COPPE/UFRJ

GERÊNCIA DA EXECUÇÃO DE WORKFLOWS CIENTÍFICOS DE BIOINFORMÁTICA EM AMBIENTES DISTRIBUÍDOS

Patrícia Machado de Barros

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador(es): Marta Lima de Queirós Mattoso

Paulo Mascarello Bisch

Rio de Janeiro

Março de 2009

GERÊNCIA DA EXECUÇÃO DE WORKFLOWS CIENTÍFICOS DE BIOINFORMÁTICA EM AMBIENTES DISTRIBUÍDOS

Patrícia Machado de Barros

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Marta Lima de Queirós Mattoso, D.Sc.

Prof. Paulo Mascarello Bisch, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Profa. Maria Cláudia Reis Cavalcanti, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2009

Barros, Patrícia Machado de

Gerência da Execução de Workflows Científicos de Bioinformática em Ambientes Distribuídos/ Patrícia Machado de Barros. - Rio de Janeiro: UFRJ/COPPE, 2009.

XIII, 91 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Paulo Mascarello Bisch

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2009.

Referências Bibliográficas: p. 85 - 91.

1. Bioinformática. 2. Workflows Científicos. 3. Ambientes Distribuídos. 4. Proveniência de Dados. I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A meus pais.

"Sempre que buscamos novos desafios, a vida nos reserva alguns obstáculos para alcançá-los.

O importante é perseguir o desejo de ser feliz e alcançar seu sonho, lembrando que você não está só em sua luta.

Às vezes o pensamento em desistir, nos rodeia, mas a força para prosseguir e a vontade de realizar seus objetivos é maior e você vence a batalha.

No final, resta a sensação de dever cumprido e o amor das pessoas que caminharam ao seu lado nesta grande etapa da vida.

E você tem a certeza de que tudo vale a pena".

Marcela Rocha

AGRADECIMENTOS

Inicialmente, agradeço aos meus orientadores, Professora Marta Mattoso, pela orientação, confiança, atenção e idéias indispensáveis para a conclusão desta dissertação e, Professor Paulo Bisch, pela orientação, confiança, atenção e pela oportunidade ímpar em fazer parte do seu grupo de trabalho no Instituto de Biofísica Carlos Chagas Filho (IBCCF/UFRJ). Obrigada também por me permitir a utilização de toda a infra-estrutura computacional necessária para a realização dos experimentos realizados nesta dissertação.

Aos meus pais, Luiz e Magda, e minha irmã, Carine, pelo exemplo, amor, dedicação, ensinamentos e incentivos de sempre.

Ao Emerson, pelo companheirismo, paciência e compreensão nos momentos em que não pude lhe dar a devida atenção, pelas palavras de incentivo de sempre e por todo seu carinho.

Ao meu afilhado Artur, pelo carinho e pelos momentos de alegria.

Agradeço também as amigas Regina Soares e Marcela Rocha pelas suas amizades e companheirismo.

À amiga Melissa Paes, pela amizade, companheirismo, por todos os momentos em que passamos juntas durante o curso de mestrado e pelas palavras de incentivo.

Ao Sergio Serra, pelo seu apoio, interesse no meu trabalho e pelas idéias surgidas ao longo de nossas reuniões. Muita obrigada também pela paciência nos dias em que dizia que eu estava um pouco estressada.

A todos os colegas do Lab FISBIO (Laboratório de Física Biológica – IBCCF/UFRJ), em especial aos amigos, Manuela Leal, Mainá Bitar e Daniel Costa, por seus ensinamentos técnicos na área biológica e muito valiosos para a conclusão desta dissertação, por nossa amizade, incentivos e todos os momentos de descontração. E ao Jeferson Rodrigues, pela sua colaboração nas atividades diárias do laboratório.

Aos colegas do LMDM (Laboratório de Modelagem e Dinâmica Molecular – IBCCF/UFRJ), pela atenção e disponibilidade em me passar seus conhecimentos em

dinâmica molecular e sempre solícitos em tirar as dúvidas que apareceram durante a pesquisa.

À Professora Carla Osthoff, pelo incentivo ao meu ingresso no curso de mestrado e por me orientar nos primeiros passos da pesquisa científica.

Aos Professores Maria Cláudia Cavalcanti e Geraldo Xexéo pela participação na minha banca de avaliação.

A todas as pessoas que de alguma maneira fizeram parte desta jornada.

Agradeço a Deus pela realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

GERÊNCIA DA EXECUÇÃO DE WORKFLOWS CIENTÍFICOS DE BIOINFORMÁTICA EM AMBIENTES DISTRIBUÍDOS

Patrícia Machado de Barros

Março/2009

Orientadores: Marta Lima de Queirós Mattoso
Paulo Mascarello Bisch

Programa: Engenharia de Sistemas e Computação

Pesquisadores de bioinformática vêm utilizando sistemas de gerência de *workflows* científicos (SGWfC) para ajudar na realização de experimentos envolvendo simulações de dinâmica molecular. Estas simulações demandam grande poder computacional e se beneficiam de clusters de PC e grades computacionais. A gerência de *workflows* científicos envolvendo execução de programas em ambientes distribuídos é considerada uma tarefa complexa. Para continuar o controle da execução nos ambientes distribuídos e capturar os dados de proveniência é preciso adotar diferentes mecanismos, que aumentam ainda mais a complexidade dessa gerência. Esta dissertação propõe uma arquitetura para auxiliar os SGWfC na gerência da execução de *workflows* científicos em ambientes distribuídos, tendo como funcionalidades a definição e o controle da execução remota dos *workflows* em clusters de PC, além de capturar os dados de proveniência relevantes ao experimento e armazená-los em um repositório de dados. Esta arquitetura visa diminuir a complexidade da execução paralela de programas definidos ao longo de *workflows*. Foi desenvolvido também um portal *web*, GromDExp, com o objetivo de oferecer os serviços desenvolvidos na arquitetura de forma transparente ao usuário. Para validar a arquitetura proposta foi definido um *workflow* para realização de um experimento real de simulação de dinâmica molecular, denominado GromDFlow. O GromDFlow foi modelado no SGWfC Kepler e a dinâmica molecular foi executada de modo remoto no cluster de PC do IBCCF/UFRJ. O experimento evidenciou as vantagens da solução apresentada que está disponível e vem sendo usada para diversas simulações.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DISTRIBUTED EXECUTION MANAGEMENT IN BIOINFORMATICS SCIENTIFIC WORKFLOWS

Patrícia Machado de Barros

March/2009

Advisors: Marta Lima de Queirós Mattoso
Paulo Mascarello Bisch

Department: Computer Science and Systems Engineering

Bioinformatics researchers have been using scientific workflow management systems (SWfMS) to help their molecular dynamics simulation experiments. These simulations are computationally intensive demanding clusters of PC and computational grid environments. Managing scientific workflows with parallel program execution in distributed environments is a complex task. Keeping track of the remote parallel execution and capturing provenance data among distributed resources is even more complex. This dissertation proposes an architecture to help executing scientific workflows in distributed environments. It provides for definition and remote execution control of workflows involving parallel execution on PC clusters and grids. In addition, it captures and stores provenance data of the workflow execution in a repository. The goal of this architecture is to lower the complexity of parallel execution of some programs of the workflow. GromDExp, a web portal, has also been developed aiming to provide architectural services to end users in a transparent manner. To validate the proposed architecture, a workflow has been defined for the execution of a real molecular dynamics simulation experiment, called GromDFlow, modelled on the Kepler SWfMS. The simulation has been executed remotely on the PC cluster of IBCCF/UFRJ. The experiment highlighted the advantages of the proposed architecture, which is now available and is being used for several simulations.

ÍNDICE

LISTA DE SIGLAS

DM	Dinâmica Molecular
EELA	E-Infrastructure shared between Europe and Latin America
GPL	General Public License
GROMACS	Groningen Machine for Chemical Simulation
GROMOS	Groningen Molecular Simulation
MPI	Message Passing Interface
PAD	Processamento de Alto Desempenho
PDB	Protein Data Bank
PBS	Portable Batch System
SGBD	Sistema Gerenciador de Banco de Dados
SGWf	Sistema de Gerência de Workflow
SGWfC	Sistema de Gerência de Workflow Científico
SSH	Secure Shell

Capítulo 1

1 Introdução

1.1 Motivação

Durante a década de noventa, com o surgimento dos seqüenciadores automáticos de DNA, surgiu também a necessidade de processamento e armazenamento de uma vasta quantidade de seqüência, e, com isso, exigiu-se recursos computacionais cada vez mais eficientes. Entretanto, paralelamente ao armazenamento, existia também a necessidade de análise dos dados armazenados, o que tornava indispensável à utilização de recursos computacionais eficientes para a análise dos resultados obtidos. Assim nascia a Bioinformática (PROSDOCIMI *et al.*, 2003). A bioinformática é uma área interdisciplinar que envolve a união de diversas linhas de conhecimento, como a biologia, a estatística, a matemática, a física, a química e a computação.

Atualmente, os projetos de bioinformática vêm tratando um grande desafio. O uso das tecnologias e dos recursos computacionais existentes permite que diversas análises biológicas sejam feitas, muitas destas impossíveis ou demasiadamente longas para serem realizadas sem o apoio computacional de hoje.

Os pesquisadores de bioinformática estão cada vez mais utilizando recursos computacionais sofisticados para execução e análise de seus experimentos. Estes experimentos são chamados de experimentos *in silico*, que podem ser definidos como experimentos científicos que utilizam recursos computacionais, em contraposição aos experimentos *in vitro* de bancada. Geralmente, os programas computacionais dos experimentos *in silico* são executados de maneira encadeada, onde os dados de saída produzidos em cada etapa do experimento podem ser utilizados como entrada para a próxima etapa a ser executada.

Existem diversos métodos que podem ser empregados dentro da bioinformática para a geração e análise de dados biológicos, dentre eles, podemos citar, o alinhamento de seqüência, as anotações gênicas e a modelagem molecular (PROSDOCIMI *et al.*,

2003). Através da modelagem molecular podemos investigar as características que determinam a estrutura de uma proteína e, conseqüentemente, estudar sua função. Uma das técnicas aplicadas à modelagem molecular é a simulação de dinâmica molecular.

A dinâmica molecular (DM) é uma técnica que consiste em estudar a evolução temporal de um dado sistema molecular, constituído de muitos átomos e moléculas. As configurações moleculares sucessivas são geradas através das equações de movimento de Newton (MUNDIM, 2002).

Em geral, os experimentos de dinâmica molecular são compostos pela execução de uma série de tarefas encadeadas que são realizadas manualmente pelos pesquisadores, no entanto, o uso de uma ferramenta, como *workflows*, pode auxiliar os pesquisadores de bioinformática a automatizar o processo de criação e modelagem de seus experimentos. Um *workflow* é uma coleção de tarefas organizadas para realizar um determinado processo (VAN DER AALST *et al.*, 2003). Através do *workflow* é definida a ordem de execução das tarefas e o fluxo das informações pertencentes ao processo a ser realizado. O termo *workflow* científico vem sendo empregado aos *workflows* que representam um experimento científico.

