma meta.

2. **Sistema Gerenciador de Workflow (WfMS):** os sistemas gerenciadores de *workflow* propiciam os meios tecnológicos necessários para a realização das tarefas pertinentes ao processo de negócio que compõe o *workflow*. Podendo ou não, solicitar intervenção humana durante sua execução. Tais sistemas também podem ou não possuirem inteligência para tomar decisões de acordo com a natureza das tarefas e suas entradas [48].

3. **Definição de Processo:** é a representação de um processo por meio de linguagens específicas aos meios tecnológicos nos quais o *workflow* será executado [48]. Ex: padrões baseados em XML [4][26][27] e linguagens formais em geral.

4. **Sub-Processo:** unidade de menor granularidade que é um processo, porém faz o papel de unidade de um outro processo [48].

5. **Tarefa:** o mesmo que atividade.

6. **Atividade:** é uma unidade, composta de uma ou mais operações a serem realizadas e determinado artefato que forma um passo lógico de acordo com a definição do processo [48]. As atividades podem ser manuais (por exemplo, tarefas que exigem decisão humana) ou computadorizadas.

7. **Instância de Processo:** é uma linha de execução de um determinado processo. Um sistema de gerenciamento de *workflow* deve ser composto de um módulo que

o capacita a gerenciar várias instâncias simultâneas de um mesmo processo. Tais instâncias poderão ou não serem executadas concorrenetemente.

8. **Instância de Atividade:** é a representação de uma atividade em uma única execução de um processo. Tal atividade é gerenciada pelo sistema de gerenciamento de *workflow* [48].
9. **Item de Trabalho:** é um componente lógico da atividade. Uma atividade é composta por vários itens de trabalho.
10. **Chamada de Aplicação:** é a transferência do controle do processo para uma aplicação particular para processamento de um item de trabalho.
11. **Participante:** O mesmo que ator.
12. **Ator:** é o responsável pela execução de uma tarefa em sua totalidade ou parcialmente.
13. **Cargo/função(papel):** é o conjunto de atividades atribuidas a um determinado ator do processo.

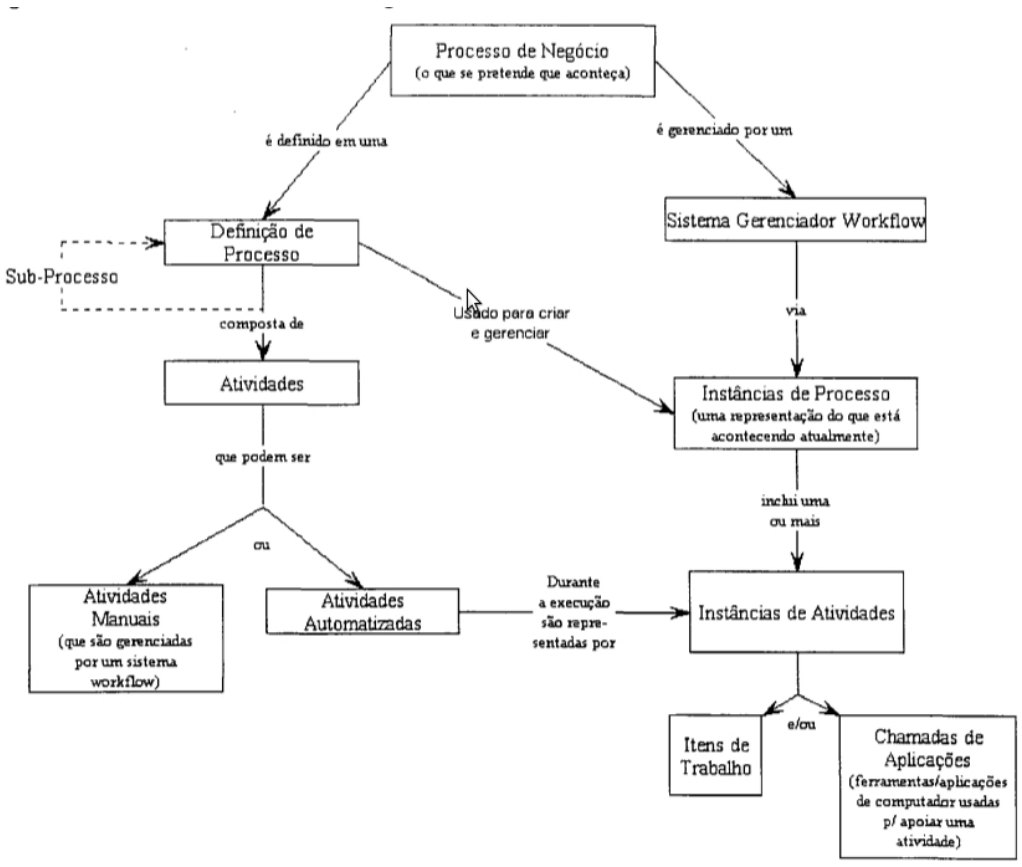

Figura 2.1: Ilustração dos componentes de um *workflow*. Adaptado de [48]

No cenário da década de 1990, baseada em comparações entre vários produtos disponíveis no mercado, a WfMC identificou a necessidade do desenvolvimento de um modelo genérico

de implementação de um sistema de gerenciamento de workflow que poderia ser ajustado à maioria dos produtos existentes no mercado da época. Tal modelo identifica os principais componentes funcionais e suas respectivas interfaces dentro de um sistema de gerenciamento de *workflow*. Tal arquitetura genérica é expressa graficamente na figura 2.4.

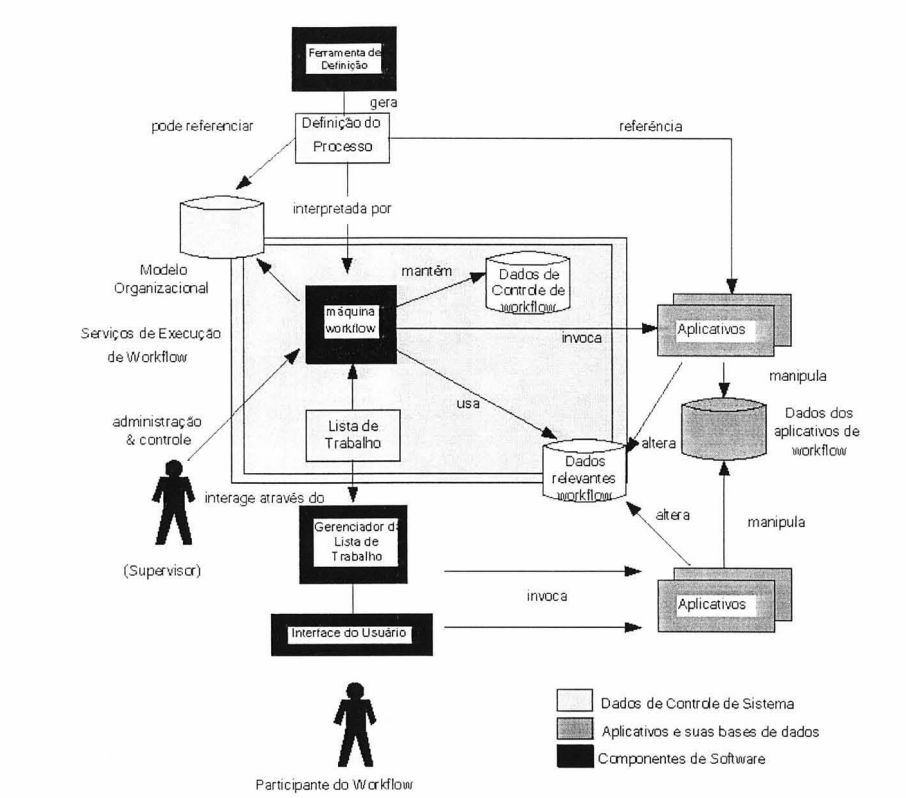

Figura 2.2: Arquitetura genérica de um WfMS. Adaptado de [48]

A partir de tal arquitetura genérica, os desenvolvedores dos sistemas de serenciamento de workflows podem utilizar o modelo como referência básica a ser seguida, podendo adaptar algumas interfaces e componentes de acordo com as necessidades da aplicação. Para fins de estudo apropriado, é possível dividir os blocos em três grandes grupos:

1. **Componentes de Software:** são os componenetes de software que oferecem suporte as funcionalidades do sistema de gerenciamento de workflows (como a ferramenta definição, máquina de *workflow*, gerenciador da lista de trabalho e interface do usuário).
2. **Dados de Controle do Sistema:** são dados utilizados pelos componentes do workflow (dados de controle, linguagens de marcação e os dados pertencentes ao fluxo de execução).
3. **Aplicativos e suas Bases de Dados:** são os aplicativos que não fazem parte do sistema de gerenciamento de workflow, mas podem ser invocados por ele como parte do sistema em si.

os sistemas gerenciadores de *workflow*, de acordo com a *Workflow Management Coalliton* [48] são os sistemas que definem, criam e gerenciam a execução de *workflow* por meio

do uso de produtos de software, executando-o em uma ou mais máquinas, os quais estão aptos a interpretarem a definição de um processo, interagirem com os participantes do *workflow* e, onde necessário, invocar o uso de ferramentas da tecnologia da informação ou aplicativos. Os WfMS propiciam os meios para a automação dos processos de negócios, de acordo com os procedimentos de determinada atividade, solicitam recursos humanos ou de máquinas e gerenciam atividades de trabalho. Segundo a WfMC [48], tais gerenciadores fornecem suporte a três áreas funcionais:

1. **Construção do *Workflow*:** é a definição do processo de *workflow* e suas tarefas. É o mapeamento formal do processo de negócio do mundo real para que este possa ser entendido pelos sistemas computacionais.
2. **Interação entre Participantes:** tratam da interação com usuários humanos e aplicações para o processamento de várias etapas de uma atividade.
3. **Execução do *Workflow* :** é atribuída a este componente o gerenciamento do *workflow* em um ambiente operacional, bem como o escalonamento das tarefas a serem empregadas em cada processo.

Nesse cenário, a Figura 2.3 mostra a relação entre às áreas funcionais.

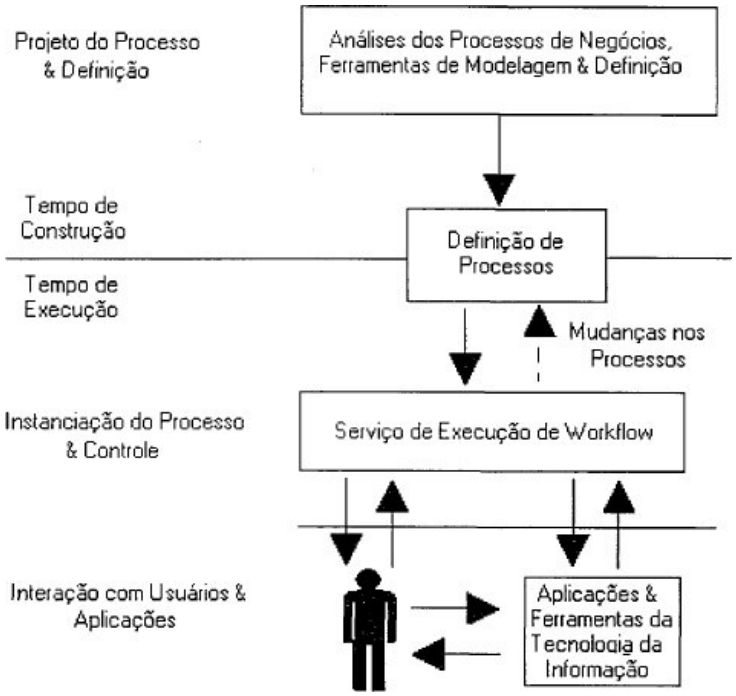

Figura 2.3: Arquitetura genérica da interação de um WfMS e seus agentes externos, adaptado de [48]

O Modelo de Referência para WfMS, é a proposta da WfMC para padronizar o desenvolvimento de aplicações baseadas na tecnologia do *workflow*, e propõe uma arquitetura básica para o desenvolvimento das aplicações. A figura 2.4 esboça a arquitetura genérica de um *workflow*.

A seguir estão as descrições dos componenetes do modelo de referência [48].

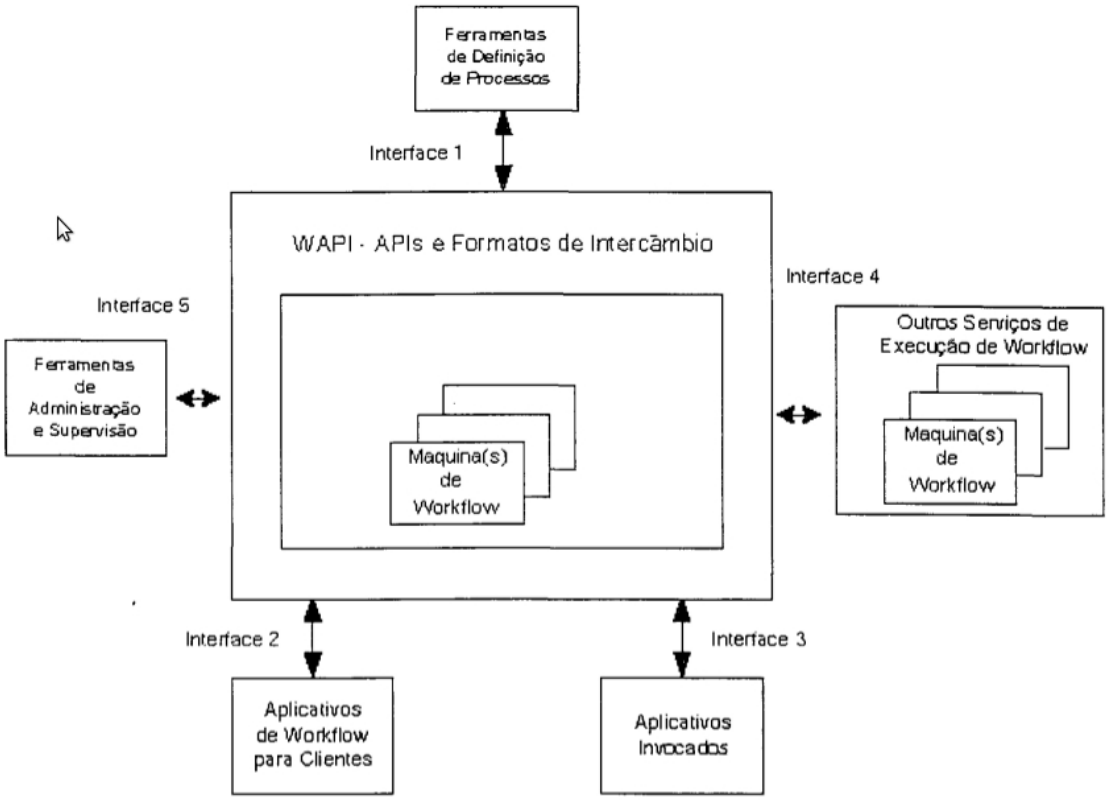

Figura 2.4: Modelo genérico de um WfMS, adaptado de [48]

1. **Serviço de Execução do *Workflow*:** serviço de software que consiste de uma ou mais máquinas de *workflow* para criar, gerenciar e executar as instâncias de *workflow*. As aplicações comunicam-se com este serviço por meio da WAPI (*Workflow Application Programming Interface*). Este serviço fornece o ambiente e os mecanismos de execução nos aquais as ativações e instanciações dos processos acontecem. A interação com o serviço de execução ocorrerá via interface de aplicações de *workflow* para clientes (Interface 2) ou interface de aplicações invocadas (Interface 3).

2. **Definição de Processos:** diferentes ferramentas podem ser utilizadas para analisar, modelar, descrever e documentar um processo de negócio (Interface 1). Pressupõe-se que cada produto que trabalhe com *workflow* forneça uma ferramenta para este fim. É importante que ao final do projeto e modelagem, o processo de definição possa ser interpretado e executado pelas máquinas de *workflow*.

3. **Aplicações de *Workflow* para Clientes:** é o software gerenciador de listas de trabalhos que interage com o usuário final nas atividades que envolvem recursos humanos (Interface 2). Uma lista de trabalho é uma lista de itens de trabalho atribuídos a um usuário ou grupo de usuários por uma máquina *workflow*.

4. **Interoperabilidade entre Serviços de Execução de *Workflow*:** permite a troca de itens de trabalho entre diferentes WfMSs (Interface 4). Logo, diferentes

máquinas *workflow* podem trabalhar em conjunto. O foco de padronização neste item está na forma de cooperação entre os WfMSs.

5. **Ferramentas de Supervisão e Administração:** estas ferramentas de supervisão e administração permitem a avaliação do estado geral e extração de métricas do sistema, o que é importante para as organizações (Interface 5).

2.2 *Workflows* Científicos

Apesar dos benefícios obtidos com o esforço da padronização, pela WfMC, várias pesquisas ainda foram necessárias para a adaptação do conceito de *workflows* para o domínio da pesquisa científica. Vários conceitos básicos sobre este tema serão expostos a seguir.

2.2.1 Definição

A pesquisa científica, ao longo dos anos, tornou-se um domínio de grande importância na aplicação da tecnologia dos *workflows*. De fato, existem várias pesquisas com resultados concretos sobre o tema [55][57][59], de maneira que os *Worflows* Científicos emergiram como um campo de pesquisa com definição própria. Várias ferramentas próprias para este tipo de domínio foram desenvolvidas [67] com o propósito único de apoiar os processos inerentes à pesquisa científica. Os *Workflows* científicos são redes de passos analíticos que representam a computação necessária à análise de grandes volumes de dados [50]. Eles podem envolver acessos à bases de dados, análises e mineração dos mesmos e muitos outros passos inclusive tarefas com grande esforço computacional em *workflows*, clusters de computadores de alta performance [65]. Os workflows científicos podem ser inclusive de grande escala não apenas em termos computacionais como também em número de tarefas, tempo total de execução entre outros requisitos não-funcionais [50]. No cenário atual, a ciência empírica tem se expandido através do intenso estudo por meio de experimentos físicos e o auxílio de ferramentas computacionais [40]. Tomando como exemplo, o ramo da astronomia, a pesquisa de imagens em bancos de dados astronômicos, frequentemente é parte dos processos de trabalho da instituição de pesquisa. Na bioinformática, que é o foco deste trabalho, grandes repositórios de dados armazenados em infraestruturas de escala mundial provêem dados para capacitarem os pesquisadores a compararem seus resultados com pesquisas realizadas em países, ou continentes distintos.

2.2.2 Requisitos de *Workflows* Científicos

Os workflows científicos, possuem uma série de requisitos que os distiguem de um workflow pertinente ao domínio dos negócios [55][41]. A seguir, uma lista dos mais comuns, de acordo com o trabalho realizado em [54]:

1. **Acesso transparente a recursos e serviços:** os acessos a recursos e serviços utilizados devem ser independentes da implementação interna dos mesmos. Isto é fornecido por meio de interfaces especificadas no próprio WfMS.

2. **Design de *workflows* por meio de composição de serviços e reuso:** o usuário deve ser capaz de agrupar vários serviços de baixo nível em um único serviço de mais alto nível.

3. **Escalabilidade:** para suportar o aumento escalável do volume de dados e computação necessários para alguns *workflows*, são necessárias interfaces adequadas para *middleware*.

4. **Execução Destacada:** os *workflows* devem permitir que o mecanismo de controle rode em segundo plano em uma máquina remota sem a necesidade de conexão constante à aplicação do usuário.

5. **Confiabilidade e Tolerância a Falhas:** O *workflow* deve permitir que planos de contingência possam ser especificados para o caso de erros.

6. **Interação com Usuário:** Alguns *workflows* podem permitir que o usuário pause o processo e inspecione resultados intermediários.

7. **Re-runs Inteligentes:** caso um usuário resolva rodar novamente o processo apenas com algumas diferenças de parâmetros, o *workflow* não deverá reiniciar mas sim refazer apenas os passos alterados.

8. **Provenança de dados:** um sistema deve gerar um *log* dos passos realizados garantindo a possibilidade de reprodução dos experimentos e resultados obtidos.

Provenança de dados

A proveniêcia de dados dos *workflows* científicos tornou-se um subcampo de pesquisa, devido a sua grande importância para os ambientes de pesquisa científica em geral. Padrões abertos como o *Open Provenance Model* [18] foram estabelecidos para criar-se uma uniformização, para que diferentes WfMSs possam utilizar o mesmo esquema de provenança. Trabalhos realizados em [36][30][63] ressaltam tal importância.

2.2.3 O Ciclo de Vida de um *Worfklow* Científico

Ludascher et al.[55] subdivide o ciclo de vida de um workflow científico em quatro fases distintas no que tange suas funcionalidades. Desde a sua concepção (design do workflow) até sua fase final (análise pós-execução).

1. **Design do *workflow*:** fase onde o usuário define o *workflow*, para o teste de um nova hipótese, ou então, obter um resultado específico. Durante esta fase, os usuários podem reutilizar *workflows* já existentes, parcialmente ou totalmente. Eles podem compartilhar a descrição do *workflow* (artefato gerado nesta fase) através de um repositorio público como o *myExperiment* [42].

2. **Preparação do *workflow*:** fase na qual os recursos necessários para a execução do workflow são definidos e alocados.

3. **Execução do *workflow*:** é a execução das tarefas em si. Fase onde dados serão processados e novos dados serão gerados. Para *workflows* de larga escala, que levam

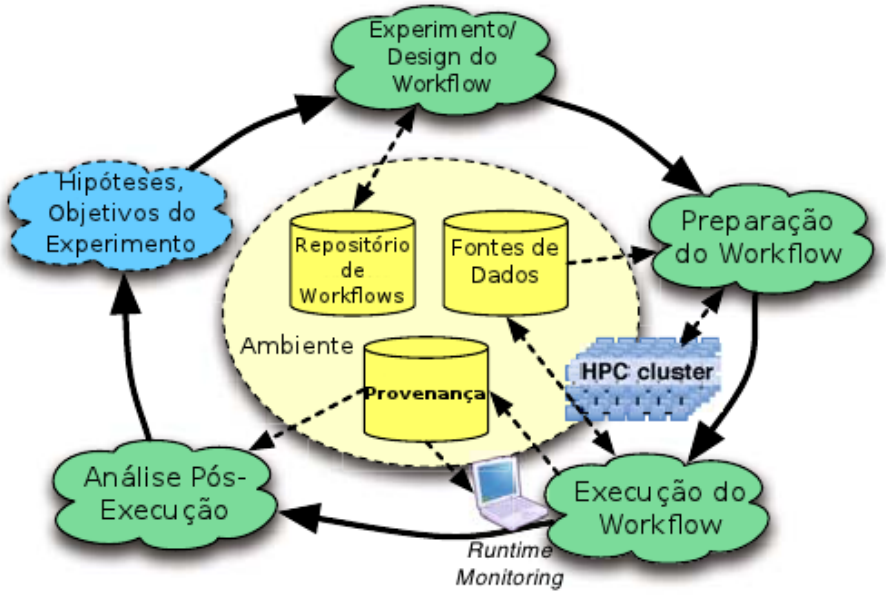

Figura 2.5: Ciclo de vida de um workflow, com suas fases. Adaptado de [55] .

um longo tempo de execução (na ordem de dias ou semanas), um serviço de monitoramento é necessário, através da interface de provenança. O usuário deverá ser capaz de interromper a execução do *workflow* quando necessário.

4. **Análise Pos-Execução:** os usuários fazem a validação dos requisitos e talvez otimizam o *workflow*, visualizando qual etapa tomou um maior tempo de execução.

2.3 Computação em Nuvem - *Cloud Computing*

A seguir, alguns conceitos básicos sobre Nuvens Computacionais.

2.3.1 Conceitos Básicos

Nos últimos anos, os sistemas distribuídos tem evoluido de tal maneira que a oferta de serviços de acordo com a demanda das aplicações tornou-se um paradigma aceitável na indústria tecnológica. Embora a definição do termo computação em nuvem (do inglês, *Cloud Computing*) gere muitas controvérsias entre acadêmicos [39], *Foster et. al* destaca que a computação em nuvem não é um novo paradigma de infraestrutura de recursos, tão somente um novo modelo de negócios, no qual usuários e organizações utilizariam os recursos disponíveis de acordo com a demanda de suas aplicações, em um formato de contrato de serviços. Desta forma, os usuários pagariam a computação pelos serviços, em vez de custearem toda uma infraestrutura particular. Existem várias definições acerca do termo computação em nuvem. *Foster et. al* [39] define o termo em da seguinte forma:

> Um sistema de distribuição de recursos computacionais em larga escala que é dirigido de acordo com a demanda das aplições, no qual um *pool* de recursos abstraídos, virtualizados, dinamicamente escaláveis, gerenciados por um serviço central bem como, poder de processamento, armazenamento e plataformas inteiras são fornecidos sob demanda para clientes externos por meio da rede mundial de computadores, a Internet.

Esta definição engloba muitos aspectos de um determinado sistema distribuído que podem caracterizá-lo como um sistema de Computação em Nuvem. Vários outros trabalhos estabelecem definições para a computação em nuvem [56][45][66], até mesmo definições formais já foram pesquisadas para este modelo de fornecimento de serviços [44].

2.3.2 Paradigma de Serviços

Várias nomenclaturas foram estabelecidas desde o advento da Computação em Nuvem. Em especial, devido à natureza da oferta sob demanda de serviços desde tipo de plataforma, o sufixo 'como serviço' (do inglês, *as a Service*) adquiriu notória popularidade no meio acadêmico e mercadológico [6]. Entretanto, dentre a variedade de serviços, três grupos se destacaram, para fins didáticos:

infraestrutura como Serviço - IaaS

É a entrega de recursos de infraestrutura de hardware como um serviço. Um ponto-chave deste grupo de serviços é o uso de recursos de hardware sob demanda. Isto permite que os usuários comprem mais recursos de acordo com suas necessidades ou disponibilidade de recursos. É a principal área de pesquisa da computação em nuvem das próximas décadas [34].

Geralmente, uma interface de gerenciamento é disponibilizada aos serviços das camadas superiores para que sejam possíveis a inicialização a partir da mesma. Como exemplo de tecnologias IaaS oferecidas por provedores, destacam-se:

- Amazon EC2 [1]: um serviço de processamento computacional sob demanda;
- Amazon S3 [2]: espaço para armazenamento sob demanda;
- GoGrid [9]: serviço para *deploy* de máquinas virtuais.

Plataforma como Serviço - PaaS

Define-se Plataforma como Serviço (do inglês, *Platform as a Service*) o conjunto de serviços capazes de oferecer funcionalidades como plataformas de programação, ambientes de implantação e execução, como um serviço. Os provedores de tais serviços deverão utilizar serviços de IaaS locais, ou oferecidos por outros provedores.

Assim, por meio de APIs fornecidas pelos provedores de serviços, é possível desenvolver ou adaptar aplicações para os ambientes de PaaS fornecido pelos provedores. Uma das vantagens decorrentes do uso destas plataformas é a escalabilidade das aplicações, que

pode ser aumentada apenas obtendo-se mais recursos dos provedores de IaaS, sem a necessidade de investimentos em mais equipamentos físicos.

Exemplos deste tipo de serviço são:

- Plataforma Heroku [11]: uma plataforma para desenvolvimento e *deploy* de aplicações web;
- Google App Engine [10]: uma plataforma para desenvolvimento e *deploy* de aplicações web;

Sofware como Serviço - SaaS

O Sofware Como Serviço (do inglês, *Software as a Service*) é a implementação do sofware sobre uma infraestrutura de computação em nuvem, e suas funcionalidades podem ser acessadas pela internet. O software como serviço permite que os usuário usufruam das funcionalidades do software, sem a necessidade de instalá-los em seus ambientes locais. O software como serviço, é o topo da pilha de serviços oferecidos pelas infraestruturas de computação em nuvem, e seus usuários não têm acesso aos serviços IaaS e Paas, podendo apenas executar a aplicação de acordo com configurações pré-estabelecidas [46].

Exemplos de aplicações que funcionam sobre o paradigma *SaaS* são:

- Google Drive: Um conjunto de aplicações de escritório e armazenamento de documentos que podem ser utilizadas através da internet.
- Grooveshark: Um reprodutor de mídia acessado diretamente por um *browser web* sem a necessidade de instalação de *software* local.

A Figura 2.6 abaixo ilustra o relacionamento entre as camadas de serviços explicitadas acima.

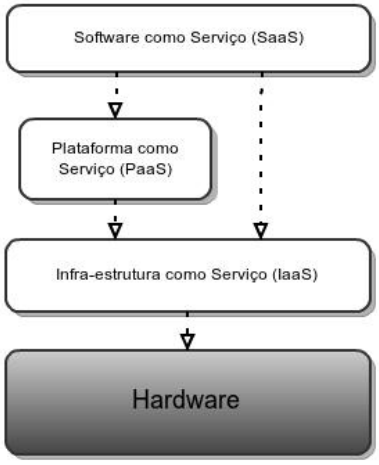

Figura 2.6: Relacionamento entre as camadas IaaS, PaaS, SaaS e o hardware do sistema.

2.3.3 Nuvens Computacionais Federadas

Em geral, tanto as grades computacionais quanto as nuvens computacionais tem como objetivo principal oferecer recursos sob demanda para o usuário, criando a ilusão de recursos ilimitados. Contudo, existem limitações em qualquer sistema de computação em nuvem, mesmo que este possua dimensões geográficas. Para a solução deste tipo de problema, a federação de serviços entre nuvens computacionais emergiu como uma solução. Provedores de serviços que não possuam recursos suficientes para executarem suas tarefas, podem delegá-las a outros provedores de serviços por meio das federações, ou então, em situação oposta, provedores que possuam recursos ociosos podem oferecê-los à outras partes via federação.

Bittman [7] dividiu o curso da evolução da computação em nuvem em três fases, as quais se distinguem de acordo com características específicas, e a terceira fase seria a atual fase. A primeira fase consiste na fase monolítica, na qual serviços de computação em nuvem são fornecidos por provedores de grande porte, construídos sobre arquitetures proprietarias. Como exemplos, pode-se citar o Google, Microsoft, Amazon e Salesforce. A segunda fase (cadeia vertical de fornecimento) é a fase na qual provedores de menor porte utilizam serviços de de provedores de maior porte. Como exemplo, podemos citar o Microsoft Azure [14] e o Google App Engine [10] que têm suas infraestruturas usadas para armazenar e rodar aplicações de empresas de menor porte. Neste ambiente, apesar da individualidade inerente às organizações, inicia-se uma integração entre nuvens, uma vez que clientes dos serviços oferecidos pelos provedores de grande porte citados, podem ser outra nuvem. A terceira fase proposta em [7] caracteriza-se como o estado-da-arte das nuvens computacionais, onde provedores de serviços federam-se horizontalmente para atingirem maior escabilidade e eficiência na execução de suas tarefas. Como consequência da grande necessidade por interoperabilidade, padrões começam a ser discutidos para fins de integração entre as plataformas, como por exemplo, o trabalho realizado em [53] propõe uma arquitetura de referência para implementação de camadas de federação de nuvens computacionais.

As nuvens computacionais federadas estão em um processo de popularização na indústria e no ambiente acadêmico. Assim sendo, existem alguns trabalhos que objetivam definir os requisitos para este tipo de ambiente e definir arquiteturas de referência para serem seguidas por futuras implementações [33][32].

Celesti et al. [33] define que os as nuvens computacionais federadas devem atender aos seguintes requisitos:

- Automatismo e escalabilidade: uma nuvem principal, utilizando mecanismos de descoberta, deverá ser dotada da capacidade de identificar as demais nuvens da federação bem como seus recursos, reagindo a mudanças como a retirada ou adição de uma nuvem à federação de maneira transparente e automática.
- Segurança e interoperabilidade: uma nuvem, mesmo que seja oriunda de um fabricante diferente da principal, deverá obedecer às políticas de segurança pré-estabelecidas para acoplar-se a federação, ou seja, deverá autenticar a si mesma para ganhar acesso aos recursos disponíveis.

Buyya et al.[32] ainda afima que as federações de nuvens computacionais devem atender os seguintes requisitos:

- Previsão do comportamento da aplicação: o sistema deverá ser capaz de tomar decisões sobre o melhor esquema de escalonamento para os serviços da federação, a partir de informações sobre caraterísticas e comportamentos dos serviços disponíveis. Previsões precisas de informações como poder de processamento e armazenamento disponível, banda de rede e outros deverão ser realizadas.
- Mapeamento flexível de serviços para recursos: Devido ao aumento dos custos de operação e demanda de energia dos sistemas federados, um requisito crítico a ser atendido é a maximização da eficiência de utilização dos recursos. O processo de mapeamento de serviços para recursos é um desafio complicado de ser superado, pois requer que o sistema compute a melhor configuração de maneira que a melhor qualidade de serviço (QoS) seja obtida. Isto não é algo trivial de se obter devido ao comportamento imprevisível dos recursos disponíveis.
- Monitoramento escalável de recursos do sistema: embora os componentes que integram a federação geralmente são distribuídos, as técnicas usuais empregam soluções centralizadas para gerenciamento e monitoramento do sistema. Tais soluções não são apropriadas para ambientes de federação devido à problemas de escalabilidade, peformance e confiabilidade que podem surgir a partir do gerenciamento de múltiplas filas de serviço e o grande volume de requisições. Portanto, *Buyya et al.* advoga que os serviços de gerenciamento e monitoramento de serviços devem ser baseados em modelos descentralizados de mensagens e indexação.

A figura 2.7 esquema gráfico abaixo ilustra a relação entre os componentes da arquitetura de referência proposta em [32] :

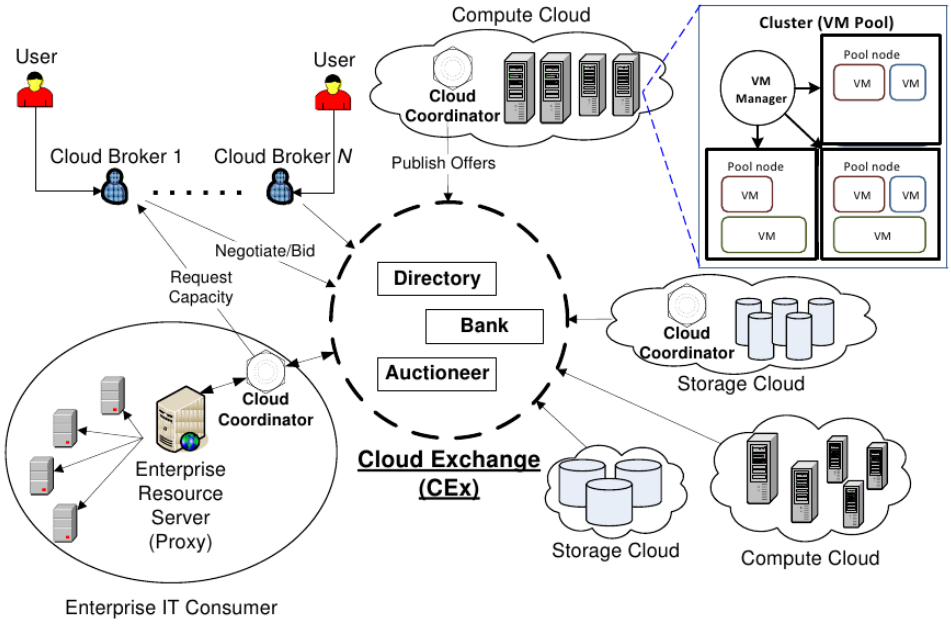

Figura 2.7: Relacionamento entre os componentes arquiteturais definidos em [32]

A figura 2.7 representa a arquitetura proposta por *Buyya et al.* [32]. Nela, o usuário interage com um componente denominado *Cloud Broker* (CB). Este componente é reponsável por criar um canal de comunicação entre o usuário e a federação, e por mapear

os recursos disponíveis conforme requisições sejam enviadas para a federação, atendendo os requisitos de qualidade previamente negociados. Para fins de obtenção de dados das nuvens disponíveis, o CB faz uso do componente denominado *Cloud Exchange* (CEx). Este componente é consultado pelo *Cloud Broker* de cada usuário para obtenção de informações como custos de utilização e recursos disponíveis. E por fim, existe um componente denominado *Cloud Coordinator* (CC), responsável por agregar estruturas na federação e expor os recursos aos restantes. E também, este componente implementa a autenticação e a negociação de QoS para cada requisição.

2.4 A Arquitetura Bionimbus

O Bionimbus [60], é uma arquitetura híbrida de federação de nuvens computacionais, que permite a integração de diferentes tipos de estrutura como nuvens computacionais, *grids* e até mesmo nós individuais de computação, que possuam ferramentas de bioninformática para serem disponibilizadas como serviços pra federação, de maneira transparente, flexível, tolerante a falhas e com grande capacidade de armazenamento e processamento [60]. Busca-se oferecer elasticidade às aplicações que fazerm requisições dos serviços que implemente a arquitetura. A seguir, uma visão geral da arquitetura é apresentada, seguida de sua interface de comunicação utilizada para este trabalho, juntamente com os casos de uso principais.

2.4.1 Visão Geral

O Bionimbus subdivide-se em vários componentes coesos, com responsabilidades bem definidas. A comunicação segue um esquema distribuído totalmente baseado em P2P com mensagens HTTP. A arquitetura define o conceito de *plugin* de integração, que consiste em um componente intermediador entre o provedor de um dado serviço e a federação. E nesse que é realizada a tradução da linguagem específica do provedor do serviço para a linguagem da federação.