Podemos considerar um *workflow* científico de dinâmica molecular como uma aplicação que requer uma grande quantidade de recursos computacionais para sua execução. Algumas etapas são relativamente rápidas enquanto outras consomem muito tempo de processamento, geralmente, as execuções levam dias, necessitando de um ambiente distribuído com capacidade de processamento de alto desempenho (PAD), como um cluster de PC ou uma grade computacional[1]. Apenas as tarefas do *workflow* que demandam muito tempo de processamento precisam ser executadas em um ambiente PAD. Isso cria um cenário distribuído e heterogêneo para a execução dos *workflows*.

Durante a definição e execução dos *workflows* científicos é desejável registrar a proveniência dos dados (BUNEMAN *et al.*, 2001), ou seja, todas as informações referentes ao experimento. Os dados devem ser armazenados, como os utilizados na definição do *workflow* (quem executou, quando executou, quais os parâmetros e quais os arquivos de entrada foram utilizados), assim como os dados gerados durante a

[1] Conhecido na literatura como *grid computing* (FOSTER e KESSELMAN, 1999).

execução das tarefas remotas do *workflow* no ambiente distribuído. A captura da proveniência dos dados é essencial para a validação do estudo e análise de cada experimento.

Para atender suas necessidades, os pesquisadores de bioinformática podem beneficiar-se do uso de sistemas de gerência de *workflows* científicos (SGWfC). Um SGWfC pode ser definido como um sistema para a definição, criação e gerência da execução de fluxos de tarefas (*workflows*) através do uso de software (WFMC, 2008).

Os SGWfC como o Kepler (KEPLER, 2008), o VisTrails (VISTRAILS, 2008) e o Taverna (MYGRID, 2008) constituem uma importante base em termos de funcionalidade, auxiliando os pesquisadores de bioinformática na definição e automação de seus experimentos. Cada um dos SGWfC apresentam diferentes abordagens para controlar a execução do *workflow* e lidar com o registro da proveniência dos dados. Durante a análise destes sistemas, foi possível observar que ainda não há uma padronização de um esquema de dados, além de não oferecerem interfaces para as consultas aos dados de proveniência armazenados. Além disso, a particularidade em controlar a execução distribuída do *workflow* em ambientes PAD torna o gerenciamento dos experimentos envolvendo *workflows* distribuídos uma tarefa ainda muito complexa. Os SGWfC existentes gerenciam a execução de forma ou seqüencial ou distribuída/paralela. Os SGWfC ditos seqüenciais, como Kepler, VisTrails e Taverna possuem interfaces gráficas com vários recursos de definição de *workflows*. Além disso, são ricos na gerência da proveniência de dados, mas muito limitados na execução de programas com paralelismo. No outro extremo estão os SGWfC distribuídos, voltados a grade computacionais, como o Swift (ZHAO *et al.*, 2007), Triana (TAYLOR *et al.*, 2003) e Askalon (WIECZOREK *et al.*, 2005), ricos no apoio à execução paralela do *workflow* em geral, mas limitados na captura, armazenamento e consulta a dados de proveniência.

1.2 Objetivos

O objetivo desta dissertação é apresentar uma solução que auxilie os pesquisadores de bioinformática na automação, execução e gerenciamento de seus experimentos *in silico* em ambientes PAD, bem como capturar os dados de proveniência referentes ao experimento distribuído.

Para atingir este objetivo, propomos uma arquitetura para execução de *workflows* científicos em ambientes distribuídos com apoio à proveniência de dados. A arquitetura visa a auxiliar os pesquisadores de bioinformática na definição, execução e análise de seus experimentos *in silico* através do uso de SGWfC e possibilitar as suas execuções em ambientes distribuídos (cluster de PC ou grade computacional). Considerando a qualidade das interfaces gráficas de SGWfC seqüenciais e o seu apoio à gerência de proveniência, a arquitetura proposta visa complementar esses SGWfC provendo recursos de execução paralela de tarefas do *workflow*.

Quando os *workflows* precisam ser executados em ambientes de execução distintos, diferentes mecanismos podem ter que ser adotados para continuar o controle da sua execução. Uma maneira de lidar com essa heterogeneidade é desenvolver ferramentas para monitorar a execução do experimento entre os diferentes ambientes e registrar a proveniência.

A arquitetura foi desenvolvida para ser independente de qualquer SGWfC, dito sequencial. Para complementar os SGWfC, desenvolvemos um conjunto de módulos dispostos na arquitetura. A função dos módulos é monitorar a execução de tarefas remotas do *workflow* no ambiente distribuído, capturar e armazenar os dados de proveniência referentes a execução do *workflow* (experimento).

Para definir quais etapas são necessárias para um experimento de simulação de dinâmica molecular, realizamos um levantamento junto aos pesquisadores do Laboratório de Física Biológica e do Laboratório de Modelagem e Dinâmica Molecular, ambos do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio Janeiro. Foram então definidos dois *workflows*, um, denominado GromDFlow, que visa a atender os pesquisadores na definição e execução de seus experimentos de simulação de dinâmica molecular; e outro, denominado GromDFlow Avançado, uma variação do primeiro, que visa a atender os pesquisadores que possuem mais experiência e têm a necessidade de executar experimentos mais complexos de dinâmica molecular. Ambos *workflows* foram modelados para execução no SGWfC Kepler. O Kepler foi escolhido por apresentar, no momento do início da elaboração desta dissertação, recursos mais adequados ao desenvolvimento de uma solução para execução de tarefas de *workflows* em ambientes distribuídos, como por exemplo, a disponibilização de um componente

SSH para realizar uma conexão a um ambiente remoto e, também, por ser expansível, oferecendo mecanismos para a inclusão de novos componentes à sua biblioteca.

Dentre os programas utilizados para realizar as simulações de dinâmica molecular está o pacote de programas GROMACS (GROMACS, 2008). Escolhemos o GROMACS por ser o programa mais utilizado por pesquisadores em simulações de dinâmica molecular. Além disso, também foi construído um portal *web* com o objetivo de oferecer uma interface gráfica que disponibilize os módulos desenvolvidos na arquitetura de forma transparente ao usuário. O portal, denominado GromDExp, atua como uma camada intermediária entre o SGWfC e o usuário, oferecendo uma interface gráfica que permite ao pesquisador definir e executar seus *workflows* de simulações de dinâmica molecular em ambientes distribuídos, além de permitir que o usuário acompanhe a execução de seu experimento. Para auxiliar na análise do experimento, o portal também disponibiliza um ambiente para consultas aos dados de proveniência que foram armazenados referentes aos experimentos.

Os experimentos foram divididos em três estudos de caso: no primeiro, validamos o *workflow* modelado no Kepler, o GromDFlow. O experimento foi realizado executando parte do GromDFlow em um ambiente de cluster de PC. No segundo, definimos as etapas necessárias para a execução de um experimento de simulação de dinâmica molecular em um ambiente de grade computacional. E, no terceiro, os experimentos foram feitos baseados na arquitetura proposta nesta dissertação. Os experimentos foram realizados a partir do GromDExp. Estes experimentos mostraram evidências quanto a efetividade dos recursos disponíveis e a viabilidade da execução de um *workflow* real de simulação de dinâmica molecular em um ambiente distribuído.

1.3 Organização dos Capítulos

Para uma melhor compreensão, os capítulos que compõem esta dissertação estão organizados da seguinte maneira: o capítulo dois apresenta uma revisão aos conceitos relacionados à bioinformática, proteínas, a modelagem molecular e a técnica de simulação de dinâmica molecular, bem como a definição de *workflows*, *workflows* científicos e dos sistemas de gerência de *workflows* científicos. Também são apresentados e analisados alguns trabalhos correlatos na área. No capítulo três, é apresentada a arquitetura proposta descrevendo os seus módulos e funcionalidades. O

capítulo quatro descreve as etapas necessárias para que uma simulação de dinâmica molecular seja realizada, define um *workflow* e modela o *workflow* no SGWfC Kepler. O capítulo cinco apresenta os experimentos que foram realizados para avaliar as soluções propostas nesta dissertação. E, por último, o capítulo seis apresenta as conclusões desta dissertação, analisando as contribuições que foram obtidas e apontando alguns possíveis trabalhos futuros.

Capítulo 2

2 Uso de *Workflows* Científicos na Bioinformática

O objetivo deste capítulo é apresentar uma revisão bibliográfica de alguns conceitos importantes que serão referenciados ao longo desta dissertação. Na seção 2.1 definimos a bioinformática e as proteínas. Na seção 2.2 descrevemos sobre a modelagem molecular e a técnica de simulação de dinâmica molecular. Na seção 2.3 apresentamos alguns conceitos sobre *workflows*, *workflows* científicos e referenciamos alguns sistemas de gerência de *workflows* científicos. Também são apresentados alguns sistemas de gerência de *workflows* científicos no ambiente de grade computacional e alguns conceitos relacionados à proveniência de dados nos *workflows* científicos e, por último, são apresentadas algumas considerações finais.

2.1 Bioinformática

O termo Bioinformática foi concebido no final dos anos oitenta por Hwa Lim, mas só foi popularizado a partir da década de noventa através da sua associação com o projeto genoma humano (GOODMAN, 2007). Podemos definir bioinformática como uma área de pesquisa interdisciplinar onde através das áreas de biologia e computação, é possível aplicar técnicas computacionais para a administração e análise de dados biológicos, tendo como um dos principais objetivos a descoberta de informações biológicas ocultas em uma grande quantidade de dados.

Segundo GIBAS (2001), bioinformática é a aplicação da tecnologia da informação para a gestão de dados biológicos. O aspecto funcional da bioinformática é a representação, armazenamento e distribuição de dados. Os bancos de dados, a criação de ferramentas para consultar as bases de dados e o desenvolvimento de interfaces permite os pesquisadores fazerem análises complexas dos dados.

Existem diversos métodos aplicados à bioinformática para geração e análise de dados biológicos. A Figura 2.1 apresenta uma visão esquemática da Bioinformática.