Os módulos ilustrados na Figura 2.8 acima agem da seguinte forma:

- **Discovery Service:** é o componente responsável por coletar e armazenar dados sobre os diferentes nós computacionais federados pela arquitetura, como recursos disponíveis e as ferramentas disponíveis como serviço;
- **Scheduling Service:** é o componente responsável pelo escalonamento, inicialização e finalização das execuções das ferramentas de bioinformática;
- **Storage Service:** estabelece a estratégia de armazenamento dos arquivos consumidos e produzidos pelas ferramentas executadas.
- **Fault Tolerance Service:** monitora os componentes da federação, e em caso de falha, tenta recuperar os serviços ou execuções, de acordo com regras estabelecidas;
- **Monitoring Service:** registra, monitora e armazena dados sobre carga das nuvens da federação e sobre o histórico de execução de aplicações.
- **Security Service:** gerencia a política de acesso à federação e as credenciais de acesso dos usuários a cada uma das nuvens.

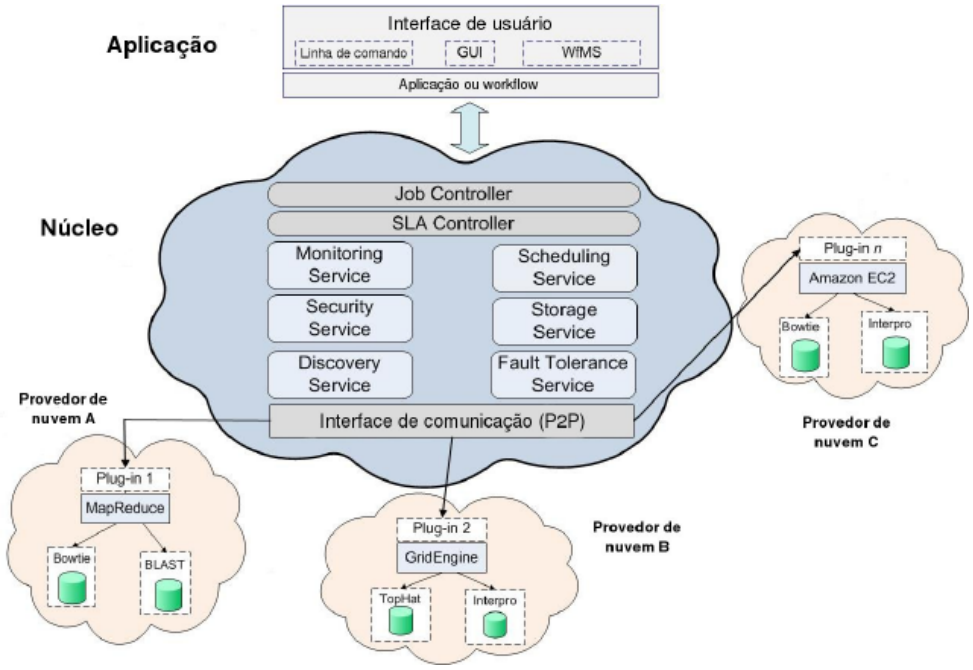

Figura 2.8: A arquitetura Bionimbus[60].

2.4.2 Classes Principais

Toda a comunicação realizada com a federação (serviços *plugin*) bem como os próprios componentes em si (serviços controladores) e aplicações de interação com o usuário em geral (WfMSs e terminais de comandos) é realizada por uma rede P2P. Cada requisição é formata por um par de requisição (*request*) e resposta(*reply*), definidas para cada tipo de requisição prevista pela arquitetura.

A arquitetura ainda define algumas classes de informação que são trocadas entre eles, e utilizadas para a implementação de funcionalidades para a estrutura:

- **PeerInfo:** informações referentes a um *peer*. Um *peer* é um membro da rede P2P. Ex.: Identificador único na federação, endereço de rede, latência de rede e tempo na rede.
- **PluginInfo:** informações referentes a um *plug-in* de integração, já definido nesta seção. Ex.: Dados da classe PeerInfo que descrevem o *peer* onde o *plug-in* executa, número total de CPUs e CPUs livres, tamanho total de armazenamento de dados e tamanho do armazenamento livre;
- **ServiceInfo:** Informações referentes a um serviço oferecido por um provedor nuvem computacional que faz parte da federação. Um serviço para a o Bionimbus é uma ferramente de bioinformática (ou outro domínio com requisitos semelhantes). Ex.: Identificador único, nome da ferramenta, parâmetros de configuração, arquivos de entrada e arquivos de saída;
- **JobInfo:** informações referentes a um *job*. Um *job* é uma requisição de execução de um serviço na federação. Ex.: identificador único, identificador do serviço solicitado e argumentos para o serviço;

- **TaskInfo:** informações referentes a uma *task*. Uma *task* é uma instância de um *job*. Por isso, uma *task* está necessáriamente associado à algum *plug-in* de integração. Ex.: identificador único, informações da classe JobInfo correspondente, a PluginInfo do local de execução e seu estado de execução na infraestrutura, os quais podem ser "EM ESPERA", "EM EXECUÇÃO"e "FINALIZADO".
- **FileInfo:** informações referentes a um arquivo armazenado na federação. Ex.: identificador único, nome e tamanho do arquivo;
- **PluginFileInfo:** informações sobre uma instância de um arquivo. Ex.: identificador único, sua FileInfo e a PluginInfo do local de armazenamento.

Além das classes de informação acima, o Bionimbus possui um interface de mensagens, totalmente baseadas em P2P, que obedecem à um esquema de requisição (*request*) e resposta (*reply*). A arquitetura prevê alguns Casos de Uso como essenciais para a execução da maioria das tarefas envolvendo a estrutura.

2.4.3 Casos de Uso principais

A seguir uma descrição dos Casos de Uso mais importantes, para a operação do Bionimbus. Como a ferramenta foi desenvolvida para a execução de ferramentas de bioinformática, então é pressuposto que as ferramentas possuem o seguinte padrão de utilização:

- Um conjunto de arquivos de entrada que serão processados pelas ferramentas, que podem ser originados diretamente de sequenciadores de bioinformática, ou outra ferramenta de bioninformática;
- Um conjunto de arquivos de saída contendo o resultado da análise realizada pela ferramenta;
- Um conjunto de parâmetros de execução que podem influenciar, por exemplo, na utilização de recursos de hardware ou no formato dos arquivos de entrda e de saída.

Assim, a arquitetura foi desenvolvida de maneira a acomodar uma série de casos de uso capazes de suportar a maioria das operações usuais de um ambiente de pesquisa em bioinformática. Qualquer serviço de interação com o usuário, sendo este um simples terminal, ou até mesmo um WfMS (objetivo deste trabalho) deverão acomodar-se sobre os casos de uso listados a seguir.

Upload de Arquivos

Para que seja possível a execução de um serviço disponível na federação, é necessário que os arquivos estejam disponíveis na mesma. Para tanto, é necessário realizar o *upload* de um arquivo, quando o objetivo for realizar um processamento sobre o mesmo. Portanto, um caso de uso de *upload* pode ser descrito da seguinte forma:

1. O serviço de interação com o usuário (terminal ou WfMS), coleta informações sobre o arquivo a ser enviado por meio da classe FileInfo, e em seguinda envia um mensagem do tipo StoreReq através do método SyncCommunication.sendReq() utilizando as informações de FileInfo como parâmetro para o *Storage Service*. Vale ressaltar que

o método é síncrono, ou seja, ele aguardará a resposta do Storage Service para continuar a execução;

2. O *Storage Service* decide o melhor local para armazenamento e envia como resposta um mensagem do tipo StoreReply;
3. Com a resposta, o serviço de interação envia uma mensagem do tipo FilePutReq para o provedor baseando-se na resposta StoreReply obtida do passo anterior.
4. Após receber o arquivo completamente, o provedor realiza o armazenamento local em sua estrutura;
5. O provedor envia uma mensagem do tipo StoreAck para o *Storage Service* e depois uma mensagem do tipo FilePutReply para o serviço de interação do usuário, confirmando o armazenamento para ambos.

Após estes passos, o Caso de Uso Listagem de Arquivos poderá ser executado para verificar se o arquivo foi enviado corretamente à federação. O diagrama de sequência a seguir ilustra a interação entre os componentes do Bionimbus para este caso de uso.

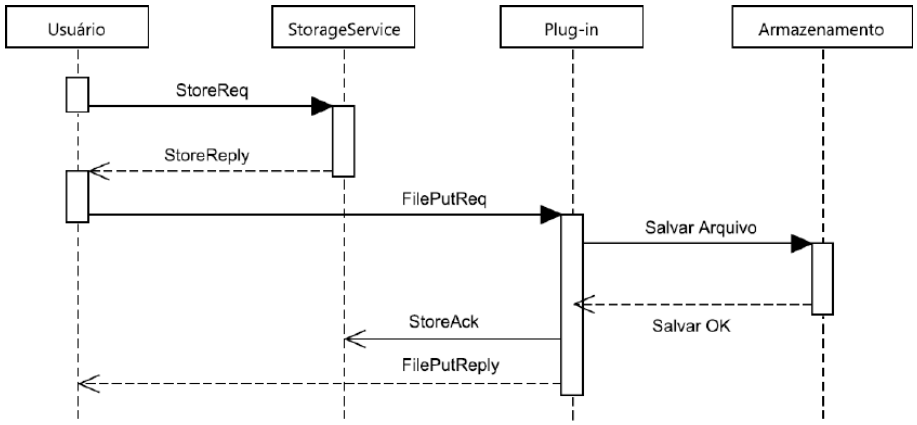

Figura 2.9: Sequência de mensagens para o *upload* de arquivos por um usuário.[60].

Listagem de Arquivos

A listagem de arquivos é necessária para se obter os identificadores únicos dos arquivos na federação. Estes identificadores são gerados durante o processo de *upload* dos arquivos e serão utilizados durante todo o ciclo de vida dos arquivos na federação, desde o seu *upload*, ou criação por outras ferramentas de bioinformática, até o seu *download* para a infraestrutura local do usuário.

Portanto, o caso de uso para listar os arquivos é feito seguindo-se os seguintes passos:

1. O Serviço de interação com o usuário envia uma mensagem do tipo ListReq pela rede P2P para o *Storage Service*;
2. O *Storage Service* responde com uma mensagem do tipo ListReply;
3. O serviço de interação, com os dados recebidos, é responsável por apresentá-los ao usuário, principalmente, os identificadores dos arquivos.

A Figura 2.10 ilustra graficamente o relacionamento entre os componentes da federação durante o processo de listagem de arquivos:

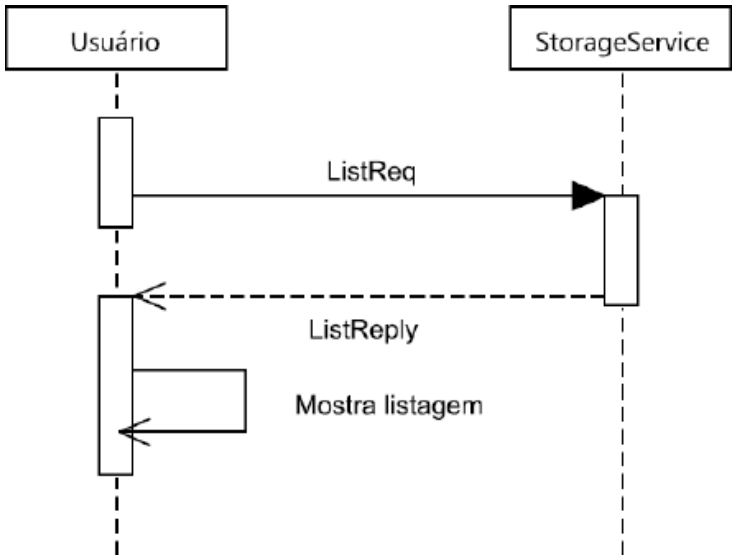

Figura 2.10: Sequência de mensagens para a listagem de arquivos.[60].

Download de Arquivos

Quando um arquivo é gerado por alguma ferramenta, ele é disponibilizado na federação. Assim, quando o usuário desejar obter um arquivo resultante de um processamento, ele deverá acionar o *download* do arquivo para obtê-lo. Para que a execução deste caso de uso se concretize, os seguintes passos deverão ser tomados:

1. O serviço de interação com o usuário envia uma mensagem do tipo GetReq para o *Storage Service*, contendo o identificador do arquivo desejado;
2. O *Storage Service* consulta suas tabelas e responde com uma mensage do tipo GetReply, com uma instância de uma classe de informação PluginInfo onde o arquivo pode ser obtido;
3. Com a informação PluginInfo, uma mensagem FilePrepReq é enviada para que o hospedeiro prepare o arquivo para ser carregado localmente;
4. O *plug-in* realiza seus preparativos, que pode ser baixar o arquivo de um armazenamento distribuído para a máquina onde ele mesmo executa, e envia a resposta do tipo FilePrepReply, indicando que está pronto para o *download*;
5. Quem iniciou o processo envia agora uma mensagem do tipo FileGetReq para o plug-in;
6. O *plug-in* em questão, responde com uma mensagem do tipo FileGetReply com o conteúdo do arquivo.

A Figura 2.11 mostra o relacionamento entre os componentes durante o processo de *download* de arquivos:

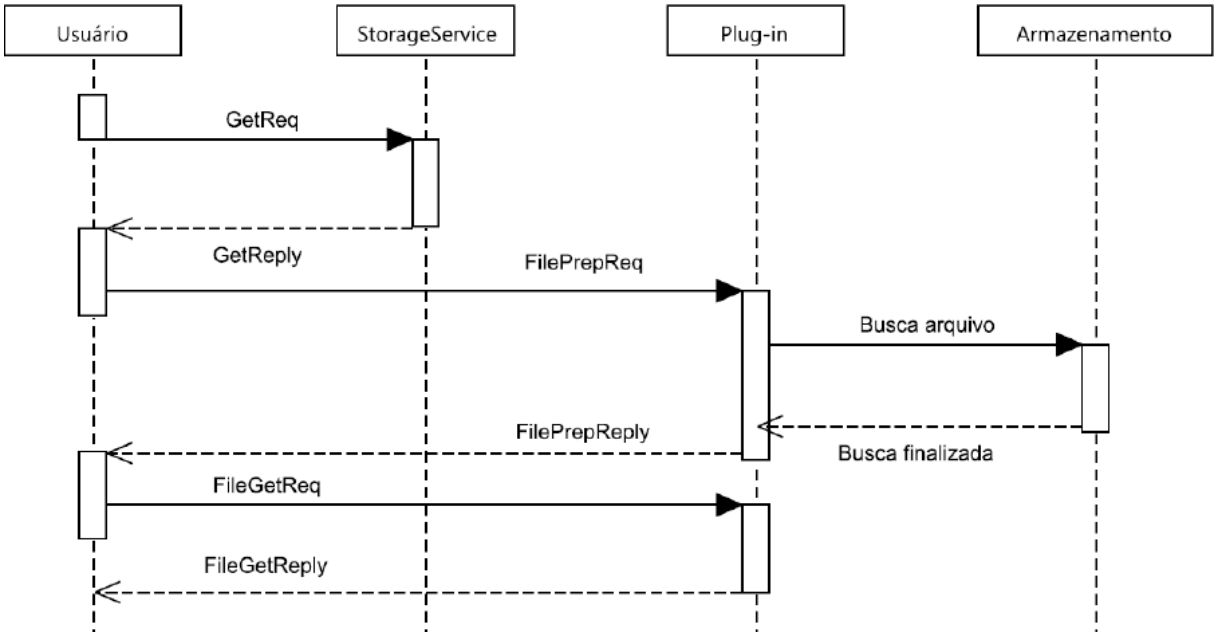

Figura 2.11: Sequência de mensagens para o *download* de arquivos.[60].

Submissão de *Jobs*

A submissão de *jobs* é a atividade central para a arquitetura Bionimbus. Nela, o serviço de interação com o usuário deverá indicar qual aplicação deverá executar e qual serão os arquivos de entrada e saída, bem como os parâmetros de execução. Vários jobs poderão ser submetidos simultâneamente. A arquitetura não pressupõe qualquer tipo de relação entre os *jobs*, sendo que os mesmos são tratados de forma individual. A relação de temporalidade entre os *jobs* deverão ser definidas inteiramente através dos serviços de interação com o usuário. Essa é exatamente a função dos WfMSs.

A sumissão de *jobs* é realizada por meio dos seguintes passos:

1. O serviço de interação com o usuário, após construir as informações sobre os *jobs*, através da classe de informação JobInfo, envia-os à federação por meio do *Job Controller*, através de uma mensagem do tipo JobStartReq;
2. O *Monitoring Service* cria o registro do *job* em sua estrutura de dados para monitoramento e requisita ao *Scheduling Service* por meio de uma mensagem so tipo JobSchedReq;
3. O *Scheduling Service* aciona a política de escalonamento configurada para cada um dos *jobs* enviados;
4. Com a indicação de onde serão executados, o *Scheduling Service* envia para cada *job* uma mensagem do tipo TaskStartReq com seus dados para o plug-in de integração da infraestrutura escolhida para a execução;
5. O *plug-in*, por sua vez, cria uma nova *task* vinculada ao *job* enviado, e responde com uma mensagem do tipo TaskStartReply para o *Scheduling Service* que usará os

dados da classe TaskInfo para consultar seu andamento, fazer reescalonamentos, se necessário, e informar o *Monitoring Service* do local de execução;

6. Os *plug-ins* verificam se há arquivos de entrada para as *tasks*, se houverem, os mesmos deverão ser obtidos por um processo de *download*, já descrito anteriormente.
7. Após os processo de *download* dos arquivos de entrada, o *plug-in*, inicia a execução da ferramenta sobre os arquivos.
8. Após encerrar a execução, o plug-in faz o *upload* dos arquivos de saída para a federação.
9. E por fim, o plug-in notifica o *Monitoring Service* com uma mensagem do tipo TaskEnd, e então, o status do *job* é atualizado para FINALIZADO, na federação.