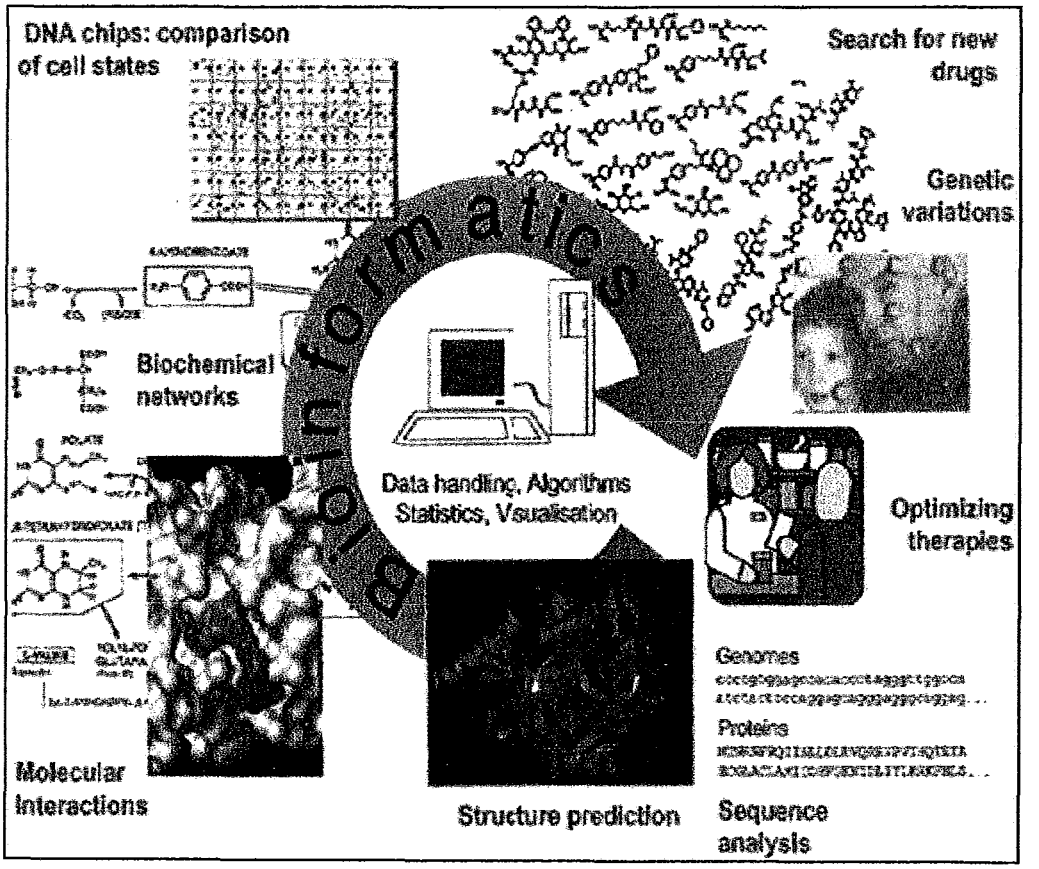

Figura 2.1 - Uma visão esquemática da bioinformática (LENGAUER, 2002).

Um dos estudos feitos através da bioinformática é a associação das proteínas às suas funções, que podem ser abordadas através da técnica de modelagem molecular. Na seção seguinte daremos uma breve introdução a proteínas, a base de dados do PDB (*Protein Data Bank*) e logo após definiremos os conceitos da modelagem molecular e da técnica de simulação de dinâmica molecular.

2.1.1 Proteínas

As proteínas são compostas essencialmente por 20 tipos diferentes de aminoácidos, cada um contendo propriedades químicas distintas, conforme Figura 2.2. Os aminoácidos são o alfabeto da estrutura protéica. Centenas de diferentes tipos de

proteínas existem em uma célula típica, cada uma delas, executando uma função ou atividade biológica diferente (LEHNINGER *et al.*, 2006).

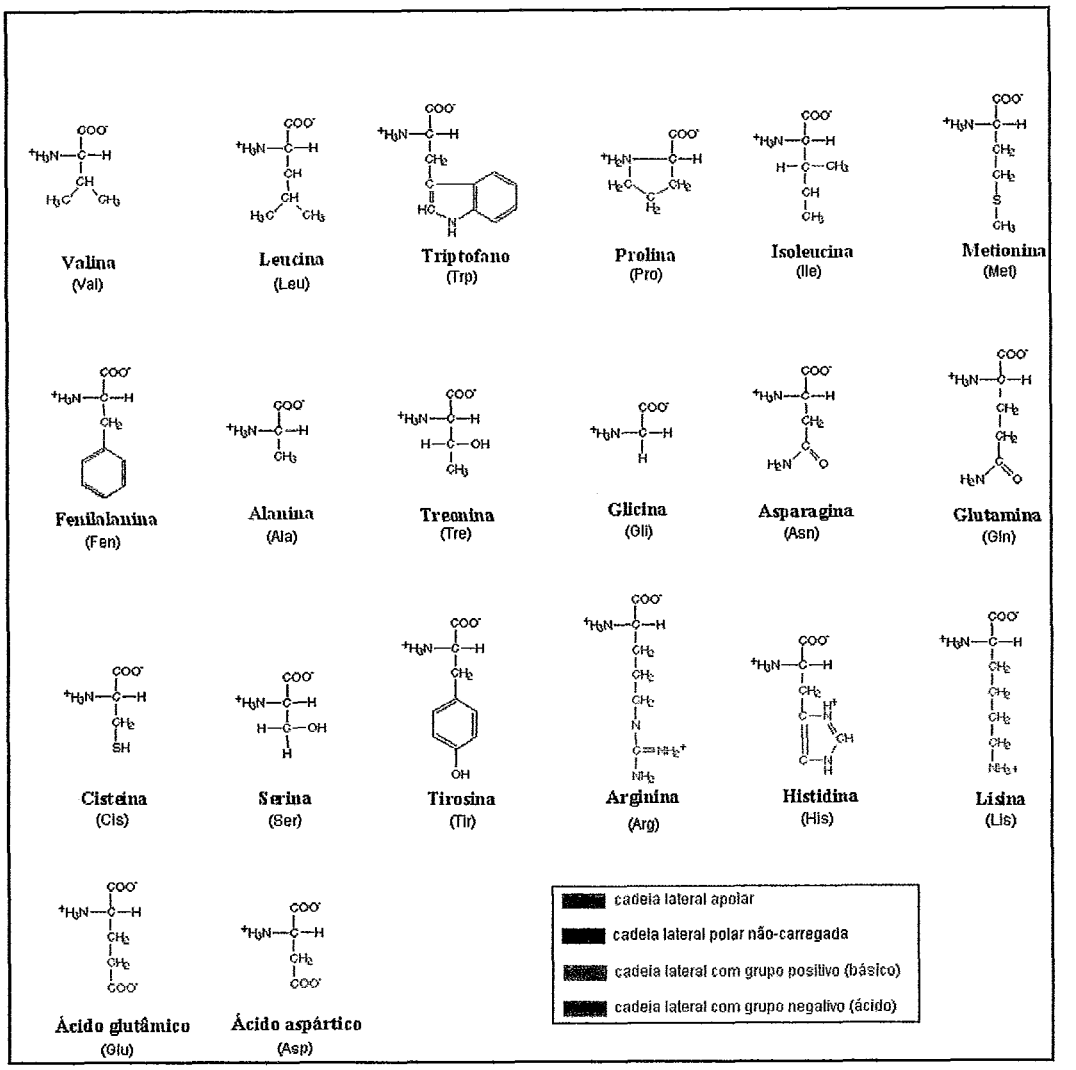

Figura 2.2 - Os vinte aminoácidos que compõem as proteínas.

Cada aminoácido é ligado ao seu vizinho por uma ligação peptídica covalente, sendo assim, as proteínas são também chamadas de polipeptídios. Uma cadeia polipeptídica é a sequência linear dos diferentes grupos químicos derivados dos aminoácidos (cadeias laterais) distribuídos ao longo da cadeia principal de ligações peptídicas (ALBERTS *et al.*, 2004). Algumas proteínas consistem em uma única cadeia polipeptídica, outras possuem duas ou mais cadeias, chamadas de multissubunidades (LEHNINGER *et al.*, 2006).

O que difere uma proteína de outra é a sequência dos resíduos de aminoácidos. As proteínas podem ser agrupadas em famílias estruturais, onde cada membro apresenta uma sequência de aminoácidos, mas com uma conformação tridimensional comum a outros membros da família (ALBERTS *et al.*, 2004).

As propriedades funcionais das proteínas dependem da sua estrutura tridimensional. Existem quatro níveis de estruturas: **primária**, que apresenta a ordem em que os aminoácidos estão ligados; **secundária,** é o arranjo dos átomos do esqueleto da cadeia polipeptídica no espaço; **terciária**, inclui o arranjo tridimensional de todos os átomos da proteína e os das cadeias laterais; **quaternária**, é o arranjo das subunidades de múltiplas cadeias polipeptídicas umas em relação às outras (CAMPBELL e FARREL, 2004), conforme mostra a Figura 2.3.

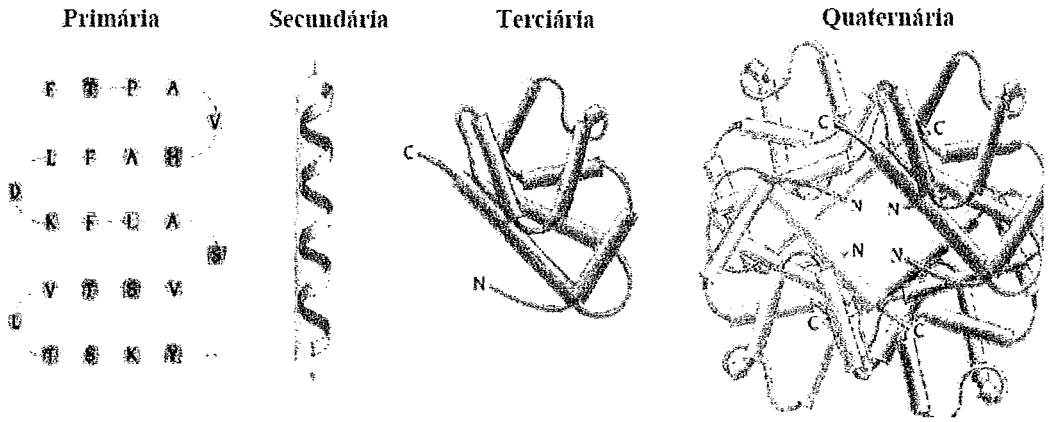

Figura 2.3 – Representação dos diferentes níveis estruturais das proteínas.

O conhecimento da estrutura de uma proteína constitui uma informação valiosa para a determinação de sua função permitindo a identificação de domínios conhecidos, como sítios catalíticos, sítios de modificações, entre outros.

Existem dois principais padrões de estruturas secundárias, o primeiro padrão chamado de **α hélice**, que se assemelha a uma escada em espiral e envolve apenas uma cadeia polipeptídica, ou seja, ela é estabilizada por pontes de hidrogênio paralelas ao seu eixo que ocorrem no interior de uma única cadeia polipeptídica; o segundo padrão, a **folha β**, gera um arranjo bidimensional envolvendo uma ou mais cadeias polipeptídicas, que se associam por meio de pontes de hidrogênio nas regiões vizinhas da cadeia polipeptídica resultando em uma estrutura achatada e rígida (CAMPBELL e FARREL, 2004). Abaixo a Figura 2.4 mostra α hélice e a folha β de uma proteína.