A Figura 2.12 abaixo ilustra graficamente por meio de um diagrama de sequência, a submissão de um *job* pra federação:

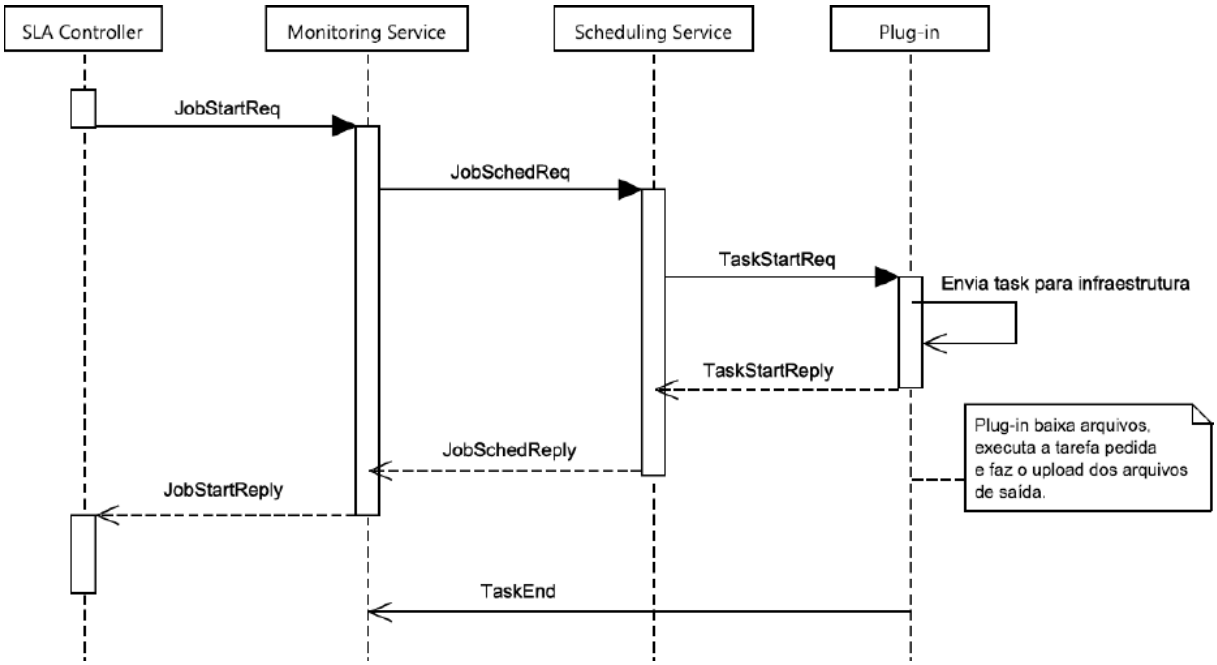

Figura 2.12: Sequência de mensagens para a submissão de um *job* para a federação.[60].

Os casos de uso acima, são as ferramentas básicas a serem utilizadas para integração de um WfMS com a federação. Vale ressaltar que até o momento da realização deste trabalho, não existia uma API completa para a utilização da federação, de maneira que a construção de uma API tornou-se um dos objetivos deste trabalho.

2.5 Workflows para Bioinformática

Na natureza, é possível distinguir todas as entidades existentes, entre vivas e não-vivas [51]. De certa maneira, ambas as formas de seres são compostas pelos mesmos

átomos. Infelizmente não podemos destacar com alta precisão o exato mecanismo que torna uma matéria viva ou não, embora seja possível discernir acerca da complexidade do arranjo das moléculas de cada ser vivo. Por exemplo, seres humanos são organismos altamente complexos e estão em constante transformação enquanto os vírus são entidades relativamente simples e unicelulares. Não obstante a complexidade inerente aos seres vivos, é viavel apontar dois elementos dos quais todos os seres são constituídos. As proteínas e os ácidos nucleicos. Tais moléculas podem ser codificadas como sequências de caracteres para tratamento computacional.

2.5.1 Proteínas

Os organismos vivos são formados, em sua maior parte, por compostos químicos denominados proteínas. As proteínas (ou substâncias proteícas) ainda podem subdividir-se em duas classificações distintas: as proteínas estruturais que agem como unidades que compõem os os organismos físicamente, enquanto outras proteínas conhecidas como enzimas, agem como agentes catalisadores das reações químicas que compõem o metabolismo dos seres vivos. Outras funções das proteínas são: o transporte de oxigênio os anticorpos que são células que atuam na defesa do organismo contra agentes externos. A proteína é uma cadeia de unidades menores denomidas aminoácidos. Os aminoácidos são moléculas orgânicas compostas por um conjunto componentes. A forma mais comum de um aminoácido é aprensentada como um composto por um carbono central ao qual se ligam quatro grupos : uma amina (NH2), grupo carboxílico (COOH), um hidrogênio e um composto único que caracteriza a molécula, diferenciando o aminoácido dos demais. Os aminoácidos podem compor cadeias, unidas por ligações peptídicas. Tais cadeias compões as proteínas. Os aminoácidos geralmente são introduzidos no organismo via alimentação ou fabricados por outros processos pertinentes ao metabolismo do organismo.

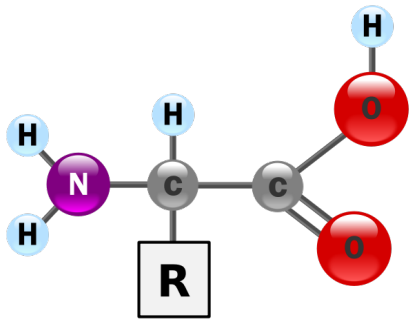

Figura 2.13: Estrutura básica de um aminoácido

2.5.2 Ácidos Nucleicos

Os ácidos nucleicos exercem um papel central no que tange a síntese proteíca. É partir destes que serão obtidas informações a partir do código genético necessárias para a geração de novas proteínas. Existem dois tipos de Ácidos Nucleicos: o ácido desoxirribonucleico, o DNA e o ácido ribonucléico, o RNA.

DNA

O DNA ou ácido desoxirribonucleico (do inglês, DNA ou *dezoxyrribonucleic acyd*) é uma molécula no formato de dupla hélice composta por duas longas fitas complementares de nucleotídeos que unem-se por pontes de hidrogênio entre os pares de base Guanina (G) - Citosina (C) e Adenina (A) - Timina (T). Armazenamento e transmissão de informações são as únicas funções conhecidas do DNA. Este codifica a informação por meio da sequência de nucleotídeos ao longo da fita. Cada base, A, C, T ou G, pode ser considerada como uma das letras de um alfabeto de quatro letras que significam mensagens biológicas na estrutura química do DNA. Nos organismos eucariotos, isto é, aqueles que possuem em suas células um núcleo delimitado, o DNA localiza-se justaposto ao núcleo celular. O DNA nuclear é dividido em uma série de diferentes cromossomos. Cada cromossomo consiste de uma única e enorme molécula de DNA linear com proteínas associadas que dobram e empacotam a fita de DNA em uma estrutura mais compacta. Dessa forma, um cromossomo é formado de uma longa molécula de DNA que contém inúmeros genes organizados linearmente.

RNA

O RNA ou ácido ribonucléico (do inglês, *ribonucleic acid*) é uma molécula composta por uma fita simples formada pela sequência de nucleotídeos que contêm as bases Guanina (G), Citosina (C), Adenina (A) e Uracila (U), sendo esta última divergente em relação ao DNA. Os RNAs possuem uma variedade de funções e, nas células, são encontrados como diversas classes. Os RNAs ribossômicos (rRNA) são componentes estruturais dos ribossomos, unidades celulares que realizam a síntese de proteínas. Os RNAs mensageiros (mRNA) são intermediários que transportam a informação genética de um ou de poucos genes até os ribossomos, onde as proteínas correspondentes podem ser sintetizadas. Os RNAs transportadores (tRNA) são moléculas adaptadores que traduzem a informação presente no mRNA em uma sequência específica de aminoácidos.

2.5.3 Mecanismos da Genética Molecular

Cromossomos e Genes

Toda a informação necessária a síntese protéica de um ser vivo encontra-se em seu DNA, ou mais especificamente, essa informação está disperso pela sequência de bases presentes nas fitas do DNA. Mesmo que o DNA armazene as informações necessárias à produção de proteínas, nem todas as sequências ne nucleotídeos do DNA codificam essa informação. Para cada proteína sintetizada por um organismo, existe uma porção de DNA a correspondente onde estão presentes os dados para sua fabricação. A esta porção de DNA contendo a informação de uma proteína damos o nome de gene. Os genes são distribuídos por meio das grandes moléculas de DNA, chamadas cromossomos. O número e o tamanho dos cromossomos varia de espécie para espécie, e não está diretamente ligado a complexidade da mesma.

Os genes armazenam a informação necessária para síntese de uma proteína. Uma dada sequência de aminoácidos permite a identificação precisa de uma proteína. No caso dos genes, os aminoácidos são codificados por meio de trincas de nucleotídeos, conhecido como

códons. Cada trinca corresponde a um aminoácido. Como existem 4 bases nitrogenadas no DNA, é possível formar 64 trincas diferentes, no entanto existem apenas 20 aminoácidos na natureza. A consequência disso é que diversos aminoácidos são codificados por mais de um códon. Essa redudância permite diminuir o impacto das mutações, que são alterações na sequência de nucleotídeos do DNA. Assim a informação presente no DNA é conservada por um período maior de tempo. A figura abaixo, mostra a associação entre códons e aminoácidos. Esta tabela, é conhecida como tabela do código genético. O STOP é um sinal específico utilizado para indicar o final do processo de síntese de uma proteína. Nessa figura, as bases mostradas não são bases do DNA, mas sim as bases do RNA. Isso é razoável, pois o RNA transporta a informação contida num gene para o local onde as proteínas serão sintetizadas, processo que será explicitado a seguir.

Primeira base	2a. base				Terceira base
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Figura 2.14: A tabela do código genético. Fonte:[22]

Transcricão, Tradução e Síntese Protéica

O *promotor* é a região que antecede cada gene no DNA e serve como indicação para o mecanismo celular que um gene está na próxima posição. O códon AUG também é um sinal de início de gene. Após reconhecer o início de uma trilha de genes, uma cópia do gene é feita para uma molécula de RNA. Este RNA resultante é o RNA mensageiro, ou mRNA. Terá exatamente a mesma sequência original, apenas substituindo U por T. Este processo é denominado **transcrição**. O mRNA também será usado nas estruturas celulares chamadas ribossomos para a síntese da proteína.

A transcrição, como fora descrito, é válida para organismos classificados como **procarióticos**. Nestes organismos o DNA encontra-se livre pela célula, pois não existe membrana nuclear. Exemplos de organismos procarióticos são bactérias e algas azuis. Todos

os outros organismos classificam-se como **eucarióticos**, uma vez que os seus núcleos são separados do resto das células por meio de uma membrana nuclear, e o DNA é posto no núcleo celular. Em tais organismos o processo de transcrição é mais complexo. Os genes dos seres eucariotos são compostos de duas partes, os íntrons e os éxons. Após a transcrição, os íntrons são removidos do mRNA. Sendo assim, em um organismo eucarioto, nem todas as bases de um gene são utilizadas na transcrição. Ao DNA contendo todas as bases do gene, denominamos DNA genômico, e às bases do DNA presentes no mRNA após a remoção dos íntrons chamamos DNA codificador (cDNA).

Finda a transcrição, a proteína será sintetizada em estruturas celulares chamadas de ribossomos. Os ribossomos são estruturas compotas de proteínas e um tipo especial de RNA, chamado de RNA ribossômico e abreviado como rRNA. Os ribossomos funcionam como linhas de montagem de proteínas, lendo a informação para síntese do mRNA e utilizando moléculas conhecidas como RNA transportadores (tRNA) para realizar a tradução dos códons para os aminoácidos correspondentes. Mecanismos celulares realizam a junção dos diversos aminoácidos. Os RNAs são as moléculas responsáveis por efetuar a conexão entre os códons e os aminoácidos correspondetes, em um processo chamado **tradução**. Cada tRNA é composto por duas partes, uma delas possui afinidade química à um dado códon, enquanto a outra liga-se com facilidade ao aminoácido correspondente ao códon. Conforme a fita de mRNA passa pelo ribossomo, um tRNA correspondente ao códon sendo lido pelo ribossomo liga-se ao códo em questão, trazendo consigo o aminoácido correspondente. Uma enzima então catalisa a ligação pepetídica para adicionar o aminoácido em questão à proteína. A síntese prossegue assim, um aminoácido de cada vez, parando apenas quando um códon do tipo STOP é encontrado. Quando isso ocorre, a proteína desliga-se do ribossomo e é liberada na célula. O mRNA é degradado para posterior reaproveitamento dos seus componentes.

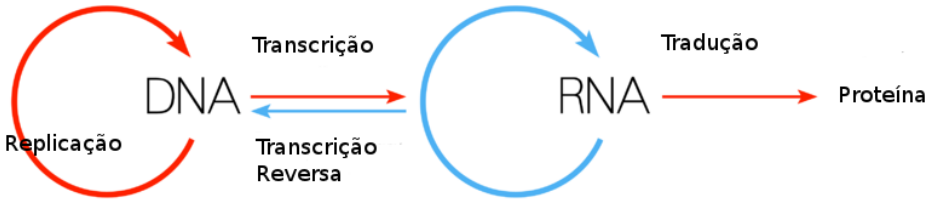

Figura 2.15: O Dogma Central da Biologia Molecular.

DNA-Não codificante e Quadros de Leitura

O DNA não-coficante (do inglês, Junk DNA ou Non-Coding DNA) é a fração do cromossomo que não é utilizada durante todo o metabolismo celular. No passado, acreditava-se que tal porção de DNA não exercia função alguma, apesar de pesquisas atuais levantarem especulações acerca de funções que o material poderia exercer, ainda que seja desconhecidas.

Especie	Número de Cromossomos	Tamanho do Genoma
Bacteriofago (virus)	1	$5 * 10^4$
Escherichia coli (bactéria)	1	$5 * 10^6$
Saccharomyces Cerevisae (levedura)	32	$1 * 10^7$
Caenorhabditis elegans (verme)	12	$1 * 10^9$
Droshophila melanogaster (bicho da goiaba)	8	$2 * 10^8$
Homo Sapiens (Humano)	46	$3 * 10^9$

Tabela 2.1: Relação de espécies e número de cromossomos. Fonte: [51]

Cromossomos

O número de cromossosmos em um genoma é característico de cada espécie. Por exemplo, todas as células do corpo humano possuem 46 cromossomos enquanto todas as espécies de ratos possuem 40 cromossomos em sua composição. A tabela abaixo relaciona o número de cromossomos e o tamanho do genoma em pares de base para várias espécies.

2.5.4 Projetos Genoma

Os avanços tecnológicos dos equipamentos utilizados nos laboratórios tornou a laboriosa tarefa de análise celular algo mais simples de ser executado. Vários equipamentos capazes de determinar a função de proteínas e identificar genes foram desenvolvidos. Dentre as ferramentas, destacam-se o isolamento, a clonagem e o **sequenciamento** de DNA. Denomina-se como sequenciamento, a tarefa de obtenção dos nucleotídeos que compõem fragmentos de sequências pertencentes ao DNA ou ao RNA de um ou mais organismos em um projeto genome, dependendo dos objetivos de cada projeto. Estes fragmentos são também chamados de *short-read-sequences* SRS.

Um projeto genoma desenvolve-se em geral através dos esforços de equipes multidisciplinares compostas por biólogos e cientistas da computação. Dentre os mais variados projetos genoma, destacam-se entre eles:

- Reconstrução do genoma de um organismo (projetos genoma);
- Obtenção dos transcritos de RNA (transcriptoma);
- Estudo de material genético coletado de amostras do ambiente (epigenética);
- Pesquisa de mudanças na expressão genética de tecidos afetados por doenças (genes diferencialmente expressos);
- Geração de árvores filogenéticas.

Em projetos de sequenciamento, um grande volume de dados é gerado, uma vez que não é incomum um organismo gerar bilhões de fragmentos de DNA ou RNA no processo de sequenciamento. Em seguida, estes dados são convertidos para cadeias de caracteres (*strings*) formadas pelas letras **A**, **C**, **G**, **T** ou **U**, cada uma correspondendo a uma das bases nitrogenadas de DNA ou RNA. Essa primeira parte do projeto é realizada nos

laboratórios de biologia molecular. As demais fases são todas realizadas em um laboratório de bioinformática, que centraliza o armazenamento, o gerenciamento e o processamento dos dados gerados pelo projeto. Vários modelos de máquinas de sequenciamento estão disponíveis no mercado, destacando-se entre elas o Illumina[12] e o 454[21]. *Saldanha*[60] destaca dois exemplos capazes de ilustrar uma situação típica de um ambiente de pesquisa em Biormática:

- Filichkin et al. [38] produziram e trabalharam com aproximadamente 271 milhões de SRS, cada uma dela com 32 pares de bases nitrogenadas, com o objetivo de identificar *splicing* alternativos da planta *Arabidopsis thailiana*. Este organismo possui um genoma relativamente pequenco, de aproximadamente 120 milhões de pares de bases. Dessa forma, foram produzidos aproximadamente 17,3 GB de dados no sequenciamento, os quais foram mapeados a mais ou menos 240 MB de dados;
- Sultan et al [64] também tinham o objetivo de identificar *splicing* alternativos, mas no genoma humano. Para isso, trabalharam com cerca de 15 milhões de SRS e os 3 bilhões de bases nitrogenadas do genoma humano completo. Isso equivale a mapear 960 MB a 6 GB.

Os exemplos acima demonstram um situação típica de um experimento *in silico* típico. Uma grande quantidade de dados foram gerados em ambos os experimentos, na ordem de dezenas de *Gigabytes*. Para cada fase de um experimento, existe uma série de ferramentas. Os sistemas gerenciadores de workflows emergiram naturalmente como uma ferramente de auxilio ao processo de diminuição da complexidade na gestão de todas essas ferramentas.

2.5.5 Filogenia

Na Biologia, a filogenia é o estudo da relação evolutiva entre grupos de organismos (por exemplo, espécies, populações), que é descoberto por meio de sequenciamento de dados moleculares e matrizes de dados morfológicos. O resultado dos estudos filogenéticos é a história evolutiva dos grupos taxonômicos, ou seja, sua filogenia[20]. A **taxonomia**, classificação, identificação e designação dos organismos, é baseada, em grande parte, da filogenia, mas são metodológicamente distintas.

Os campos de filogenina com sobreposição na taxonomia forma a sistemética filogenética, uma metodologia cladística com características derivadas (sinapomorfias) utilizadas na busca do ancestral descendente na árvore. Na sistemática biológica como um todo, as análises filogenéticas tornaram-se essenciais na pesquisa da árvoce evolucionária da vida.

Construção de Árvore Filogenética

A evolução é considerada um processo de ramificação, onde as populações são alteradas ao longo do tempo e formam especiação em ramificações separadas, hibridizam juntas ou termina em extinção. Isto pode ser visualizado em uma árvore filogenética. O problema da filogenia é que os dados genéticos estão disponíveis apenas para taxons vivos e nos registros fósseis (dados osteometricos) contendo poucos dados e características morfológicas ambíguas[62]. Uma árvore filogenética representa uma hipótese da ordem dos eventos

evolucionários ocorridos. Cladística é o atual método de escolha para inferir árvores filogenéticas. Os métodos mais frequentemente utilizados para a inferência filogenias incluem máxima parcimônia, semelhanças e MCMC baseada em inferência bayesiana. Fenética, popular no século XX, mas agora em grande parte obsoleto, usa Matriz de distâncias baseados em métodos para a construção de árvores baseadas em semelhanças globais, que muitas vezes assumem relações filogenéticas aproximadas. Todos os métodos dependem de um modelo matemático explícito ou implícito que descreve a evolução das características observadas nas espécies e são normalmente utilizados pela Filogenética molecular, no qual os caracteres são alinhadas em sequências de nucleótidos ou aminoácidos.