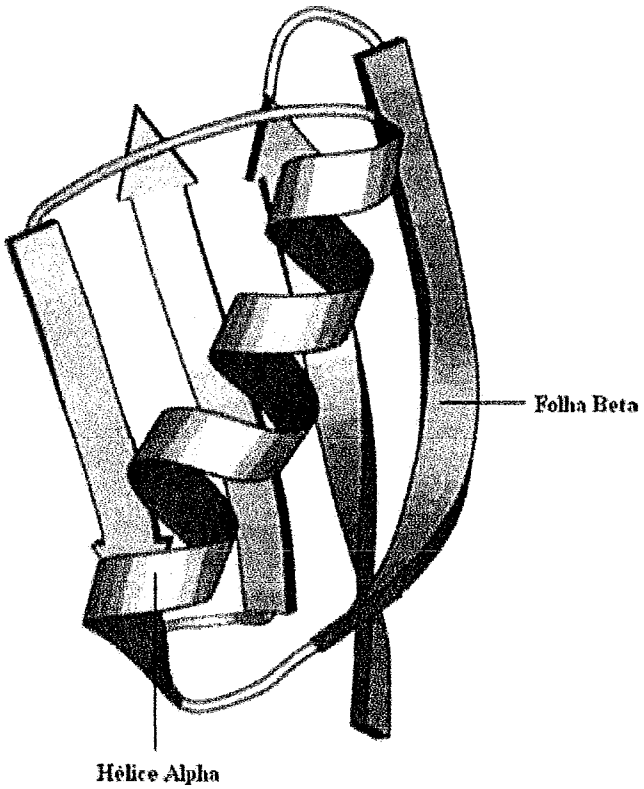

Figura 2.4 - Representação da α hélice e folha β (DALAL *et al.*, 1997).

Com os avanços em conhecimentos específicos de macromoléculas biológicas, tornou-se clara a idéia de que a estrutura protéica estava intimamente relacionada à sua função celular. Assim, muitos foram os esforços para o aprimoramento de técnicas que permitissem a obtenção de informações à cerca das características tridimensionais de proteínas. Com o crescente número de estruturas sendo resolvidas, surgia a necessidade de tornar públicas estas informações e, assim, surgiram os bancos de dados biológicos com repositórios de estruturas de macromoléculas resolvidas experimentalmente. O principal banco de dados utilizado é o PDB (*Protein Data Bank*) (PDB, 2007).

2.1.2 A Base de dados do *Protein Data Bank*

A base de dados de estruturas PDB existe desde 1971. Com o passar dos anos, o número de entradas na base de dados aumentou drasticamente, com um crescimento vertiginoso acompanhando os avanços nas diversas áreas da biologia envolvidas (BERMAN *et al.*, 2002). Com isso, o formato de acesso à base de dados do PDB sofreu grandes modificações e atualizações, enfatizando a entrada do módulo conectado à

internet, que certamente foi fator fundamental para a difusão de informação científica como um todo.

Atualmente, o PDB é mantido pela RCSB (*Research Collaboratory for Structural Bioinformatics*) que é um esforço colaborativo que envolve cientistas do Centro de Supercomputação de San Diego, da Universidade Rutgers e do Instituto Nacional de Padrões e Tecnologia (BAXEVANIS e OUELLETTE, 2005) através de um repositório *web* que abriga milhares de estruturas conhecidas de macromoléculas biológicas (PDB, 2007). Os arquivos que ele contém representam informações sobre o tipo e as coordenadas centrais dos átomos de cada proteína ou ácido nucléico. Estes dados, geralmente obtidos através das técnicas de Cristalografia de Raios X ou de Ressonância Magnética Nuclear são enviados por pesquisadores de todo o mundo.

A utilização da base de dados do PDB era inicialmente usada sobretudo por especialistas nos campos de biologia estrutural, hoje, as informações podem ser facilmente consultadas não só por especialistas, mas por qualquer outro interessado.

2.2 Modelagem Molecular

A modelagem molecular indica o processo geral de descrever sistemas químicos complexos em termos de um modelo atômico realístico, com o objetivo de compreender e prever propriedades macroscópicas baseadas em um conhecimento detalhado sobre uma escala atômica. É freqüentemente utilizada para a concepção de novos materiais, para os quais a precisão das propriedades físicas dos sistemas é necessária (VAN DER SPOEL *et al.*, 2004).

A modelagem molecular é uma técnica computacional que permite a construção e consequente visualização da estrutura de determinadas substâncias, analisando a posição dos átomos que as compõem, permitindo que se executem medições de distâncias entre diferentes moléculas, além de simular as condições de algumas reações e escolher os melhores reagentes a serem utilizados.

Um dos métodos computacionais mais utilizados na modelagem molecular é a técnica de simulação de dinâmica molecular, que no caso das proteínas, é utilizada para

estudar a estabilidade estrutural das proteínas e os sítios de ligação para outros substratos biológicos e as reações enzimáticas.

2.2.1 Função Energia Potencial

Para descrição de sistemas de macromoléculas, como as proteínas, a energia potencial é obtida por um campo de força clássico. Este campo de força descreve um sistema de muitas partículas pela sobreposição de termos simples, que descrevem a interação entre duas, três ou quatro partículas (PASCUTTI, 2004). Um dos campos de forças mais utilizados em simulações de dinâmica molecular é o GROMOS (VAN GUNSTEREN *et al.*, 1996).

As forças que atuam sobre cada átomo são obtidas calculando-se a primeira derivada do potencial em relação às posições dos átomos. Então, é a partir destas forças que se resolvem as equações de movimento para descrever como as posições atômicas variam com o tempo. As forças são reavaliadas a cada passo da dinâmica.

Para o tratamento de centenas ou milhares de átomos é usado um conjunto de funções potenciais empíricas, calibradas por informações experimentais e cálculos quânticos sobre pequenas moléculas (VAN GUNSTEREN e BERENDSEN, 1990).

2.2.2 Otimização da Estrutura

A otimização da estrutura, também chamada de minimização de energia, visa encontrar um conjunto de coordenadas que minimizam a energia potencial do sistema, obtida através do campo de força definido para o sistema.

A etapa de minimização de energia é fundamental, pois ao calcular as forças iniciais de um sistema, os valores obtidos podem ser muito altos, levando a grandes acelerações durante os passos de integração da dinâmica molecular, causando ruídos muito grandes, o que poderia invalidar a simulação. Sendo assim, a minimização corrige a configuração inicial do sistema eliminando as tensões locais e fazendo o ajuste necessário ao sistema.

O procedimento consiste em caminhar sobre a superfície de potencial na direção em que a energia decresce de maneira que o sistema é levado a um mínimo de energia.

O objetivo de um mínimo é ter uma conformação espacial que tenha relaxada as distorções nas ligações químicas, nos ângulos entre ligações e nos contatos de van der Waals (SILVA, 2003). Os algoritmos de minimização de energia abordados são os de máximo declive (*steepest descent*) e de gradiente conjugado.

2.2.2.1 Método do Máximo Declive

Máximo declive ou *steepest descent* é um método de primeira derivada que converge lentamente nas proximidades do mínimo, mas é poderoso para configurações distantes de um mínimo de energia. Com o algoritmo de máximo declive é possível construir pequenas moléculas, como aminoácidos, a partir de coordenadas atômicas aleatórias com poucos passos de otimização e depois conectá-los para formar cadeias peptídicas, com ele, também é possível melhorar estruturas cristalográficas pouco refinadas, aperfeiçoar as construídas graficamente ou construir estruturas novas (PASCUTTI, 2004).

2.2.2.2 Método do Gradiente Conjugado

Gradiente conjugado é um método mais sofisticado, que envolve a segunda derivada, porém converge com mais rapidez nas proximidades do mínimo. Possui uma busca sofisticada de um mínimo da função energia, utiliza a informação sobre a primeira derivada e leva em conta o caminho já percorrido na busca do mínimo. O valor do gradiente no ponto atual e o valor do gradiente obtido no passo anterior são utilizados para a determinação do passo seguinte (PASCUTTI, 2004).

2.2.3 Simulação de Dinâmica Molecular

Dinâmica Molecular (DM) é uma técnica que descreve os movimentos em um sistema de partículas. Ela pode ser empregada em sistemas de átomos ou moléculas. Seus elementos essenciais são o conhecimento do potencial de interação entre as partículas e das equações de movimento que governam a dinâmica das partículas (PASCUTTI, 2004). São amplamente utilizadas para se obter informações sobre as propriedades de proteínas em equilíbrio em uma solução.

Na DM calcula-se a força $\mathbf{F}_i$ que atua sobre cada partícula i a partir da derivada da função da energia potencial $E(\{r\,i\})$ que descreve a interação entre as partículas, em relação a sua posição $\mathbf{m}_i$ sendo $i = 1,2,3...N$, até o numero total de átomos.

$$\mathbf{F}_i = - \partial E(\{\mathbf{r}_i\}) / \partial \mathbf{r}_i)$$

A partir dessas forças resolvem-se as equações de movimento para descrever como as posições atômicas variam com o tempo.

Para obter a aceleração de cada átomo, divide-se a força $\mathbf{F}_i$ pela massa atômica $\mathbf{m}_i$

$$\mathbf{a}_i = \mathbf{F}_i / \mathrm{m}_i$$

Para determinar as posições em incrementos de tempo ∂t, substitui-se a aceleração no algoritmo de Verlet (VERLET, 1967).

$$\mathbf{r}_i(t + \delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \delta t) + \mathbf{a}_i(t)\,\delta t_2$$

Para a obtenção das velocidades, aplica-se o algoritmo de Verlet. As velocidades são necessárias para o cálculo da energia cinética que somada à energia potencial, resultam na energia total do sistema.

$$\mathbf{v}_i(t) = (\mathbf{r}_i(t + \delta t) - \mathbf{r}_i(t - \delta t))/2\delta t$$

As simulações de dinâmica molecular utilizam as equações de Newton, para cada partícula e em cada incremento no tempo. Através da dinâmica molecular é possível estudar a evolução temporal das configurações dos constituintes do sistema e a partir das sequências de posições geradas, determinarem as propriedades microscópicas do sistema. As partículas interagentes inicialmente dispostas em uma determinada configuração movimentam-se sob a influência de potenciais intermoleculares. Depende do próprio sistema, das propriedades que pretende estudar e da capacidade computacional o número de partículas a ser simulado.

As propriedades de equilíbrio do sistema são determinadas a partir de médias temporais sobre um intervalo de tempo suficientemente grande na escala atômica (aproximadamente entre 10^{-11} a 10^{-8} segundos de tempo real). O tempo total de

simulação depende dos processos dinâmicos que se pretende estudar e da convergência estatística das propriedades de interesses (MORGON e COUTINHO, 2007).

Um sistema pode ser simulado sob diversas condições de fronteira (SILVA, 2003). Para macromoléculas, as simulações são realizadas em uma caixa tridimensional em condições periódicas de contorno, conforme Figura 2.5, levando-se em consideração a participação explicita dos átomos do solvente e eliminando-se os efeitos de fronteira (SILVA, 2007).

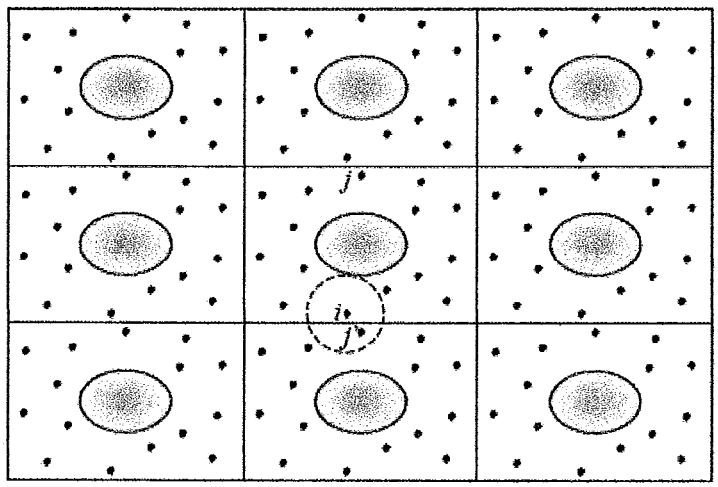

Figura 2.5 - Condições periódicas de contorno retangular. Uma proteína e seu solvente com suas oito imagens replicadas. Uma esfera de raio *Rc* é traçada a partir do átomo *i*, o qual interage com a imagem do átomo *j* (*j'*) (SILVA, 2007).

Como resultado da dinâmica molecular obtém-se as energias e trajetórias para todos os átomos e para o sistema como um todo, a partir das quais várias propriedades podem ser calculadas.

2.2.4 Programas de Dinâmica Molecular

Há alguns anos atrás, os pesquisadores precisavam desenvolver seus próprios programas de simulação. Hoje, com o avanço dos recursos computacionais, as realizações das simulações de dinâmica molecular tornaram-se mais avançadas, os cientistas não, necessariamente, precisam mais desenvolver seus próprios programas para realizar seus experimentos. Atualmente já existem diversos programas que os auxiliam a realizar suas simulações, entre eles, podemos citar, CHARMM (BROOKS e

BUCCOLERI, 1983), NAMD (PHILLIPS *et al.*, 2005) e GROMACS (VAN DER SPOEL *et al.*, 2004).

Na seção seguinte daremos mais ênfase ao pacote GROMACS por ser um dos programas mais citados na literatura e o utilizado nesta dissertação.

2.2.5 GROMACS

Desenvolvido por pesquisadores do Departamento de Química Biofísica da Universidade de Gröningen na Holanda, GROMACS que significa *Gröningen Machine for Chemical Simulation* é um software de domínio público que segue a licença GPL (*General Public License*) (VAN DER SPOEL, *et al.*, 2004). Estes programas são de uso acadêmico e livre, e já utilizados pela comunidade científica.

GROMACS é um pacote de programas para execução de simulações de dinâmica molecular e minimizações de energia. O pacote acompanha uma série de programas para execução e análise das simulações de dinâmica molecular. O seu código é portado para sistemas multiprocessados, o que permite operação em paralelo. Para execução paralela, GROMACS usa MPI (*Message Passing Interface*), que é uma biblioteca utilizada para a comunicação entre computadores. O MPI define um conjunto de rotinas para facilitar a comunicação entre tarefas paralelas (AOYAMA e NAKANO, 1999). A distribuição dos átomos é feita entre os nós de computação.

O GROMACS calcula de forma iterativa, a cada passo da simulação, a energia potencial total, as forças e as coordenadas de cada átomo do sistema. GROMACS não possui uma interface gráfica, todos os seus programas são executados por linha de comando, o que dificulta um pouco o seu uso, pois nem todos os usuários estão familiarizados com linhas de comando. Os programas do pacote GROMACS utilizados nesta dissertação são discutidos no capítulo 4.

2.3 *Workflows*

Os pesquisadores de bioinformática geralmente necessitam executar diversas tarefas que envolvem a composição de vários programas para realizarem suas pesquisas. Cada um desses programas produz uma coleção de dados, onde esta pode ser utilizada

como entrada para o próximo programa a ser executado. A composição destes programas pode ser realizada através do uso de *workflows*.

Nesta seção, abordaremos os conceitos de *workflows*, *workflows* científicos e alguns dos sistemas de gerência de *workflows* científicos encontrados na literatura.

2.3.1 Definição

O termo *workflow* originou-se na década de setenta, com o intuito de oferecer soluções para o processo de automação de escritórios. A idéia era reduzir a documentação de papel oferecendo soluções voltadas para a geração, armazenamento e compartilhamento de documentos em uma organização.

Há, na literatura, diversas definições para o termo *workflow*, mas de acordo com a *Workflow Management Coalition* (WFMC 2008), *workflow* "é a automação de um processo de negócio, inteiro ou parte dele, durante o qual documentos, informações e atividades são passadas de um participante a outro para que desenvolvam suas tarefas de acordo com um conjunto de regras".

Uma definição mais simples para *workflow* seria, segundo MORO (2006), "qualquer tarefa executada em série ou paralelo por dois ou mais membros de um grupo de trabalho visando um objetivo comum". Ou ainda, segundo (VAN DER AALST *et al.*, 2003) um *workflow* é uma coleção de tarefas organizadas para realizar um determinado processo. Através do *workflow* é definida a ordem de execução das tarefas e o fluxo das informações pertencentes ao processo a ser realizado. O principal objetivo da modelagem de um *workflow* é aumentar a eficiência dos processos e a efetividade das equipes que trabalham para executá-lo.

2.3.2 *Workflows* Científicos

Os *workflows* definidos na seção 2.3.1 também são chamados de *workflows* de negócio. Enquanto que os *workflows* de negócio modelam o processo de negócio de uma empresa, os *workflows* científicos modelam o experimento científico de um laboratório de pesquisas. Segundo VAN DER AALST *et al.* (2003), os *workflows* científicos são muito semelhantes aos *workflows* de negócio no que diz respeito a sua

forma e padrões. Analogamente, um *workflow* científico é uma coleção de tarefas científicas organizadas para realizar um determinado experimento. Através do *workflow* é definida a ordem de execução das tarefas e o fluxo das informações científicas pertencentes ao experimento a ser realizado. Porém, os *workflows* científicos são caracterizados pelo fluxo de dados, em contraposição ao fluxo de tarefas. Além disso, as tarefas que compõem um processo de negócio costumam ser simples ao passo que as tarefas do experimento científico são complexas e podem levar dias ou meses sendo executadas.

O *workflow* é a principal abstração na representação de um experimento científico. Um experimento científico, passa pela definição de um *workflow* e inúmeras variações que são analisadas e executadas ao longo de um ciclo de vida. De acordo com GOBLE *et al.* (2003) o ciclo de vida pode ser definido conforme a Figura 2.6.

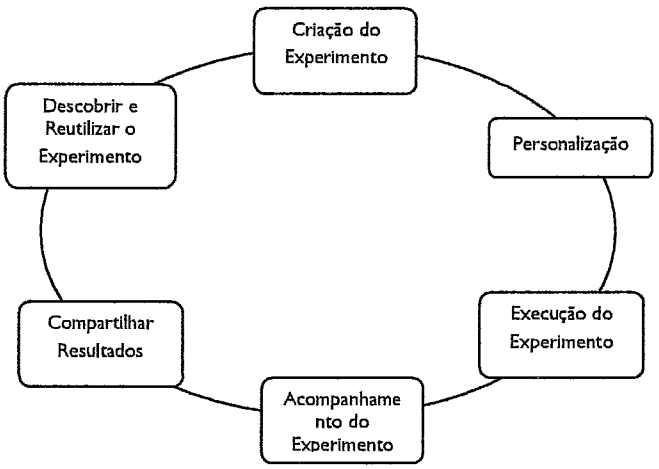

Figura 2.6 - Ciclo de vida de um experimento científico (GOBLE *et al.*, 2003).

São seis as etapas que fazem parte do ciclo de vida de um experimento científico.

1. **Criação do experimento** – etapa em que o *workflow* é modelado numa tentativa de representar o experimento científico.

2. **Personalização** – etapa em que são realizados ajustes em parâmetros e controles do *workflow* e são registradas as anotações relevantes para o experimento.

3. **Execução do experimento** – etapa em que ocorre a execução de cada uma das tarefas do *workflow* e do fluxo de dados.

4. **Acompanhamento do experimento** – etapa onde ocorrem as verificações sobre o andamento dos resultados da execução do *workflow*.

5. **Compartilhar recursos** – etapa onde é feita a publicação do *workflow* gerado, de tal modo que ele possa ser descoberto e utilizado em outros experimentos.

6. **Descobrir e reutilizar o experimento** – etapa onde se procura um *workflow* anteriormente produzido que seja relacionado ao experimento sendo modelado, para que este seja incluído e adaptado a outro experimento.

Os *workflows* científicos são geralmente compostos por um conjunto de poucas tarefas, mas algumas destas tarefas demandam um processamento de alto desempenho, fazendo com que sua execução necessite de diferentes ambientes de computação. Por requerer elevado esforço computacional, estas tarefas são candidatas a serem executadas em um ambiente de processamento de alto desempenho, como clusters de PC, que é composto por vários processadores interligados em uma rede, pertencentes a uma única unidade; ou por grades computacionais, que é um ambiente amplamente distribuído e heterogêneo, que tem como objetivo compartilhar e agregar recursos geograficamente distribuídos em múltiplos domínios administrativos (FOSTER *et al.*, 2001). Isto cria um cenário distribuído e heterogêneo para a execução do *workflow*.

Além de demandar um processamento de alto desempenho, os *workflows* científicos também geram normalmente um grande volume de dados. O armazenamento dos dados gerados é considerado muito importante, pois é a partir destes dados que o pesquisador tem acesso as informações de um determinado experimento, ou seja, ter acesso aos dados de proveniência, como por exemplo, quando o experimento foi executado e quais parâmetros foram utilizados para sua execução. Há dois tipos de armazenamento de dados: o centralizado e o distribuído (MENEZES, 2008). No ambiente centralizado, os dados gerados durante a execução de um *workflow* são armazenados em um único repositório. O repositório pode estar localizado em qualquer servidor que possua capacidade de armazenamento. Sendo assim, o dado que é gerado por cada etapa de um *workflow* é enviado para um repositório central. O armazenamento

distribuído consiste na disponibilização de diversos repositórios de dados. Os dados gerados durante cada etapa da execução de um *workflow* podem ser armazenados em qualquer um dos repositórios que estiver disponível.

O uso de *workflows* científicos vem se tornando uma solução alternativa para auxiliar os pesquisadores de bioinformática na modelagem de seus experimentos, permitindo que o processo de execução de suas tarefas seja automatizado. Para a construção, execução e análise de um *workflow* é necessário o uso de um sistema de gerência de *workflows* científicos.

2.4 Sistemas de Gerência de *Workflows* Científicos

Um sistema de gerência de *workflow* científico (SGWfC) é um software que visa a fornecer a infra-estrutura necessária para definir, executar e monitorar *workflows* (ALTINTAS, 2004). É voltado para a execução de experimentos científicos que envolvem várias áreas de pesquisa, entre elas estão a biologia, a química, a física e a engenharia. Nele, é definida a sequência em que as tarefas de um determinado experimento científico são executadas.