2.6 Exemplo de Workflow em Bioinformática

Saldanha [61] utiliza para validação da arquitetura Bionimbus [60] um workflow de bioinformática para identificação de genes diferencialmente expressos em células humanas cancerosas do rim e do fígado, com fragmentos sequenciados por sequenciadores Illumina [12]. O workflow é composto por quatro fases, como mostra a figura 2.16:

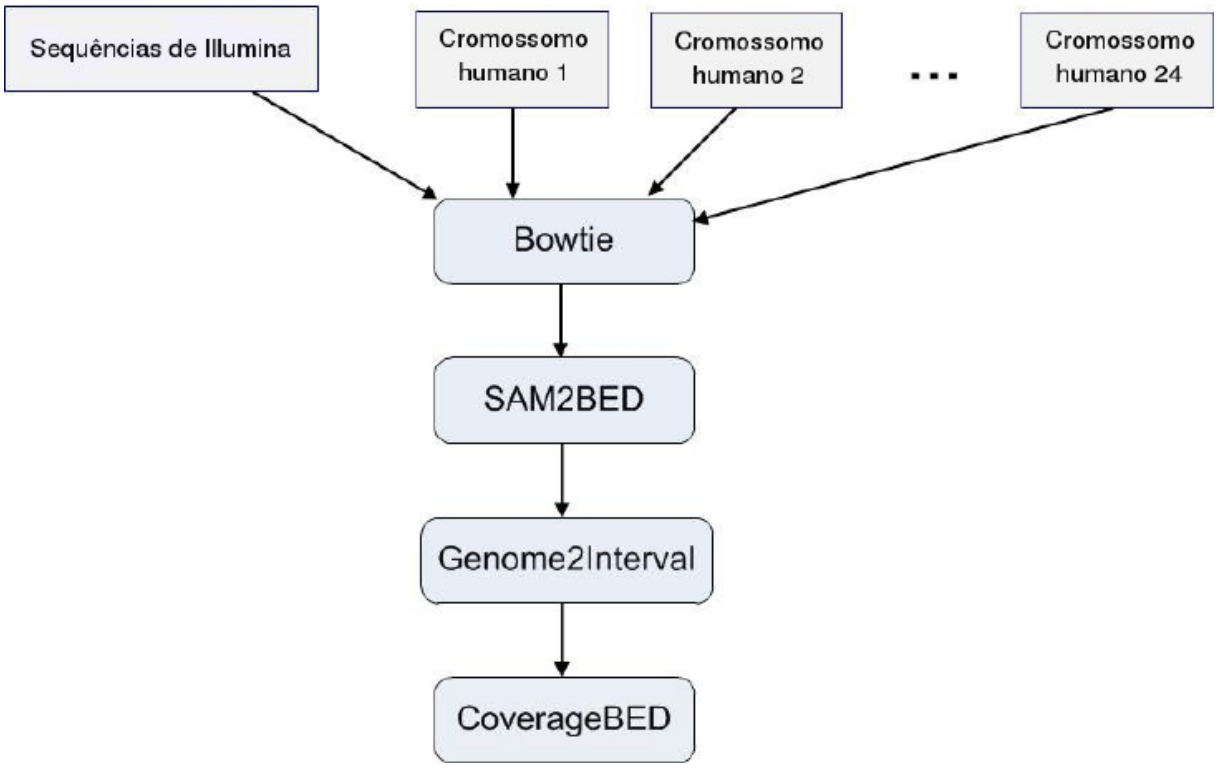

Figura 2.16: Workflow utilizado para identificar o nível de expressão de genes em células cancerosas do rim e do fígado. Fonte:[61]

Na primeira fase, de mapeamento, os fragmentos são mapeados aos 24 cromossomos humanos de referência. O objetivo é identificar a região do genoma de referência onde cada fragmento está localizado. Um conjunto de fragmentos mapeados na mesma região permitem inferir que elas possuem a mesma organização estrutural do genoma de referência. A ferramente utilizada para fazer esse mapeamento é a ferramenta Bowtie.

Na segunda fase, o formato SAM de saída do mapeamento é convertido para o formato BED com um script de conversão chamado sam2bed, implementado exclusivamente para esse workflow. Na terceira fase, com um script chamado genome2interval, são gerados intervalos de tamanho fixo baseados no tamanho de casa cromossomo que serão utilizados na fase seguinte. Nesta fase, a partir das saídas da segundae da terceira fase, são gerados histogramas que indicam o número de fragmentos mapeados por intervalo dos cromossomos com a ferramenta coverageBED da suite BEDtools.

O workflow acima, é composto por quatro componentes que podem ser adptados à interface do WfMS, de acordo com a interface que o mesmo oferecer.

Gomes L. S.[43] utiliza um workflow do domínio da Filogenia que consiste na construção de uma árvore filogenética a partir de sequências de proteínas geradas em processos de sequenciamento. A figura 2.17 abaixo ilustra um workflow implementado no sistema Bioside[3] que será utilizado como estudo de caso neste trabalho.

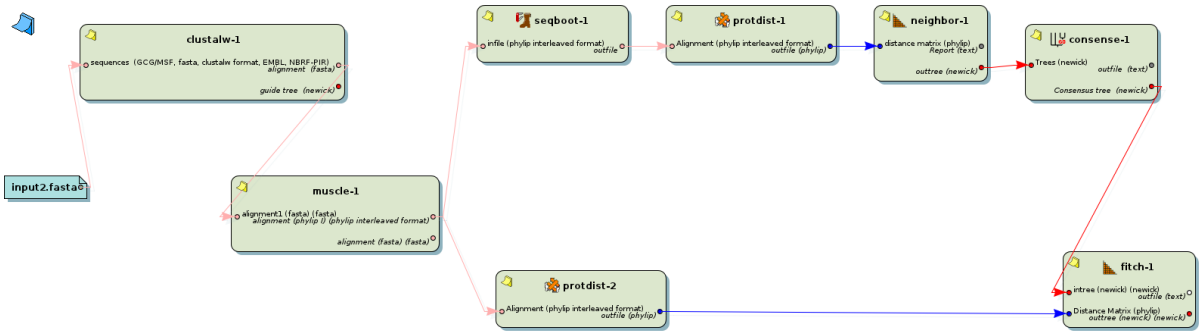

Figura 2.17: *Workflow* de construção de árvore filogenética. Fonte:[43]

Cada componente ilustrado no *workflow* acima são programas de bioinformática desenvolvidos especialmente para o domínio da Análise Filogenética. Todos os programas acima, pertencem ao pacote PHYLIP[19]. E todos estes programas estão pré-configurados no WfMS Bioside, como exposto na sessão 4.3. O *workflow* utiliza o programa Clustalw2 para um primeiro alinhamento múltiplo entre as sequências de proteínas GST e posteriormente o programa muscle para refinar o alinhamento do clustaw. Para a geração da árvore filogenética, o *workflow* fornece dois procedimentos independentes, utilizando dois métodos distintos do pacote PHYLIP, que lidam com dados na forma de matrizes com distâncias entre pares de sequências. Para a construção de uma árvore filogenética no pacote PHYLIP, usando um método de distância, é necessário primeiro executar um programa para calcular uma matriz de distância entre as sequências do alinhamento (o *workflow* utiliza o PROTDIST) para então utilizar um programa para a construção da árvore a partir dessa matriz (o *workflow* utiliza o NEIGHBOR e o FITCH).

Utilizando o NEIGHBOR, que calcula árvores pelo método Neighbor-Joining, a saída do muscle é usada como entrada para o seqboot, que realiza uma amostragem através do método de bootstrap, e o resultado é um arquivo com múltiplos alinhamentos gerados a partir do alinhamento original. A saída do seqboot é submetida para o programa protdist que calcula matrizes de distância para cada um dos conjuntos de dados, gerando um arquivo de saída que será utilizado no programa neighbor para a construção de múltiplas

árvores. O arquivo de saída do neighbor é submetido ao programa consense que avalia a significância das análises encontrando o consenso das árvores geradas para cada amostra.

O FITCH calcula árvores pelo método de Fitch-Margoliash, least-squares de Cavalli-Sforza e Edwards (ou outro método similar da mesma família). São métodos mais lentos e rigorosos e, portanto, muitas vezes o uso de bootstrap pode ser impeditivo. Nesse caso, no *workflow*, o alinhamento múltiplo original do MUSCLE serve de input direto para a construção da matriz de distância pelo PROTDIST, que servirá de input para o programa FITCH que gera uma única melhor árvore final. Os outputs do NEIGHBOR e do FITCH são arquivos que contêm diagrama(s) de árvore(s) e descrições de árvore(s) no formato Newickian. Este *workflow* utiliza apenas programas de linha de comando e como entrada um arquivo fornecido pelo usuário, que são características interessantes para o escopo deste trabalho. O estudo de caso de Geração de Árvore Filogenética já estava implementado no BioSide como um *workflow* de exemplo disponível para *download*.

Capítulo 3

Estudo Comparativo entre diferentes Sistemas Gerenciadores de Workflows

3.1 Taxonomia

Buyya et al. [67] definem alguns atributos que caracterizam um sistema geranciador de *workflows* para ambientes de grid. A taxonomia classifica os WfMSs de acordo com os seguintes atributos:

- **Projeto e Modelagem do *Workflow*:** diz respeito à maneira como os *workflows* podem ser modelados. Um *workflow* é composto por conexões entre componentes. Qualquer *workflow* pode ser modelado por um grafo dirigido, no qual o relacionamento entre os nós exprimem a relação temporal entre as tarefas. Um *workflow* baseado em Grafos Acíclicos Diridos (DAG) são *workflows* que podem ser modelados apenas por estruturas seriais, onde os componentes são enfileirados e executados em ordem, e em paralelo, no qual os componentes podem ser executados concorrentemente. Os *workflows* baseados em Grafos Cíclicos Dirigidos (*non-DAG*) são *workflows* que englobam todas as características do DAG, porém com a adição de execução de *loops* nas suas tarefas. Sua implementação é mais complexa que os *workflows* DAG.
- **Escalonamento das Tarefas:** esta característica enfatiza o mapeamento e gerenciamento da execução das tarefas do *workflow* em recursos compartilhados que não estão diretamente sobre o controle do sistema gerenciador de *workflow*. O escalonamento pode ser *centralizado*, onde um escalonador central processa as decisões para todas as tarefas do *workflow*, *hierárquico* onde existe um gerente central e múltiplos escalonadores para diferentes partes do *workflow*, e *descentralizado* não existe nenhuma entidade central, onde podem existir vários escalonadores que comunicam-se entre si, realizando balanceamento de carga entre os mesmos.
- **Tolerância a Falhas:** falhas de execução de *workflows* podem ocorrer por várias razões como falha de rede, sobrecarga de infraestrutura e não disponibilidade dos serviços, dentre muitas outras. O WfMS deve ser capaz de identificar e contornar as mesmas para oferecer um ambiente confiável. As técnicas pode se classificar em : *nível de tarefas*, que mascáram erros de tarefas individuais do *workflow* e *nível de workflow*, que consistem na manipulação da estrutura do *workflow* em geral.

- **Movimentação de Dados:** os arquivos de entradas das tarefas de um *workflow* devem estar disponíveis na estrutura na qual a tarefa será executada. Isto pode implicar em movimentações de dados em distância de ordens geográficas. As soluções em movimentação de dados classificam-se em, *centralizada*, onde todos os arquivos devem ser submetidos a um servidor central, para serem acessados pelos clientes, *mediada*, onde o sistema de arquivos é distribuído por vários nós, e *peer-to-peer* onde os nós realizam a transferência entre si diretamente dos arquivos.

Dentro da taxonomia proposta, foram tomadas algumas características como as mais importantes para este trabalho. Embora a taxonomia tenha analisado doze sistemas diferentes, apenas quatro deles estão relacionados na tabela . E alguns dos sistemas como o Watershed [37] e o Bioside [3] não fazem parte da classificação original feita por *Buyya et al.* [67], porém, os mesmo foram analisados utilizando-se os mesmos atributos. Todos os sistemas analisados possuem código aberto, de maneira a facilitar uma possível integração com a plataforma Bionimbus [60], principal objetivo deste trabalho.

3.2 Tecnologias pesquisadas

Segue abaixo, uma relação de algumas tecnologias pesquisadas.

3.2.1 Galaxy

O *Galaxy Framework* [8] é um ambiente de projeto, modelagem, execução e análise de *workflows*. Suas funcionalidades englobam todo o ciclo de vida de um *workflow* científico[29]. É dotado de uma interface gráfica específica para definição dos *workflows*, que são armazenados em formato XML. Este sistema utiliza um esquema de Grafos Acíclicos Dirigidos (DAG), ou seja, seus *workflows* não suportam *loops*. Seus *workflows* podem ser compartilhados pela rede social myExperiment[16], e também, existem projetos para converter seu formato para outras plataformas[28]. Outra vantagem é que o seu código fonte é aberto, e escrito em linguagem *Python*. Sua interface é ilustrada na figura 3.1

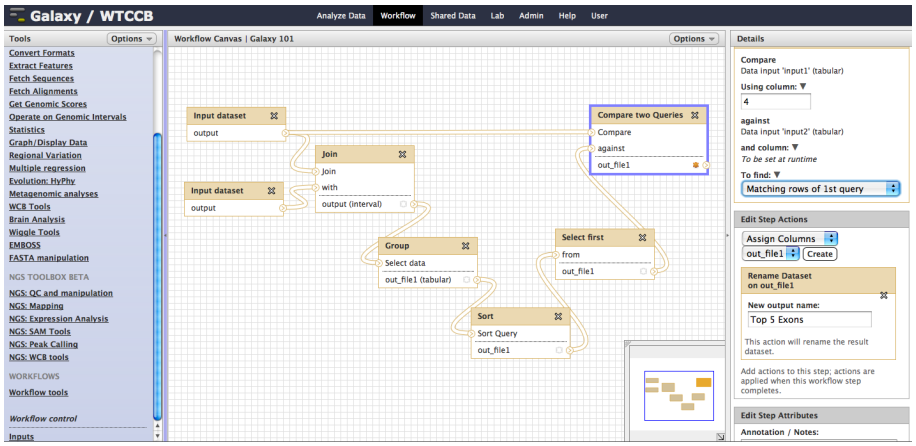

Figura 3.1: *Workflow* representado no Galaxy. Fonte: [8].

Apesar de seu código ser em *Python*, ferramentas externas devem ser integradas por meio de uma interface disponibilizada em Perl.

3.2.2 Taverna

O WfMS Taverna [49] é o resultado de um projeto de desenvolvimento do grupo myGrid [17] que consiste em uma ferramenta de construção e execução de *workflows* utilizando interfaces baseadas em *web-services* [49]. Esta ferramenta é agnóstica de domínio, ou seja, serviços pertinentes à qualquer domínio podem ser adaptados à ferramenta.

Os *workflows* são construidos por uma interface gráfica e amigável ao usuário final, e armazenados em um formato XML desenvolvido específicamente para a ferramenta, o *t2flow*. Embora este formato seja específico para a ferramenta, existem projetos capazes de converter as descrições de *workflow* para o formato Galaxy [28], e também, o formato é integrado à rede social myexperiment [16][42]. O sistema não suporta a execução de *workflows* com *loops*. Além da interface de execução de ferramentas, o sistema oferece uma interface própria para Provenança de dados de execução dos *workflows*, além de uma linguagem de consulta própria, baseada em SQL. Sua interface é exibida pela figura 3.2

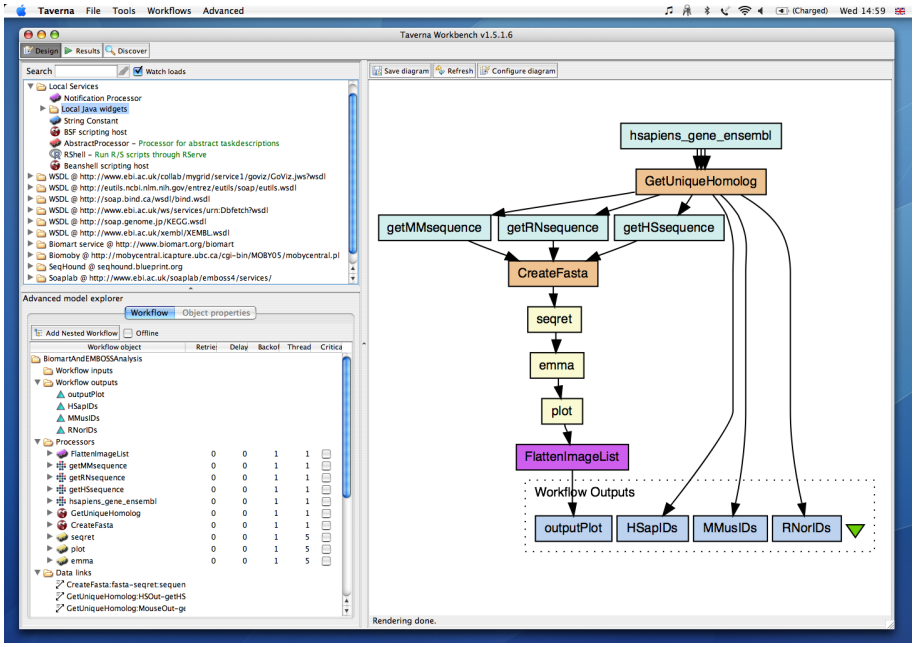

Figura 3.2: *Workflow* representado no Taverna. Fonte: [49].