SGWfC operacionais surgiram nos últimos cinco anos e estão em constante evolução. Idealmente, um SGWfC deve apoiar todo o ciclo de vida de um experimento científico. Entretanto, o estado-da-arte dos SGWfC é ainda limitado em relação ao apoio à gerência do experimento científico, focando apenas na execução de um *workflow* e de modo independente do contexto do experimento científico. Assim, considerando o ciclo de vida da Figura 2.6 os SGWfC se limitam às etapas 3 e 4. SGWfC partem de um *workflow* já modelado (por exemplo, criado e personalizado), oferecem uma interface para representá-lo em sua linguagem gráfica, gerenciam a execução do *workflow* e provêem algum apoio à proveniência dos dados da execução deste *workflow*. Assim, a descoberta, reuso e compartilhamento não são cobertos.

De acordo com LEMOS (2004) os requisitos que um SGWfC deve atender são próximos das etapas do ciclo de vida do experimento científico proposto por GOBLE *et al.* (2003) onde destacamos:

1. **Processos, dados e recursos** – o SGWfC deve incluir as tarefas, dados e recursos normalmente usados e oferecer mecanismos de extensibilidade para acomodar novas tarefas, dados e recursos.

2. **Definição** – auxiliar os pesquisadores na definição e redefinição dos *workflows*. A redefinição se faz quando os resultados finais não forem considerados úteis pelos pesquisadores.

3. **Validação** – dispor de ferramentas para validar o *workflow* definido pelo pesquisador. As entradas e saídas de cada tarefa definidas pelo pesquisador devem ser verificadas durante a validação.

4. **Otimização e Execução** – o *workflow* definido pelo pesquisador deve ser otimizado e executado, conforme a arquitetura utilizada.

5. **Agendamento** – proporcionar agendamento do *workflow*. O *workflow* pode ser executado uma única vez ou de tempos em tempos.

6. **Metadados** – armazenar metadados[2] sobre os *workflows* oferecendo mecanismos de atualizações e consultas ao pesquisador. Através dos metadados é possível definir novos *workflows*, usar resultados gerados anteriormente e extrair os resultados das execuções.

Os SGWfC podem ser ditos seqüenciais, ou seja, executados em alguns ambientes centralizados de trabalho, como em um único computador, que é responsável por coordenar todas as tarefas solicitadas pelo usuário. Os SGWfC podem ser ditos distribuídos, ou seja, executados num ambiente de processamento de alto desempenho (PAD), como um cluster de PC ou grades computacionais.

Os SGWfC gerenciam a execução de forma ou seqüencial ou distribuída. Os SGWfC dito seqüenciais, possuem interfaces gráficas com vários recursos para a definição de *workflows*, além de oferecem suporte na gerência da proveniência dos dados. Já os SGWfC distribuídos oferecem apoio a execução paralela do *workflow*, porém são limitados na captura, armazenamento e consulta aos dados de proveniência.

[2] Metadados pode ser definido como "uma informação sobre o dado que permite o acesso e gerenciamento deste dado de maneira eficiente e inteligente" (LEMOS, 2004).

Conforme mencionado anteriormente, as tarefas que compõem um experimento científico são normalmente complexas e podem levar dias ou meses sendo executadas, com isso há a necessidade de um mesmo experimento ser executado em ambientes distintos, parte no ambiente centralizado parte no ambiente distribuído, pois algumas das tarefas do *workflow* demandam alto poder computacional e necessitam de ser executadas em um ambiente distribuído.

2.4.1 Sistemas de Gerência de *Workflows* Científicos em Ambientes de Grade Computacional

O ambiente de grade computacional, conhecido na literatura como *grid computing*, pode ser definido como uma infra-estrutura global de computação científica, que integra e permite o compartilhamento de recursos heterogêneos através de redes de computadores (FOSTER e KESSELMAN, 1999). Tendo como principal objetivo a execução de aplicações que exigem grande poder computacional, podendo ser utilizado tanto em ambientes comerciais quanto em ambientes científicos. Através da grade é possível que comunidades científicas compartilhem dados, aplicações e recursos computacionais. O uso de *workflows* em ambiente de grade computacional facilita o compartilhamento destas informações.

Segundo YU e BUYYA (2005), os *workflows* em ambientes de grade computacional podem ser definidos como uma coleção de tarefas que são processadas em recursos distribuídos sob uma ordem bem definida e com um objetivo definido. A arquitetura dos sistemas de *workflow* em ambientes de grade computacional possui duas principais funções, que são tempo de construção (*build time*) e tempo de execução (*run time*). A função de tempo de construção define e modela o *workflow*, e a tempo de execução gerencia a execução do *workflow* e suas interações com os recursos de grade.

Na subseção seguinte descrevemos alguns dos SGWfC por serem os mais referenciados na literatura, além de serem voltados para as aplicações científicas.

2.4.2 Kepler

Kepler é um sistema de gerência de *workflows* científicos que representa e executa *workflows* científicos, auxiliando os pesquisadores a realizarem seus

experimentos (ALTINTAS, 2004). Desenvolvido na Universidade da Califórnia de Berkeley, é baseado no sistema orientado a fluxo de dados Ptolemy II, com código livre e desenvolvido em Java (SUN, 2008). O Ptolemy II é parte do projeto Ptolemy (PTOLEMY, 2008). O Ptolemy é um framework para modelagem, simulação e projeto de sistemas.

Kepler surgiu com a necessidade de cientistas especializados terem autonomia para criarem seus próprios *workflows* científicos executáveis, evitando parte da programação complexa requerida para a criação desses modelos executáveis. O sistema fornece uma interface gráfica, chamada Vergil. Para a criação de um modelo visual de um *workflow*, basta arrastar e soltar objetos dentro da área de criação de *workflows* e, em seguida, conectar esses objetos, desenhando o fluxo do trabalho. Entretanto, a atividade de modelagem do *workflow* tem que ser realizada por fora do SGWfC, assim como em todos os demais SGWfC. A execução de um *workflow* é representada pelo diretor e os componentes executáveis são representados pelos atores. O diretor define quando o *workflow* será executado e o ator define o que será executado. Um *workflow* orientado a ator é formado por um conjunto de atores que representam tarefas e por um conjunto de conexões de fluxos de dados que conectam os atores através de portas de dados (KEPLER, 2008).

Através do Kepler é possível visualizar todas as etapas de um *workflow*, o que fornece um melhor entendimento de como os dados fluem entre os componentes desse *workflow*. O Kepler utiliza a linguagem MoML baseada em XML (W3C, 2008), que permite a especificação do processamento das tarefas, transferência de dados e sua execução.

O Kepler dispõe de alguns componentes (atores) que permite realizar uma conexão a um ambiente remoto, como o ator SSH e, também o ator GridFTP, para apoio a execução de *workflows* no ambiente de grade computacional, porém os componentes para execução do *workflow* na grade ainda estão em fase de desenvolvimento, o que dificulta seu uso. Também provê componentes para apoio a proveniência de dados, como o *Provenance Recorder*, que captura as informações sobre a execução do *workflow* e o *Smart Rerun Manager* que permite a re-execução parcial de um *workflow*, porém estes componentes ainda não estão disponíveis para uso.

2.4.3 VisTrails

VisTrails é um sistema de gerência de *workflow* científico desenvolvido em Python (PYTHON, 2008) pela Universidade de Utah, que fornece suporte para exploração e visualização de dados, permitindo o rastreamento da evolução do *workflow* (CALLANHAN *et al.*, 2006a).

A principal característica do VisTrails é o suporte para proveniência do processo de exploração (CALLANHAN *et al.*, 2006b). É o único SGWfC, dentre os analisados, que oferece um mecanismo para armazenar o registro da evolução de um *workflow*. O VisTrails captura a evolução do fluxo do dado – todos os passos são seguidos para construir um conjunto de *workflows*. O fluxo do dado é uma sequência de operações usadas para gerar uma visualização (CALLANHAN *et al.*, 2006c). Nele, são representadas várias versões do *workflow* (que difere em suas especificações, suas relações e suas instâncias (que difere nos parâmetros usados em cada execução particular)). VisTrails usa um modelo baseado em mudança para capturar a proveniência. Como os cientistas fazem modificações em um fluxo de dado particular, o mecanismo de proveniência registra essas mudanças. Em vez de armazenar um conjunto de fluxo de dados relacionados, armazena as operações ou mudanças que são aplicadas para o fluxo do dado, como exemplo, a adição de um módulo ou a modificação de um parâmetro. A informação serve como um *log* dos passos seguidos para gerar uma série de visualizações e um registro da proveniência de visualização.

O componente chave do VisTrails é a especificação parametrizada do fluxo de dado – uma especificação formal de um *pipeline*. No VisTrails existe uma clara separação entre a especificação de um *pipeline* e a execução de suas instâncias. Essa separação fornece um mecanismo escalável para gerar um grande número de visualizações. A definição de um *pipeline* pode ser usada como um modelo e instanciada com conjuntos diferentes de parâmetros para gerar várias visualizações em um modelo escalável. Um das funcionalidades do VisTrails é prover mecanismos para a visualização múltipla da definição dos *workflows*, possibilitando ao usuário controlar suas alterações. O VisTrails permite acesso a ambientes remotos através de serviços *web*.

2.4.4 Taverna

O Taverna é um sistema de gerência de *workflow* científico focado nos experimentos de bioinformática. Desenvolvido na Universidade de Manchester no Reino Unido com colaboração de outras universidades (MYGRID, 2008). É um software de código aberto, baseado em *scripts* e na linguagem Java (SUN, 2008).

Como seu foco é em experimentos de bioinformática, Taverna oferece vários recursos para facilitar a definição dos *workflows* aos pesquisadores desta área, como, estrutura de dados específicos do domínio da bioinformática e suporte semântico através de ontologias.

Possui um componente, o *Taverna Workflow Workbench*, que provê a interface gráfica para representação dos *workflows*. Os *workflows* são definidos graficamente e mapeados automaticamente na linguagem Scufl (*Simple Conceptual Unified Flow Language*) que é uma linguagem para definição de *workflows*, baseada em WSFL (*Web Services Flow Language*) (OINN *et al.*, 2004). A Scufl foi desenvolvida pela própria equipe do Taverna. Sua máquina de execução de *workflow* é chamada FreeFluo (FREEFLUO, 2008), também desenvolvida pela equipe do Taverna.

No Taverna, o processo de captura de dados é altamente dependente da sua máquina de execução de *workflows* e do modelo de dados utilizado por ele. Chegou a ser desenvolvido um componente, chamado MIR, com a função de armazenar os *workflows* em um repositório de dados, porém não foi dada continuidade ao seu desenvolvimento. O Taverna permite que o armazenamento de seus resultados seja feito em um SGBD, por padrão ele usa o MySQL.

2.4.5 GridFlow

GridFlow é um sistema de gerência de *workflows* que permite o usuário construir, simular, executar e monitorar *workflows* em ambientes de grade computacional. Cada atividade do *workflow* é representada por um *subworkflow*. No *subworkflow* é definido o fluxo de tarefas relacionadas que precisam ser executadas em uma determinada sequência dentro de uma grade. Uma tarefa é o menor elemento em um *workflow* e é geralmente paralela usando MPI.

A definição da aplicação é feita através de uma interface disponibilizada por um portal *web*. O usuário precisa definir as tarefas e a ordem de execução para criar um *workflow*. É usada uma especificação XML para descrever o *workflow* (CAO *et al.*, 2003).

Apesar das vantagens apresentadas pelo GridFlow, não foi possível verificar como é feita a especificação dos *workflows* através do portal, não há um repositório onde o usuário possa selecionar os componentes do *workflow* nem o seu funcionamento na grade. O GridFlow também ainda não oferece apoio a captura dos dados de proveniência.

2.4.6 GriddLes

GriddLeS (*Grid Enabling Legacy Software*) fornece um conjunto de facilidades para comunicação inter-processos para construção de *workflows* na grade computacional, podendo ser usadas para compartilhar dados em um *workflow* (ABRAMSON *et al.*, 2005). GriddLes emprega uma arquitetura flexível para suportar diferentes mecanismos de acesso. Um dos seus componentes, chamado GridFiles é um dispositivo que permite a comunicação inter-processo baseada em arquivo entre os componentes de software. GridFiles é suportado por um componente chamado *FileMultiplexer* que fornece um mecanismo I/O que chama uma aplicação (abrir, ler, escrever) e mapeia essas operações dinamicamente para os serviços apropriados. Estes mecanismos podem ser usados para implementar algum suporte à proveniência distribuída. GriddLeS é uma camada intermediária integrada ao SGWfC Kepler, porém ainda está em fase de desenvolvimento, não estando disponível para testes no Kepler.

2.5 Proveniência de Dados em *Workflows* Científicos

Conforme dito anteriormente, um grande volume de dados pode ser gerado por um *workflow* científico. É importante guardar o registro do processo de obtenção dos dados para que se possa analisá-lo, garantir a reprodutibilidade de um dado experimento ou ainda reusá-lo em outros experimentos. Podemos chamar o mecanismo de registro destes dados de proveniência de dados.

Segundo BUNEMAN *et al.*, (2001), o termo proveniência de dados, também conhecido como linhagem de dados ou *pedigree,* é a descrição da origem de um dado e de todo o processo pelo qual foi produzido, ou seja, proveniência de dados refere-se a fontes de consultas ou a serviços baseados no processamento de resultados.

STEVENS *et al.* (2007) definiram quatro tipos de proveniência que podem ser coletadas:

1. **Processo** – é similar aos tradicionais *logs*, registra a ordem dos serviços invocados e os dados processados ou produzidos por cada invocação do serviço.

2. **Dados** – está relacionado com as origens de uma parte do dado, ou a entrada e saída do *workflow* ou mesmo um dado intermediário produzido.

3. **Organização** – expõe o criador do dado produzido: o serviço, o *workflow* ou o projeto que o experimento pertence.

4. **Conhecimento** - fornece uma visão mais abstrata a mais que os *logs* e as derivações, ou um entendimento específico sobre os dados processados ou gerados durante as execuções.

Há duas formas de proveniência, a **prospectiva** que captura a especificação da tarefa computacional e os passos correspondentes que devem ser seguidos para gerar os dados e, a **retrospectiva** que captura os passos executados como informação sobre o ambiente usado para derivar um dado produzido. As informações capturadas são: o que foi executado, quem executou, qual o tempo de execução, hora de inicio e termino da execução (FREIRE *et al.* 2008)

No contexto desta dissertação, que tem como um dos objetivos capturar os dados de proveniência, o tipo de proveniência definida por STEVENS *et al.* (2007) que será coletada são os dados referentes a execução do *workflow*, ou seja, a proveniência retrospectiva, pois registrar os dados referentes a execução de um *workflow* adiciona um valor significativo para os projetos científicos com grande fluxo de dados. É a partir destes dados que o pesquisador tem acesso as informações de um determinado experimento, como por exemplo, quais parâmetros foram utilizados para sua execução, quais dados foram gerados e quais os tempos de execução.

É fundamental que os SGWfC ofereçam recursos para armazenar e consultar as informações relacionadas a execução dos *workflows*. Os principais objetivos da importância da captura de proveniência de dados nos *workflows* científicos são:

- Identificar quem realizou e quando o experimento foi realizado;
- Identificar os dados de entrada de um experimento;
- Ajudar na compreensão do experimento;
- Auxiliar na identificação dos resultados.

Além disso, também é fundamental saber onde o experimento foi executado, pois parte da execução das tarefas do *workflow* podem ser executadas em ambientes computacionais distintos, daí a necessidade da captura da proveniência dos dados distribuídos.

2.6 Considerações Finais

Dentre os SGWfC analisados o único que tem o foco em bioinformática é o Taverna, os demais estão ligados aos experimentos científicos em geral.

O Kepler foi o sistema pioneiro que surgiu com foco em experimentos científicos em geral. Kepler permite a execução de uma tarefa remota do *workflow* através do seu componente (ator) SSH, para realizar uma conexão a um ambiente remoto. Também dispõe de alguns componentes (atores) para execução de *workflow* em uma grade computacional, como o GridFTP, porém o uso destes componentes de grade computacional ainda estão em fase de desenvolvimento. Para apoio à proveniência de dados, o Kepler propõe o *Provenance Framework* (KPF, 2008), que contém classes que observa a execução do *workflow* e armazena os dados gerados durante a execução. Os componentes são o *Provenance Recorder* e *Smart Rerun Manager*, porém estes componentes, até o momento da elaboração desta dissertação, ainda não estavam disponíveis à comunidade para testes.

O VisTrails dispõe de um serviço *web* para realizar uma conexão a um ambiente remoto. Uma das principais vantagens do VisTrails é a visualização e o controle de versões dos *workflows*. O apoio à proveniência se dá na etapa da definição do *workflow*.

Ele armazena toda a evolução da definição do *workflow*, possibilitando que o pesquisador controle suas alterações e visualize todas as versões dos *workflows* que foram geradas. Uma das desvantagens do VisTrails é que seu foco principal está na proveniência da etapa de definição do *workflow* e não nos dados de proveniência referentes a execução do *workflow*, porém foi desenvolvido um *plugin* para armazenar os dados de execução do *workflow* em uma base de dados. Este *plugin* ainda apresenta uma solução bem simples, capturando apenas alguns poucos dados referente à execução do *workflow*.

Apesar do foco principal do Taverna ser voltado para aplicações na área de bioinformática, não há nenhum componente voltado para os experimentos de simulação de dinâmica molecular. No Taverna, a captura dos dados de proveniência são dependentes da máquina de execução do *workflow* e do modelo de dados utilizado pelo próprio Taverna. O Taverna possui um componente, chamado MIR, para o armazenamento da proveniência, seu modelo de dados era extremamente simples e voltado apenas para o domínio da bioinformática. Uma outra alternativa do Taverna são os *logs* que são gerados durante a execução do *workflow* que contém os dados relacionados a execução do *workflow*. O Taverna permite que o armazenamento de seus resultados seja feito no SGBD MySQL.

Após análise realizada entre os SGWfC Kepler, VisTrails e Taverna, podemos concluir que eles apresentam diferentes abordagens para controlar a execução do *workflow* e lidar com os registros de proveniência de dados. Nenhum deles apóia a gerência da execução remota das tarefas do *workflow* em um ambiente distribuído e nem a captura da proveniência dos dados gerados no ambiente distribuído.

Os outros sistemas analisados, GriDFlow, GriddLeS apresentam soluções instáveis, ainda em desenvolvimento para execução dos *workflows* em um ambiente de grade computacional. Apesar do GriddLes ser uma camada integrada ao Kepler, seus componentes ainda não estão disponíveis para testes. No entanto, nenhum deles oferece serviços de apoio a captura e armazenamento de dados de proveniência distribuída.

Um ponto fraco dos SGWfC mencionados é a falta de uma interface para consultas aos dados gerados durante a execução do *workflow*. Os SGWfC ainda não possuem soluções estáveis, estando em constante fase de melhorias em seus sistemas.

A Tabela 2.1 apresenta uma comparação entre os SGWfC analisados.

Tabela 2.1 – Comparação entre os SGWfC analisados

	KEPLER	VISTRAILS	TAVERNA	GRIDFLOW	GRIDDLES
Domínio da Aplicação	Científico em geral	Científico em geral	Bioinformática	Científico em geral	Científico em geral
Execução remota de uma tarefa	Sim	Não	Não	Sim	Sim
Execução paralela de uma tarefa remota	Não	Não	Não	Sim	Sim
Utilização de serviços de grade computacional	Não	Não	Não	Sim	Sim
Armazenamento de dados	Sim	Sim	Sim	Não	Não
Proveniência de dados	Não	Sim	Sim	Não	Não
Proveniência de dados distribuída	Não	Não	Não	Não	Não
Interface para consultas	Não	Não	Não	Não	Não

De acordo com a Tabela 2.1 é possível verificar que os SGWfC seqüenciais oferecem apoio a proveniência e o armazenamento de dados, enquanto os SGWfC distribuídos são limitados na captura de proveniência, porém oferecem suporte na execução remota do *workflow*. Com base nesta análise, é possível perceber que é fundamental o apoio não apenas na definição e execução das tarefas do *workflow* no ambiente centralizado, mas também no ambiente distribuído. Uma solução para a gerência da execução das tarefas remotas dos *workflows*, bem como apoio a captura e armazenamento dos dados de proveniência gerados durante a execução da tarefa remota no ambiente distribuído seria complementar os SGWfC sequenciais provendo recursos de execução das tarefas remotas em um ambiente distribuído, além de fornecer um ambiente para consultas aos dados de proveniência armazenados.

Capítulo 3

3 Arquitetura para Execução de *Workflows* Científicos em Ambientes Distribuídos

Este capítulo propõe uma arquitetura para auxiliar os SGWfC na gerência da execução de tarefas de *workflows* científicos em ambientes distribuídos, tendo como características principais a de monitorar a execução remota da tarefa paralela dos *workflows* e capturar os dados de proveniência da execução remota do *workflow*.

O capítulo está dividido da seguinte maneira: a seção 3.1 define as camadas que compõem a arquitetura proposta e a seção 3.2 descreve os módulos que foram implementados e fazem parte da arquitetura proposta.

3.1 Arquitetura Proposta

A arquitetura proposta nesta dissertação tem como objetivo auxiliar os pesquisadores de bioinformática a automatizar seus experimentos de simulação de dinâmica molecular utilizando o programa GROMACS através do uso de *workflows* científicos e possibilitar que os experimentos sejam executados em ambientes distribuídos, além de oferecer apoio a captura e armazenamento dos dados de proveniência. Dentre o apoio à proveniência, destacamos a captura de quem executou, quando executou, quais os parâmetros e quais os arquivos de entrada foram utilizados, bem como os dados gerados durante a execução da tarefa remota do *workflow* no ambiente distribuído, além de disponibilizar um ambiente para consultas aos dados de proveniência armazenados.