3.2.3 Kepler

O Kepler [54] é um WfMS construído através de uma parceria entre várias universidades americanas com suporte do próprio governo americano [13]. De todas as ferramentas pesquisadas, é o único que suporta *loops* em seus *workflows*, ou seja, ele segue um esquema *non-DAG*. A ferramenta utiliza um fluxo de dados entre os componentes do *workflow* denominado *atores*. Depois de definido, o workflow é armazenado em um formato XML específico para o sistema denominado MoML. A ferramente possui por padrão, funções de provenança de dados programadas diretamente na ferramenta. A arquitetura prevê alguns pontos de extensão, das mais variadas formas, como por exemplo, APIs específicas para autenticação, interface gráfica e menus. Sua interface é exibida na Figura 3.4

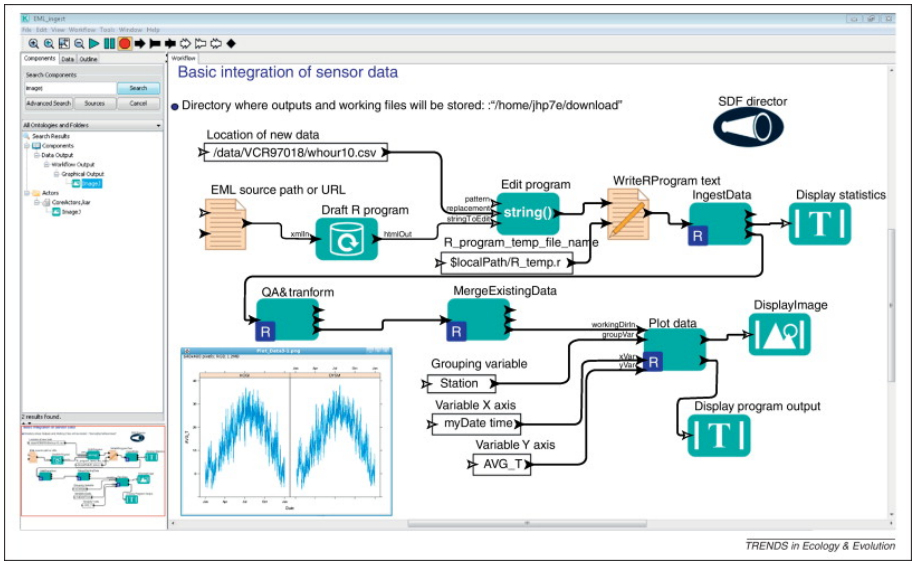

Figura 3.3: *Workflow* representado no Kepler. Fonte: [13].

3.2.4 Triana

O Triana [24][35] é um WfMS desenvolvido a partir de um projeto de pesquisa da Universidade *Cadiff* que combina uma interface gráfica de modelagem de *workflows* e ferramentas de análise e processamento de dados. Seu uso extende-se à vários domínios de pesquisa científica, como processamento de sinais, texto e imagem, geociências e bioinformática.

Seu código-fonte é inteiramente escrito em linguagem Java, e seus *workflows* possuem acesso a interfaces baseadas em *web-services*, e serviços de grids. Novos componentes, escrito em qualquer linguagem, podem ser adicionados a sua interface, bastando-se apenas a criação de uma classe com nome específico, e com definições de portas de entrada e saída. Os *workflows* são inteiramente codificados e armazenados em linguagem XML. Suporta *loops* em seus *workflows* (non-DAG)

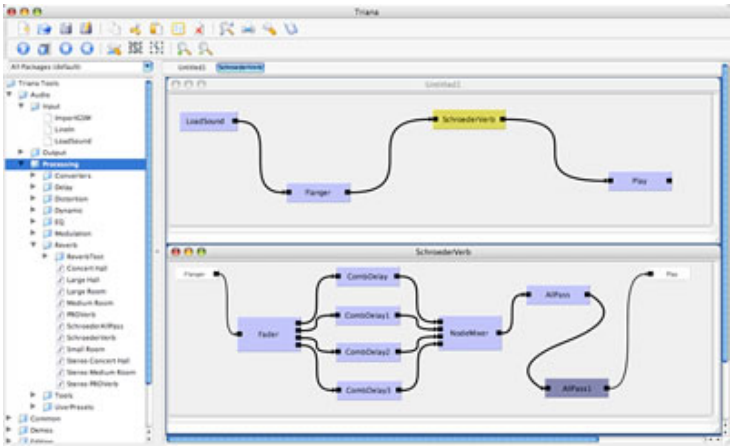

Figura 3.4: *Workflow* representado no Triana. Fonte: [24].

3.2.5 Watershed *Framework*

O *watershed framework* [37] é um ambiente de programação distribuída, que abstrai o *framework* open-MPI, para a execução de pipeline de aplicações em geral. Seus teste em ambientes de *cluster* demonstram uma capacidade de escalonamento e balanceamento de carga aplicável em ambientes de computação em nuvem. O trabalho proposto em [58] demonstra a possibilidade deste mesmo framework ser capaz de distribuir processos em ambientes de nuvem, como a Amazon EC2[1].

A possibilidade de seu uso como *engine* para um sistema gerenciador de *workflows* foi bastante discutida, uma vez que o mesmo atende várias requisitos especificados no modelo de referência para *Workflows*. A adaptação do *Watershed* em ambientes de Nuvens Computacionais é uma hipótese bastante razoável, haja vista que trabalhos sobre a execução de aplicações MPI em Cloud computing já é algo estudado por alguns grupos de pesquisa[58]. O grande desafio será utilizar o *framework* como uma nova estrutura para a implementação de *Engines* de Sistemas de Gerenciamentos de *Workflows* Científicos. Apesar do *framework* oferecer suporte a um arquivo de confíguração baseado em XML para a composição de seus serviços, o mesmo não segue nenhum padrão estabelecido de linguagem de descrição de *Workflows*. O desenvolvimento de uma interface amigável será necessário para a devida aplicação desta ferramenta no contexto de *workflows* científicos.

O *framework* watershed, foi testado em uma pequena rede doméstica, composta por dois computadores de porte simples, para uso pessoal. Duas aplicações, as quais já se encontravam como exemplo de uso da plataforma, foram testadas neste ambiente. A execução obteve os resultados esperados. Porém, ainda será necessária um processo de teste mais rigoroso, para o seu uso em ambientes de nuvem. O trabalho proposto em [58], abre a possibilidade do uso deste framework em nuvens computacionais. O framework não possui nenhuma interface gráfica de execução de tarefas. Sendo este trabalho realizado por meio de arquivos XML.

Não foi realizado nenhum experimento específico do domínio de bioinformática. Os dois experimentos realizados pertecem ao domínio da mineração de dados textuais. O framework não possui nenhuma interface gráfica, sendo que sua configuração é toda abstraida por uma interface XML.

3.2.6 Bioside

O Bioside [3] é um WfMS resultante de uma parceria entre a *Station Biologique de Roscoff(CNRS)*, e a *TELECOM Bretagne*, ambas instituições de pesquisa francesas. A ferramente objetiva auxiliar os biológos a executarem softwares de bioinformática e realizar experimentos *in silico* em geral (workflows). Utiliza uma linguagem baseada em XML para armazenamento dos *workflows*. Não suporta *workflows* com *loops*. A sua interface padrão, é otimizada para execução de *workflows* pertinentes ao domínio da Filogenia. Em especial, para o pacote PHYLIP[19].

A ferramenta adota um esquema de fluxo de dados simples, inteiramente baseado em armazenamento e leitura dos arquivos necessários em um diretório criado em uma área temporária no sistema do usuário. Cada *workflow* é constituído por um ou mais programas. A ferramente possui alguns pontos de extensão, podendo esta ser adaptada para diversos tipos de ambientes, como por exemplo, máquina local do usuário, clusters e

grids computacionais, e nuvens computacionais, que é o objetivo deste trabalho. A figura 3.5 ilustra um *workflow* gerado pela ferramenta:

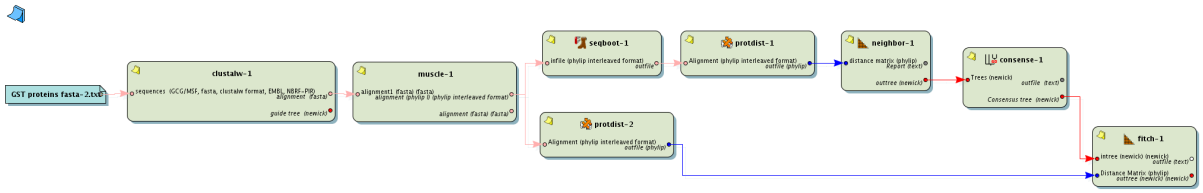

Figura 3.5: Esquema de Workflow gerado pela ferramenta Bioside.

O Bioside é uma ferramenta composta por três componentes. Em uma visão mais geral, eles podem ser divididos em sua *Engine* que é o pacote *Praxis*, a interface gráfica, ainda pertencente ao pacote *praxis*, e a interface Bioside-Biogenouest, que é basicamente uma camada de especialização de engine praxis para o domínio de bioinformática. A Figura 3.6 a seguir ilustra o relacionamento entre estes três componentes.

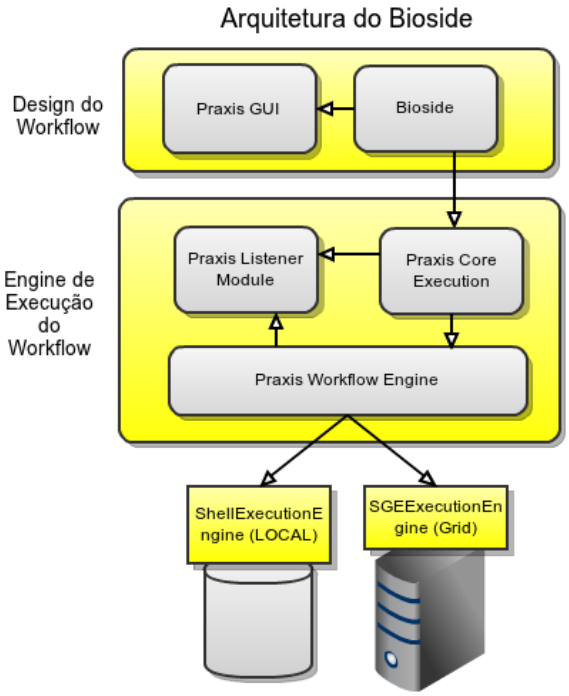

Figura 3.6: Componentes da arquitetura Bioside.

Na Figura 3.6 acima, os componentes ShellExecutionEngine e SGEExecutionEngine ilustrados, são interfaces para execução de *workflows* nos ambientes de máquina local e grid SGE, respectivamente. A engine de execução de *workflows* do Bioside, o pacote *praxis*, é composta por algumas classes de informações sobre o *workflow* em execução, e a classe ExecutionEngine que deverá ser adaptada ao ambiente fornecedor de serviços por especialização. A Figura 3.7 abaixo ilustra as classes mais importantes deste pacote:

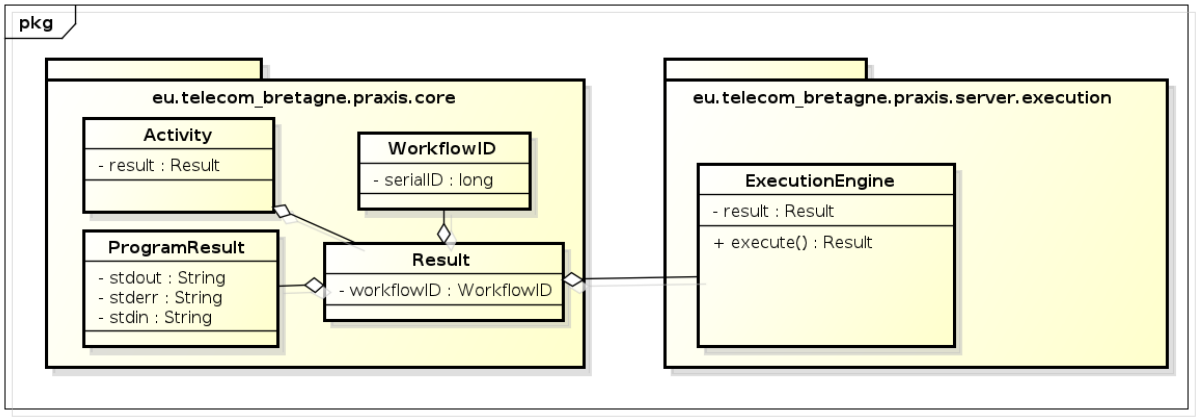

Figura 3.7: Principais componentes do pacote praxis do Bioside.

3.3 Análise dos WfMSs

Vários sistemas gerenciadores de *workflows* científicos foram pesquisados, tomando como base, os requisitos pertinentes ao domínio da bioinformática previamente levantados, baseando-se nos atributos da taxonomia. Em adição às caracteristicas analisadas na seção anterior, foram definidos a extensibilidade e usabilidade do sistema como atributos a serem analisados. A tabela a seguir, relaciona todas as tecnologias pesquisadas:

Após uma extensa análise, os melhores candidatos para aplicação em ambientes de nuvens computacionais foram o Galaxy e o Bioside. Entretanto, este último foi eleito como ferramenta de definição de *workflows* a ser utilizada no ambiente, por esta ferramenta ser inteiramente codificada em Java, a mesma linguagem utilizada para codificação do *framework* Bionimbus, e por ser suportada por uma grande comunidade de pesquisa.

WfMS	Modelagem do Workflow	Escalonamento das Tarefas	Tolerância a Falhas	Movimentação de Dados	Usabilidade	Extensibilidade
Galaxy[29]	DAG	Centralizada	A nível de Tarefas	Centralizada	Fácil uso, interface amigável ao usuário	Extensível através de módulos escritos em linguagem Perl
Taverna[49]	DAG	Centralizada	A nível de tarefas	Centralizada	Fácil uso, interface amigável ao usuário	Facilmente extensível, atráves de uma API Java.
Kepler [54]	non-DAG	Centralizada	A nível de Workflow	Mediada	Fácil uso, interface amigável ao usuário	Facilmente extensível, atráves de uma API Java.
Triana [35]	non-DAG	Descentralizado	Decidido pelo usuário	Peer-to-peer	Fácil uso, interface amigável ao usuário	Uma API Java permite a integração de novos componentes e plataformas
Watershed[37]	-	Descentralizado	-	Peer-to-peer	Uso não-trivial, mais estudos serão necessários	Não possui ponto de extensão claro. Deverá ser implmentado.
Bioside[3]	DAG	Centralizada	A nível de Tarefas	Centralizada	Fácil uso, interface amigável ao usuário	Possui uma engine facilmente modificável para diversos tipos de ambiente.

Tabela 3.1: Síntese dos resultados da pesquisa sobre os WfMSs

Capítulo 4

Proposta de Arquitetura do Ambiente e Protótipo Desenvolvido

Neste capítulo, é descrito brevemente, a proposta inicial da arquitetura de um ambiente de definição, modelagem e execução de *workflows* científicos. E logo após, e discutida a abordagem utilizada para a implantação do ambiente, que consiste em uma integração entre um sistema gerenciador de *workflows* já existente, o Bioside, uma vez que este sistema já possuia uma parte dos componentes previstos pela arquitetura, e o *framework* de federação de nuvens computacionais Bionimbus, que possui os módulos necessários para que a arquitetura fosse totalmente atendida. Portanto, é interessante destacar a observação de que todos os esforços foram concentrados para o desenvolvimento da camada de integração entre os dois ambientes.

4.1 Proposta de um Sistema Gerenciador de Workflows Científicos para Cloud Computing

Após um processo de análise de vários WfMSs, bem como o levantamento dos requisitos pertinentes ao ambiente de bioinformática, uma arquitetura foi modelada e proposta, de forma que os componentes do sistema encontram-se bem definidos e divididos, o que permite maior facilidade de modificações e extensão futuramente.

A figura 4.1 exibe graficamente o relacionamento entre os componentes da arquitetura proposta. Dessa forma, uma descrição dos componentes ilustrados na Figura 4.1 é exibida.

1. **Definição do *Workflow*:** este componente é o responsável por acomodar os componentes de definição do *workflow*, como a interface gráfica com o Usuário juntamente com a sua conversão em uma linguagem específica de *Workflows* [27] [26]. Este componente apoia o ciclo inicial de um *Workflow* Científico, que consiste em sua definição e mapeamento.
2. **Language *Parsing*:** este componente é responsável por receber uma descrição de *Workflow* em linguagem XML para *workflows* como YAWL [27], XPDL [26], e criar instâncias de *workflows* concretos a partir das descrições abstratas.

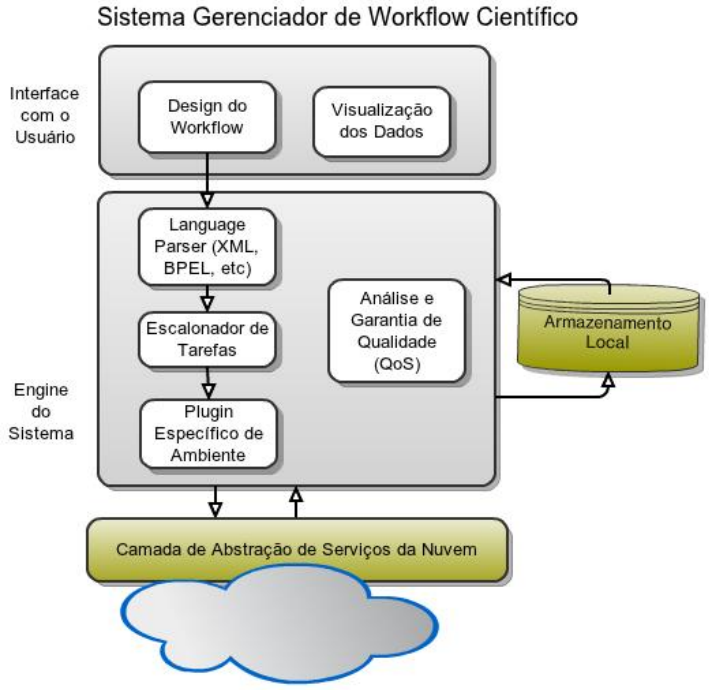

Figura 4.1: Proposta de Arquitetura para um WfMS.

3. **Plugin Específico de Ambiente:** este componente seria a conexão entre a *Engine* do sistema proposto e o ambiente de nuvem computacional. Por exemplo, poderiamos utilizar este componente para enviar e receber requisições da API da Amazon EC2 [1] para oferecer os serviços necessários. Este seria o único componente a sofrer variação caso o Sistema proposto fosse implantado em outro ambiente.

4. **Camada de Abstração de Serviços da Nuvem:** esta camada seria responsável por abstrair os serviços da nuvem. Trabalhos como o Globus Tookit[31] e as próprias classes de comunicação do Bionimbus [60] são componentes que enquadram-se neste componente arquitetural.

5. **Análise de Garantia de Qualidade:** seria responsável por oferecer funcionalidades de garantia de qualidade do serviço (QoS). Este módulo seria responsável pela geração de relatórios de monitoramento da execução do *workflow* de acordo com regras pre-definidas pelos usuários.

Existe um grupo de Casos de Uso que são considerados como essenciais para um sistema gerenciador de *workflows*. Segue uma breve descrição dos mesmos:

1. **Composição gráfica do Workflow:** O caso de Uso 'Modelar *Workflow*' agruparia as tarefas de projeto e submissão à *Workflow engine*, e talvez a tradução entre uma linguagem padrão de *Workflow* e uma linguagem especifica da *Engine*.

2. **Submeter *Workflow* em um repositório:** O *workflow* seria persistido em um Banco de Dados, para consultas futuras.

3. **Consultar *workflow* em um Repositório:** O *workflow* seria persistido em um Banco de Dados, para consultas futuras. Em um banco de dados local, para fins de provenança, ou em um repositório de escala geográfica, como o myExperiment [16].
4. **Verificar estado do *Workflow*:** O caso 'Verificar Estado *Workflow*' agruparia as funções de acompanhamento do estado da execução do *workflow*.
5. **Verificar Histórico de Execução:** Este caso de Uso, seria responsável pelo acompanhamento do histórico de execução do *workflow*. Refinamentos serão necessários.

A figura 4.2 abaixo ilustra graficamente, em um diagrama de casos de uso simples, todos os casos acima:

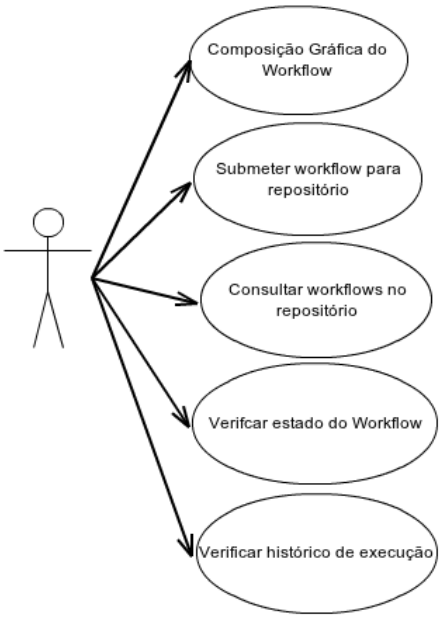

Figura 4.2: Casos de Uso do sistema.

4.2 Protótipo Desenvolvido

Para fins de prova de conceito da arquitetura proposta, foi desenvolvido um protótipo de um sistema gerenciador de *workflows* integrado com o *framework* Bionimbus, utilizando uma abordagem baseada na integração de componentes. O sistema gerenciador de *workflows* científicos Bioside [3] foi utilizado como interface de definição de *workflows* e sumissão de *jobs* para o plataforma Bionimbus[60]. Segue uma breve descrição das interfaces dos dois componentes, seguido da descrição de todo o esquema de integração das duas plataformas.

A Interface de Integração do Sistema Bioside

A engine de execução de *workflows* do Bioside, o pacote *praxis*, possui uma classe chamada `ExecutionEngine`, de maneira que a mesma precisa ser especializada por herança para fins de adaptação ao ambiente específico sobre o qual o sistema será implantado.

Portanto, como solução para o objetivo deste trabalho, que é a integração do sistema Bioside com o *framework* Bionimbus, foi criada uma classe denominada `BionimbusExecutionEngine`, que é a responsável pela tradução das requisições do Bioside para *jobs* do Bionimbus. A figura 4.3 baseada em UML demonstra o relacionamento entre as classes para tornar possível a integração.

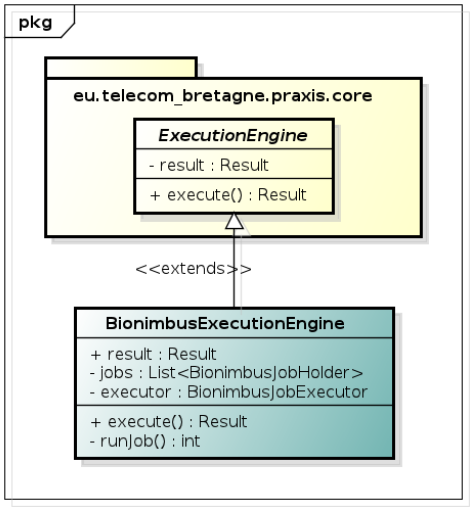

Figura 4.3: Relacionamento entre ExecutionEngine e BionimbusExecutionEngine

Basicamente, a classe **BionimbusExecutionEngine** é a classe que o sistema Bioside carregará em tempo de execução, como interface para o fornecedor de serviços, neste caso o Bionimbus. Como a classe é criada a partir da especialização de **ExecutionEngine**, então todos os atributos são herdados, e os métodos deverão ser escritos, uma vez que a classe **ExecutionEngine** é abstrata.

O método ExecutionEngine.execute() é o principal método utilizado para a comunicação com a plataforma da qual o Bioside consome os serviços. Como esta classe é abstrata, então o método BionimbusExecutionEngine.execute() deverá ser escrito, para lidar com as requisições do Bioside. A classe **Result** a principal classe de resultados de execução de cada programa de um *workflow* no Bioside. O método execute() deverá modificar instanciar o atributo de classe *result* do tipo **Result** e retorná-lo com as principais informações sobre a execução.

A Interface de Integração da Federação Bionimbus

Toda a interação com a federação Bionimbus se dará por meio de mensagens do tipo P2P, definidas pela própria arquitetura, exatamente como explicitado na seção 2.4. Para fins de simplificação da interação, foi desenvolvida uma classe denominada BionimbusUtils que é reponsável por abstrair as operações mais comuns, como o envio de arquivos, captura

de um ID de um arquivo específico, e execução de um job. A figura abaixo, baseada em notação UML, ilustra a estrutura da classe criada:

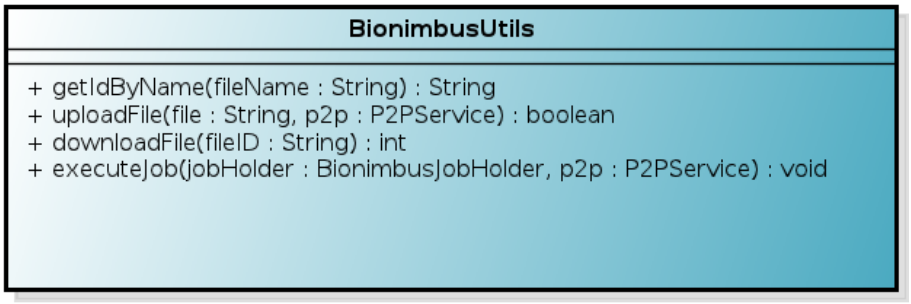

Figura 4.4: Classe BionimbusUtils que representa a camada de integração de serviços da nuvem.

A interação da classe BionimbusUtils com a federação é realizada através do envio de mensagens com o auxílio da classe P2PService, pertencente ao módulo de mensagens do Bionimbus, e essa interação é ilustrada na figura 4.7.

Esquema de integração das duas plataformas

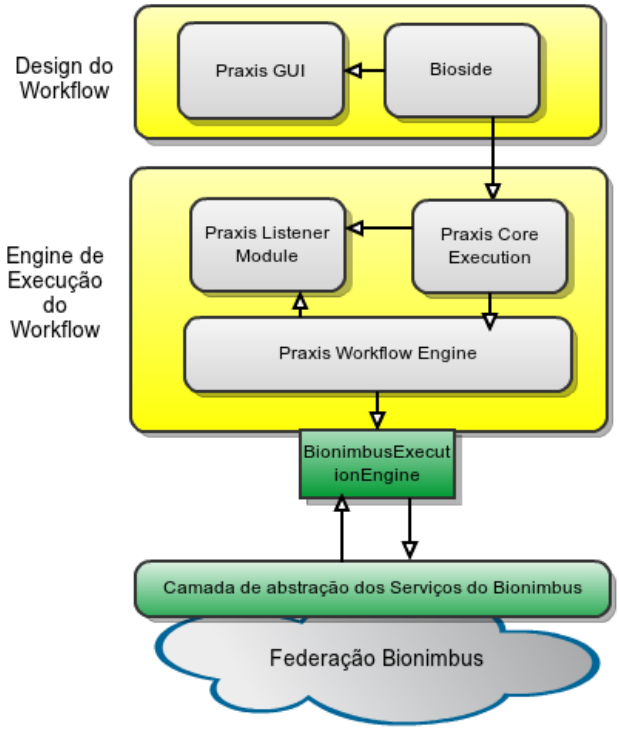

Figura 4.5: Protótipo desenvolvido como prova de conceito para arquitetura proposta.

A figura 4.5 demonstra graficamente o relacionamento entre as duas plataformas, o componente BionimbusExecutionEngine é implementado como uma classe com mesmo

nome no pacote de classes *praxis* do Bioside, é a principal classe de interação entre as duas plataformas. Esta classe faz a tradução entre os dados de programas do Bioside e converte-os em *jobs* do Bionimbus. A camada de abstração de serviços do Bionimbus, foi implementada para facilitar a interação deste com componentes externos.

Algumas classes foram criadas para fins de organização do código. A figura 4.6 abaixo, ilustra graficamente, o diagrama UML das classes criadas. Segue uma breve explicação das classes na figura 4.6:

- **BionimbusExecutionEngine:** esta é a principal classe da integração entre o framework Bioside e a plataforma Bionimbus. Esta classe é responsável por sobreescrever o método execute() da classe ExecutionEngine através de herança. Este método irá acionar todo o mecanismo de execução da tarefa na plataforma Bionimbus, e deverá retornar um objeto do tipo Result, do sistema Bioside, que é encarregado de armazenar informações sobre a execução do programa, ou neste caso, um *job* do Bionimbus.
- **BionimbusJobHolder:** esta classe é responsável por armazenar informações para a execução do *job* no Bionimbus. As principais informações são: A própria classe de informação **JobInfo** do Bionimbus, uma lista do tipo HashMap, que armazena todos os arquivos de entrada para o *job*, e outra lista do mesmo tipo, para armazenamento dos arquivos de saída do *job*.
- **BionimbusJobExecutor:** esta classe é a responsável pela conexão com a federação, juntamente com o método runJob(), para a execução do *job* no Bionimbus. Vários mecanismos de sincronização de tarefas entre as duas plataformas foram implementados em métodos desta classe.
- **BionimbusUtils:** esta classe é a única que interage diretamente com o Bionimbus, através de métodos de envio de mensagens, da classe P2PService. Esta classe é uma implementação da Camada de abstração de serviços da nuvem, como fora anteriormente descrito na seção 4.1.

No diagrama representado pela Figura 4.6 a seguir, estão representadas graficamente todas as classes implementadas para a integração entre as duas plataformas.

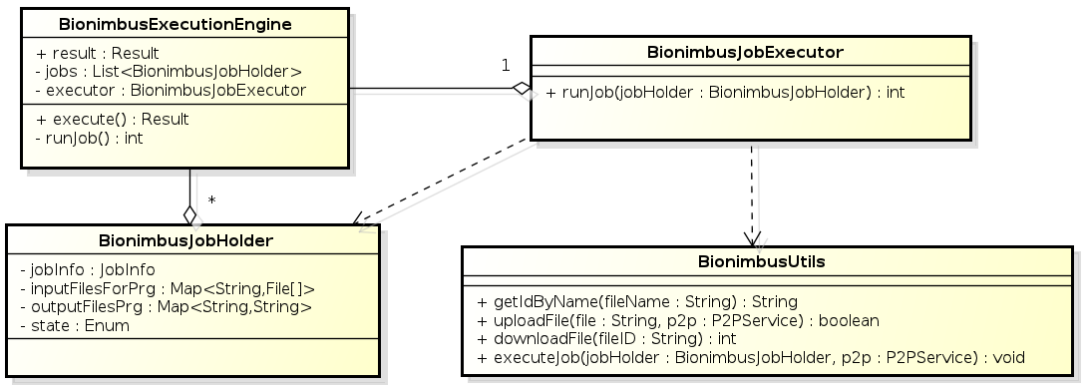

Figura 4.6: Diagrama de classes da camada de integração entre Bioside e Bionimbus.

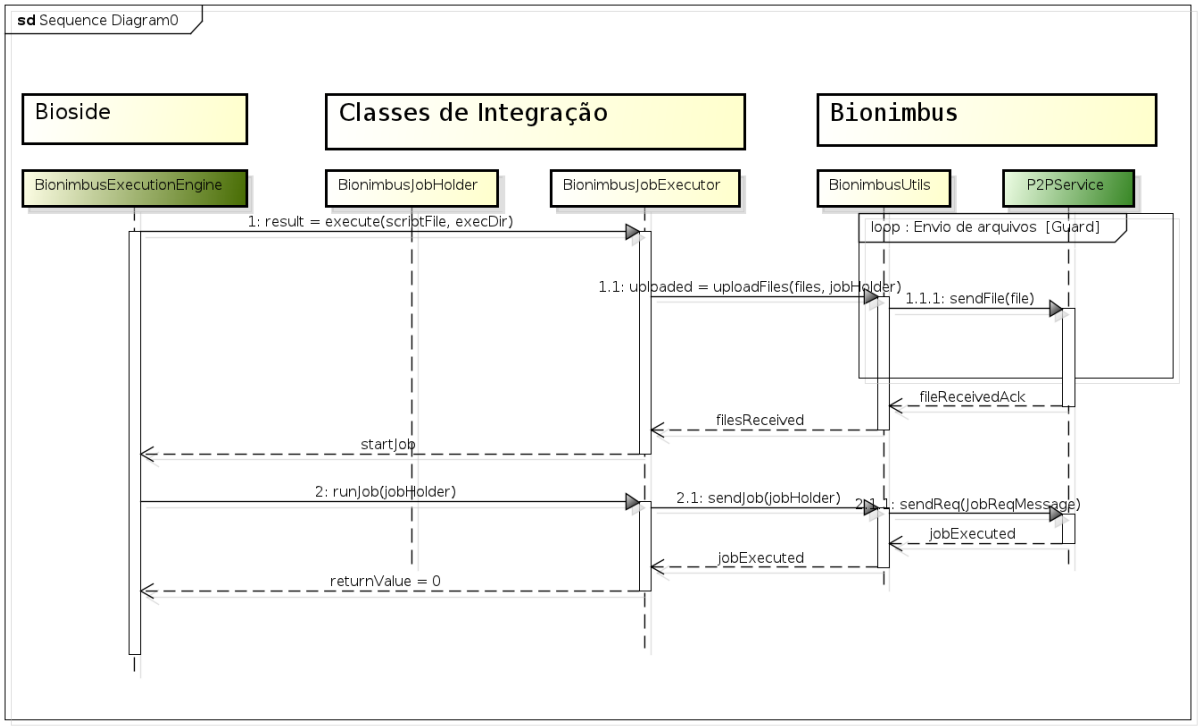

Figura 4.7: Relacionamento entre as classes do Bioside de Bionimbus durante e execução de um Job.

A Figura 4.7 abaixo, ilustra através de um diagrama de sequência, a interação entre estas classes durante a execução de um *job* no Bionimbus.

Para um maior detalhamento da implementação, o trecho de código Java abaixo , ilustra um trecho que representa o ponto inicial da integração entre o Bioside e o Bionimbus.

Dificuldades encontradas para integração

Algumas dificuldades foram encontradas durante o processo de desenvolvimento da camada de integração entre as duas plataformas, uma vez que as duas plataformas foram projetadas para atender ferramentas de diferentes naturezas, apesar das duas atenderem processos pertinentes ao domínio da Bioinformática. O Bionimbus, é uma plataforma projetada para a execução de ferramentas executáveis diretamente em linha de comando, enquanto o Bioside, possui uma interface otimizada para a execução de ferramentas do pacote *PHYLIP*[19], onde a interação com os softwares não é feita diretamente por linha de comando de maneira usual, mas sim, por um menu que depende da interação com o usuário. Desta maneira, um software deste pacote não pode receber os arquivos diretamente pela linha de comando, no momento da sua execução, como a maioria dos programas de Bioinformática, de forma que, para a automação destes programas em um *pipeline* de execução, ou em um sistema gerenciador de *workflows*, uma instância de um terminal deve ser criada com o *script* de execução como argumento para este terminal. O Bioside utiliza o terminal *TCSH*[23] para a execução de todas as suas ferramentas, juntamente com um *script* gerado em tempo de execução do *workflow*. Portanto, um dos problemas enfrentados, foi a adaptação deste paradigma à linguagem de execução desenvolvida para

Código Fonte 1 Principal trecho da classe BionimbusExecutionEngine

```
package eu.telecom_bretagne.praxis.server.execution.platform;

import eu.telecom_bretagne.praxis.core.execution.Result;

 public class BionimbusExecutionEngine extends ExecutionEngine {
 /*Atributos da classe e metodos utilitarios*/
 /**
 * Interface entre o Bioside e o Bionimbus, este metodo retorna um objeto do tipo
 * Result, classe do Bioside.
  **/
 public Result execute(File executionDirectory, File zippedInputFiles)
                        throws InterruptedException, InterruptedIOException {

       BionimbusJobHolder = new BionimbusJobHolder(new JobInfo());

       //... outras instrucoes do metodo.

       int exitValue;
       exitValue = bionimbus_execute(executionDirectory, zippedInputFiles, jobHolder);

        //se retorna zero, entao a execucao foi realizada com sucesso
        //e retorna o objeto result. Senao, retorna um objeto nulo, e o metodo lanca uma excecao.
        if(exitValue = 0)
                return result;
        else
            return null;
 }

 private int bionimbus_execute(String scriptFilePath, String executionDirectory, BionimbusJobHolder
      jobHolder) throws InterruptedException {
                /* ... preparacao do objeto jobHolder, com as informacoes
                                        necessarias para execucao do job. */
                int res = BionimbusJobExecutor.runJob(scriptFilePath, executionDirectory, jobHolder);
                return res;
        }
 }
```

o Bionimbus. A segunda dificuldade enfrentada, foi um conjunto de problemas diretamente relacionados com sincronização de processos entre as duas plataformas. Pois não existia um mecanismo confiável de execução dos *jobs* no Bionimbus, de maneira que não era trivial saber quando um *job* terminou a execução. Uma solução temporária, foi criar uma *thread* que verificava de tempos em tempos (geralmente na faixa de alguns segundos) se os arquivos de saída do *job* já existiam na máquina local do usuário, quando a verificação era positiva, o Bioside poderia prosseguir com sua execução normal. Portanto, para que o Bionimbus se torna uma plataforma mais robusta, mecanismos de aviso sobre recebimento de arquivos e *jobs* completos podem ser implementados.

4.3 Estudo de Caso

4.3.1 *Workflow* para análise Filogenética

Como estudo de caso utilizado como prova de conceito da arquitetura proposta, foi escolhido um *workflow* pertinente ao domínio da Filogenia. O *workflow* a seguir foi implementado no sistema Bioside. Ele é responsável por realizar análise filogênica de sequências de DNA obtidas diretamente por processos de sequenciamento. Vale lembrar que todas as ferramentas utilizadas no *workflow*, pertence ao pacote PHYLIP.

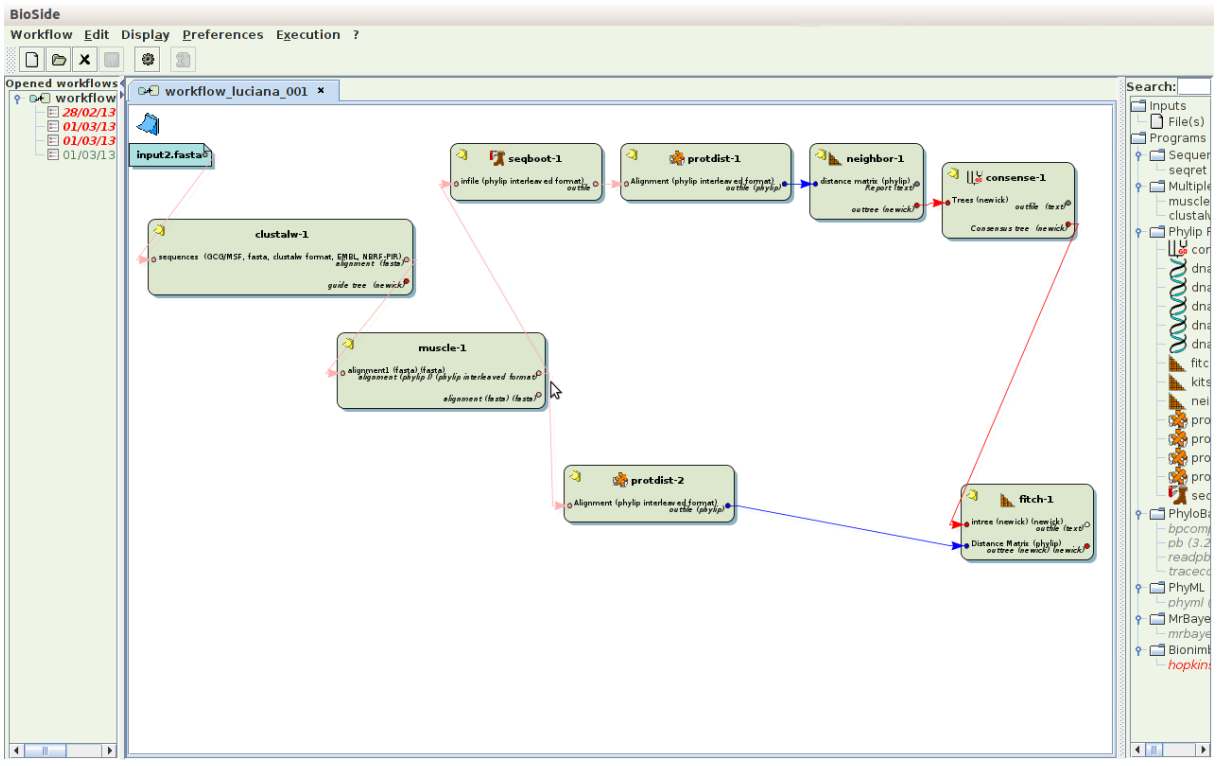

Figura 4.8: Workflow da figura 2.17 implementado no Bioside.

O pacote PHYLIP

O pacote PHYLIP é uma suite de ferramentas de Bioninformática, construídas para a execução de tarefas pertinentes à Análise Filogenética. A tabela a seguir, relaciona os programas utilizados no *workflow* deste estudo de caso, porém, uma relação completa dos mesmos pode ser obtida em [19].

Programa	Descrição
seqboot	Programa de Bootstrapping/Jackknifing. Lê um conjunto de dados, e produz vários conjuntos de dados a partir dele por reamostragem de bootstrap.
protdist	Método da distância de seqüências de proteínas que computa a medida de distância para sequências de proteínas
consense	Programa de árvores de consenso que calcula árvores de consenso pelo método de árvore com regra de maior consenso, que também permite facilmente encontrar a árvore de consenso estrito.
fitch	Método de matriz de distâncias de Fitch-Margoliash. Estima filogenias a partir de dados da matriz de distância no âmbito do "modelo de árvore de aditivo", segundo o qual as distâncias são esperadas a se igualar as somas de comprimentos dos ramos entre as espécies.
neighbor	Uma implementação do método Agrupamento de vizinhos (Neighbor-Joining) e do método UPGMA.

Além destes programas, foram utilizados o *MUSCLE*[15], um software utilizado para Alinhamento múltiplo de sequências de domínio público, usando sequências de nucleótidos ou proteínas, e o *ClustalW*[5], um software para alinhamento múltiplo de sequências, com interação via linha de comando.

4.3.2 Descrição do Experimento

Ambiente utilizado

Para fins de teste do protótipo desenvolvido, foi utilizada uma pequena rede doméstica, constituída por apenas dois computadores pessoais de porte médio. A seguir uma breve descrição sobre os mesmos.

- Uma máquina com processador *Intel Core 2 Duo* com processador de 2 núcleos de 2.66Ghz cada, memória RAM de 4Gb e sistema Operacional Ubuntu Linux 11.10. Esta é a máquina na qual o Bioside se encontrava instalado, e o Bionimbus rodava no modo mestre.
- Outra máquina com processador *Intel Core 2 Extreme* de 2 núcleos com 3.2Ghz cada e 8Gb de memória RAM. Esta máquina possuia todos os serviços instalados em sua estrutura local, bem como todo o armazenamento de arquivos foi delegado pra esta máquina.

Embora esta infra-estrutura não seja sequer algo próximo de um ambiente de Nuvem Computacional real, as capacidades de federação de serviços do Bionimbus puderam ser exploradas neste ambiente.

Execução do experimento

O mesmo *workflow* foi executado em dois momentos, em duas condições diferentes:

1. O *workflow* foi executado localmente, na primeira máquina descrita anteriormente sem o uso do protótipo desenvolvido neste trabalho, utilizando a própria **ShellExecutionEngine** já disponível no sistema Bioside.
2. Neste momento, o *workflow* foi submetido para executar sobre a estrutura do Bionimbus, utilizando as duas máquinas em um esquema de federação, através do protótipo desenvolvido.

Foram exercitados os casos de uso de upload de arquivos, download de arquivos e submissão de *jobs* para a federação. Para a execução do *worflow*, foi escolhido como entrada, um arquivo no formato *FASTA* em sua forma mais simples, com um pouco mais de 1MB de tamanho. Este arquivo não representa uma amostra de um típico arquivo FASTA utilizado em experimentos reais em ambientes reais, pois a infra-estrutura disponível para testes não era o suficientemente robusta para executar o *workflow* sobre arquivos provenientes de ambientes reais em um tempo computacional aceitável.

Resultados e discussão

O *workflow* descrito nesta seção foi executado nos dois ambientes anteriormentes descritos. A tabela 4.1 abaixo, relaciona os ambientes, com os tempos de execução obtidos:

Tabela 4.1: Relação dos resultados obtidos

Ambiente	**Tempo de execução obtido**
Máquina Local (1)	00:24:13
Ambiente Federado (2)	00:26:25

A execução do experimento sobre o ambiente Bionimbus mostrou-se algo viável a partir dos experimentos. Os tempos de execução mostraram-se bastante próximos, porém com algum acréscimo no segundo caso. Este acréscimo deve-se ao tempo de tradução da linguagem de *workflows* do Bioside para *jobs* do Bionimbus, e a transferência do arquivo de entrada de uma máquina para a outra e reconhecimento do mesmo pela federação. As duas máquinas apresentavam capacidade de processamento bastante próximas, portanto, o tempo de execução foi o esperado.

Capítulo 5

Conclusão e Trabalhos futuros

Este trabalho destinou-se a especificar um ambiente no qual usuários não cientistas da computação pudessem executar *workflows* científicos dos mais variados domínios em uma estrutura de *Cloud Computing*, embora o foco em Bioinformática tenha se tornado uma necessidade durante o processo de desenvolvimento deste trabalho.

Ainda que a integração entre sistemas gerenciadores de *workflows* e Nuvens Computacionais sejam um tendência recente na indústria e academia, vários trabalhos serviram uma base que sem a qual este trabalho não seria possível.

O Bionimbus mostrou-se uma arquitetura que, apesar de se encontrar em seu estágio inicial de desenvolvimento, oferece uma estrutura flexível que o permite adaptar-se à diversos tipo de ambiente, até mesmo aqueles não relacionados diretamente com Nuvens Computacionais.

A integração entre o sistema gerenciador de *workflows* Bioside e o sistema Bionimbus mostrou-se algo possível, de maneira que a o sistema Bioside é executado de maneira transparente sobre a plataforma Bionimbus, consumindo seus serviços sobre demanda, como é definido no paradigma da computação em nuvem. Entretanto, apesar da integração ter sido realizada, várias dificuldade oriundas de sincronização de processos entre os dois ambientes foram encontradas. Portanto, como trabalhos futuros, o Bioninbus necessita dos seguintes mecanismos de aviso de recebimento de arquivo e aviso de completude de um *job*.

Questões importantes como confiabilidade do sistema em outros ambientes, tolerância a variados tipo de falha e até mesmo a segurança do ambiente não foram abordadas, embora as mesmas não fossem o foco deste trabalho.

A criação de uma interface padrão baseada em padrões abertos como *web-services* para a plataforma Bionimbus é um questão a ser trabalhada, caso a necessidade de consumo de seus serviços por outros sistemas gerenciadores de workflows torne-se uma necessidade futura.

Referências

[1] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/pt/ec2/. Acesso em: 19/02/2013. 12, 36, 41

[2] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/pt/s2/. Acesso em: 19/02/2013. 12

[3] Bioside : a tool for biologists performing, learning, replicating and storing complex bio-analys. http://www.bioside.org/. Acesso em: 20/02/2013. 30, 33, 36, 39, 42

[4] Business process model and notation. http://www.bpmn.org/. Acesso em: 19/02/2013. 4

[5] Clustal: Multiple sequence alignment. http://www.clustal.org/. Acesso em: 24/02/2013. 49

[6] Everything as a service. http://www.hp.com/hpinfo/initiatives/eaas/index.html. Acesso em: 19/02/2013. 12

[7] The evolution of the cloud computing market. http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market/. Acesso em: 19/02/2013. 14

[8] The galaxy project. http://galaxyproject.org/. Acesso em: 23/02/2013. vii, 33

[9] Gogrid cloud hosting. http://www.gogrid.com/. Acesso em: 19/02/2013. 12

[10] Google app engine. https://developers.google.com/appengine/. Acesso em: 19/02/2013. 13, 14

[11] Heroku cloud application platform. http://www.heroku.com/. Acesso em: 19/02/2013. 13

[12] Illumina sequencing. http://www.illumina.com/. Acesso em: 22/02/2013. 28, 29

[13] The kepler project. https://kepler-project.org/. Acesso em: 20/02/2013. vii, 34, 35

[14] Microsoft azure. http://www.windowsazure.com/. Acesso em: 19/02/2013. 14

[15] Muscle : Popular multiple alignment software. http://www.drive5.com/muscle/. Acesso em: 24/02/2013. 49

[16] myexperiment. http://www.myexperiment.org/. Acesso em: 20/02/2013. 33, 34, 42

[17] mygrid group. http://www.mygrid.org.uk/. Acesso em: 20/02/2013. 34

[18] Open provenance model. http://openprovenance.org/. Acesso em: 20/02/2013. 10

[19] Phylip package. http://evolution.genetics.washington.edu/phylip.html. Acesso em: 23/02/2013. 30, 36, 46, 49

[20] Phylogeny definition. http://www.biology-online.org/dictionary/Phylogeny. Acesso em: 23/02/2013. 28

[21] Roche 454 sequencing. http://454.com/. Acesso em: 22/02/2013. 28

[22] A síntese protéica. http://www.ufpe.br/biolmol/Genetica-Medicina/sintese_proteica.htm. Acesso em: 24/02/2013. vii, 25

[23] Tcsh shell. http://www.tcsh.org/. Acesso em: 24/02/2013. 46

[24] Triana : The open source problem solving environment. http://www.trianacode.org/. Acesso em: 24/02/2013. vii, 35

[25] Workflow management coallition. http://www.wfmc.org/. Acesso em: 19/02/2013. 4

[26] Xml process definition language (xpdl). http://www.xpdl.org/. Acesso em: 19/02/2013. 4, 40

[27] Yawl: Yet another workflow language. http://yawlfoundation.org/. Acesso em: 19/02/2013. 4, 40

[28] Mohamed Abouelhoda, Shadi Issa, and Moustafa Ghanem. Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support. *BMC Bioinformatics*, 13(1):77+, May 2012. 33, 34

[29] Enis Afgan, Jeremy Goecks, Dannon Baker, Nate Coraor, Anton Nekrutenko, and James Taylor. Galaxy: A Gateway to Tools in e-Science. In K. Yang, editor, *Guide to e-Science: Next Generation Scientific Research and Discovery*, page 35. Springer, 2011. 33, 39

[30] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In *Proceedings of the 2006 international conference on Provenance and Annotation of Data*, IPAW'06, pages 118–132, Berlin, Heidelberg, 2006. Springer-Verlag. 10

[31] Rajkumar Buyya, Suraj Pandey, and Christian Vecchiola. Cloudbus toolkit for market-oriented cloud computing. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 24–44, Berlin, Heidelberg, 2009. Springer-Verlag. 41

[32] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Intercloud: utility-oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 13–31, Berlin, Heidelberg, 2010. Springer-Verlag. vii, 14, 15

[33] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 337 –345, july 2010. 14

[34] Yizeng Chen, Xingui Li, and Fangning Chen. Overview and analysis of cloud computing research and application. In *E -Business and E -Government (ICEE), 2011 International Conference on*, pages 1 –4, may 2011. 12

[35] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming scientific and distributed workflow with triana services: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1021–1037, August 2006. 35, 39

[36] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1345–1350, New York, NY, USA, 2008. ACM. 10

[37] T.L.A. de Souza Ramos, R.S. Oliveira, A.P. de Carvalho, R.A.C. Ferreira, and W. Meira. Watershed: A high performance distributed stream processing system. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2011 23rd International Symposium on*, pages 191 –198, oct. 2011. 33, 36, 39

[38] Sergei A. Filichkin, Henry D. Priest, Scott A. Givan, Rongkun Shen, Douglas W. Bryant, Samuel E. Fox, Weng-Keen Wong, and Todd C. Mockler. Genome-wide mapping of alternative splicing in arabidopsis thaliana. *Genome Research*, 20(1):45–58, 2010. 28

[39] I. Foster, Yong Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, nov. 2008. 2, 11

[40] Ewa Deelman Gideon Juve. Scientific workflows and clouds. *ACM Crossroads*, 2010. 9

[41] Yolanda Gil, Pedro A. González-Calero, and Ewa Deelman. On the black art of designing computational workflows. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, WORKS '07, pages 53–62, New York, NY, USA, 2007. ACM. 9

[42] Carole Anne Goble and David Charles De Roure. myexperiment: social networking for workflow-using e-scientists. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, WORKS '07, pages 1–2, New York, NY, USA, 2007. ACM. 10, 34

[43] Luciana S. A. Gomes. Proveniência para workflows de bioinformática. Mestrado, Pontíficia Universidade Católica do Rio de Janeiro, Centro Técnico e Científico, PUC-Rio, 2011. vii, 30

[44] T. Grandison, E.M. Maximilien, S. Thorpe, and A. Alba. Towards a formal definition of a computing cloud. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 191 –192, july 2010. 12

[45] R.L. Grossman. The case for cloud computing. *IT Professional*, 11(2):23 –27, march-april 2009. 12

[46] C.N. Hoefer and G. Karagiannis. Taxonomy of cloud computing services. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1345 –1350, dec. 2010. 13

[47] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the use of cloud computing for scientific workflows. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 640 –645, dec. 2008. 2

[48] David Hollingsworth. *The Workflow Reference Model*. Number TC00-1003. Person Education do Brasil, Hampshire, UK, 1995. vii, 1, 3, 4, 5, 6, 7, 8

[49] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(suppl 2):W729–W732, 1 July 2006. vii, 34, 39

[50] J. Wiglarz J. Nabrzyski, J.M. Schopf. *Grid resource management: state of the art and future trends*. Springer, 2003. 9

[51] Joao Meidanis Joao Carlos Setubal. *Introduction to Computational Molecular Biology*. Brooks/Cole, 1997. ix, 22, 27

[52] Gideon Juve and Ewa Deelman. Scientific workflows and clouds. *Crossroads*, 16(3):14–18, March 2010. 2

[53] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai, and Marcel Kunze. Cloud Federation. In *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*. IARIA, September 2011. 14

[54] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, August 2006. 9, 34, 39

[55] Bertram Ludäscher, Mathias Weske, Timothy Mcphillips, and Shawn Bowers. Scientific workflows: Business as usual? In *Proceedings of the 7th International Conference on Business Process Management*, 2009. vii, 9, 10, 11

[56] K.H. Masiyev, I. Qasymov, V. Bakhishova, and M. Bahri. Cloud computing for business. In *Application of Information and Communication Technologies (AICT), 2012 6th International Conference on*, pages 1 –4, oct. 2012. 12

[57] Timothy McPhillips, Shawn Bowers, Daniel Zinn, and Bertram Ludäscher. Scientific workflow design for mere mortals. *Future Gener. Comput. Syst.*, 25(5):541–551, May 2009. 9

[58] Ying Peng and Fang Wang. Cloud computing model based on mpi and openmp. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 7, pages V7–85 –V7–87, april 2010. 36

[59] Deana D. Pennington. Supporting large-scale science with workflows. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, WORKS '07, pages 45–52, New York, NY, USA, 2007. ACM. 9

[60] Hugo Saldanha, Edward Ribeiro, Maristela Holanda, Aleteia Araujo, Genaina Rodrigues, Maria Emilia Telles Walter, João Carlos Setubal, and Alberto M. R. Dávila. A cloud architecture for bioinformatics workflows. In *CLOSER'11*, pages 477–483, 2011. vii, 16, 17, 19, 20, 21, 22, 28, 29, 33, 41, 42

[61] Hugo V. Saldanha. Bionimbus: uma arquitetura de federação de nuvens computacionais híbridas para execução de workflows de bioinformática. Mestrado, Instituto de Ciências Exatas, Departamento de Ciência da Computação, UnB, 2012. vii, 29

[62] Cavalli L. L. Sforza and A. W. F. Edwards. Phylogenetic analysis: Models and estimation procedures. *American Journal of Human genetics*, 19:223–257, 1967. 28

[63] Yogesh Simmhan and Roger Barga. Analysis of approaches for supporting the open provenance model: A case study of the trident workflow workbench. *Future Gener. Comput. Syst.*, 27(6):790–796, June 2011. 10

[64] Marc Sultan, Marcel H. Schulz, Hugues Richard, Alon Magen, Andreas Klingenhoff, Matthias Scherf, Martin Seifert, Tatjana Borodina, Aleksey Soldatov, Dmitri Parkhomchuk, Dominic Schmidt, Sean O'Keeffe, Stefan Haas, Martin Vingron, Hans Lehrach, and Marie-Laure Yaspo. A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Human Transcriptome. *Science*, 321(5891):956–960, August 2008. 28

[65] B. Ludascher I. Altintas C. Berkley D. Higgins E. Jaeger M. Jones E.A. Lee J. Tao and Y. Zhao. Scientific workflow management and the kepler system. concurrency and computation: Practice and experience. pages 18(10):1039–1065, 2006. 9

[66] Dong Xu. Cloud computing: An emerging technology. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 1, pages V1–100 –V1–104, june 2010. 12

[67] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3), September 2005. 9, 32, 33