A arquitetura é baseada em quatro camadas: camada de interface, camada de aplicação, camada de serviços de proveniência e camada de persistência, que podem ser vistas na Figura 3.1 e descritas a seguir. A arquitetura baseada em camadas é um padrão de arquitetura conhecida por facilitar a comunicação entre os componentes. Suas

principais vantagens são a organização, flexibilidade, facilidade de integração e independência de sistemas.

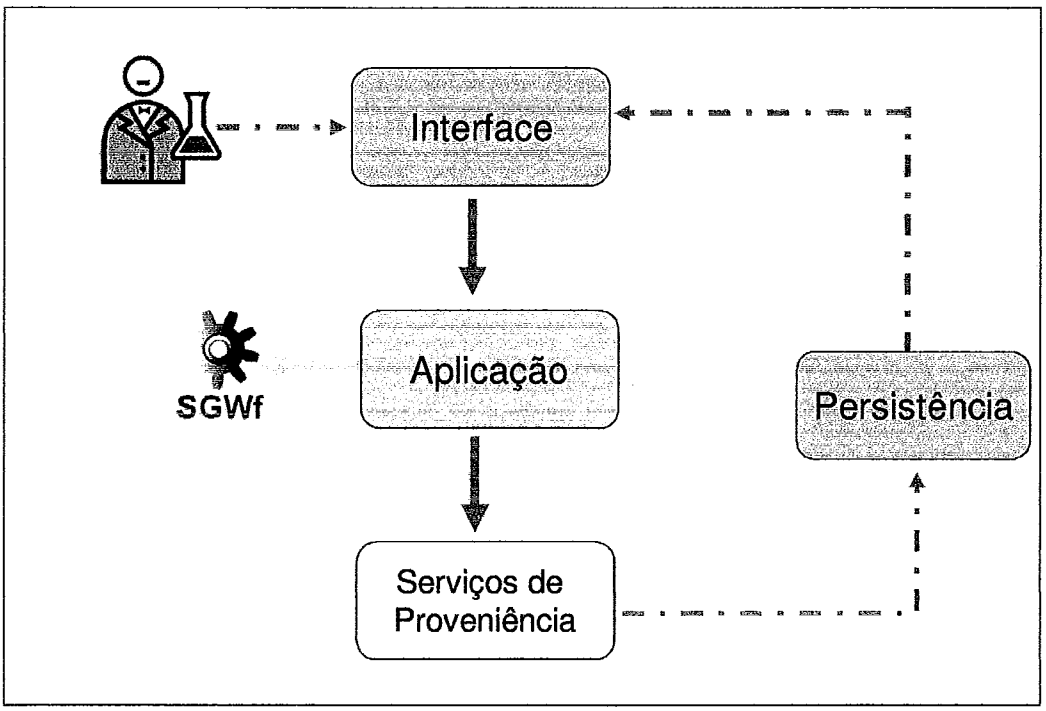

Figura 3.1 – Camadas da arquitetura proposta.

3.1.1 Camada de Interface

A camada de interface é responsável por fazer a interação direta com o usuário final, é a partir dela que o pesquisador executa seus experimentos, verifica o andamento das execuções e, também, pode realizar consultas aos dados de proveniência que foram armazenados.

Através da camada de interface, o pesquisador estabelece os parâmetros e arquivos de entrada que serão utilizados no seu experimento. Estas informações são armazenadas em um banco de dados, de acordo com um modelo de dados definido. O pesquisador não precisa ter a preocupação em conhecer o SGWfC que está sendo invocado, nem o ambiente em que está sendo executado o *workflow*, isso tudo é feito de forma transparente ao usuário. Também é possível acompanhar a execução das tarefas remotas do *workflow* no ambiente distribuído, além de consultar os dados de

proveniência armazenados. Essa interface gráfica é disponibilizada ao pesquisador através de um portal *web*. O portal será apresentado no capítulo 5 desta dissertação.

3.1.2 Camada de Aplicação

A camada de aplicação está localizada entre a camada de interface e a camada de serviços de proveniência. É a camada responsável pela execução das tarefas do *workflow*. Na camada de aplicação está localizada a máquina de execução do *workflow* que será utilizada para executar cada tarefa definida no *workflow*. A camada de aplicação é independente do SGWfC utilizado.

3.1.3 Camada de Serviços de Proveniência

A camada de serviços de proveniência contém um conjunto de módulos. Estes módulos desempenham um papel fundamental na arquitetura. O objetivo principal da camada de serviços de proveniência é monitorar a execução dos experimentos realizados nos ambientes distribuídos e capturar os dados de proveniência gerados durante a execução do experimento no ambiente distribuído de forma transparente ao usuário. Além de capturar, os dados também são armazenados de acordo com um modelo de dados definido.

3.1.4 Camada de Persistência

Na camada de persistência é onde está localizada a base de dados que armazena todos os dados importantes referentes a um determinado experimento, conforme um modelo de dados definido. Os dados são armazenados de acordo com a ordem de definição do experimento, como quem executou, quando executou, os parâmetros e arquivos de entrada utilizados e também os dados gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído. Estes dados são disponibilizados através de consultas via portal *web* para os pesquisadores. A camada de persistência acompanha todas as camadas, pois ela armazena dados das camadas de interface, aplicação e serviços de proveniência.

3.2 Módulos da Arquitetura

Os módulos que compõem as camadas de interface, de aplicação, de serviços de proveniência e de persistência fazem parte da arquitetura proposta por CRUZ *et al.* (2008). A arquitetura, denominada Matrioshka, é baseada em um conjunto de serviços que podem ser acoplados aos SGWfC sequenciais. Os principais objetivos da Matrioshka são:

- Operar acima do ambiente distribuído (como exemplo, um cluster de PC ou uma grade computacional);
- Ser independente do SGWfC utilizado;
- Ser independente do sistema de armazenamento de dados;
- Ser capaz de lidar com representações heterogêneas de conjunto de dados;
- Suportar consultas aos repositórios de proveniência de dados;
- Escalar com diferentes números de usuários e não interferir no desempenho do *workflow*;

A Figura 3.2 apresenta os módulos da arquitetura proposta. O portal *web* faz parte da camada de interface (componente em verde), o SGWfC pertence a camada de aplicação (componente em azul), os componentes cliente de proveniência e servidor de proveniência fazem parte da camada de serviços de proveniência (componentes em amarelo) e o repositório de proveniência está localizado na camada de persistência (componente em roxo).

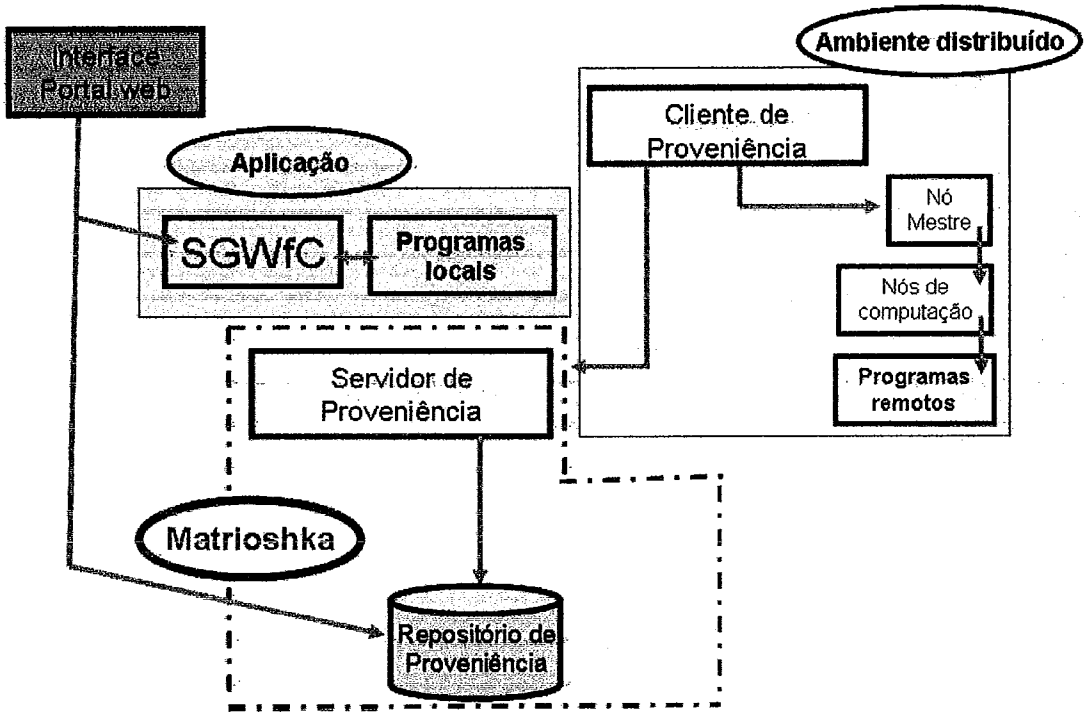

Figura 3.2 – Módulos da arquitetura proposta.

O principal objetivo da Matrioshka é monitorar a execução das tarefas remotas do *workflow* e capturar os dados de proveniência gerados durante a execução das tarefas remotas do *workflow*, fazendo a integração do SGWfC sequencial e o ambiente distribuído. O *workflow* definido no portal *web* desta arquitetura é encaminhado para ser executado por um SGWfC sequencial, porém com tarefas adicionadas para acompanhar a execução remota em um ambiente de processamento de alto desempenho e capturar os dados de proveniência gerados durante a execução das tarefas remotas do *workflow*.

3.2.1 Módulos da Camada de Serviços de Proveniência

O objetivo dos módulos da camada de serviços de proveniência é monitorar a execução das tarefas remotas do *workflow* no ambiente distribuído e capturar os dados de proveniência gerados durante a execução das tarefas do *workflow* no ambiente distribuído.

Durante a execução das tarefas remotas do *workflow*, são capturados os dados de proveniência que vão sendo gerados ao longo da execução das tarefas no ambiente

distribuído. Nos ambientes distribuídos, as aplicações são executadas em diferentes computadores e trocam informações através de uma rede de comunicação.

Os módulos que fazem parte da camada de serviços de proveniência são: o servidor de proveniência, o cliente de proveniência e o repositório de proveniência. Estes módulos estão representados na Figura 3.2.

3.2.1.1 Servidor de Proveniência

O módulo **servidor de proveniência** é um mediador que fica entre o cliente de proveniência e o repositório de proveniência (Figura 3.2). Ele desempenha um papel fundamental na arquitetura. A primeira tarefa do servidor de proveniência é associar a comunicação entre um SGWfC sequencial e o cliente de proveniência. Sua principal função é receber as mensagens enviadas pelo cliente de proveniência. Estas mensagens são referentes aos dados de proveniência que são gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído.

A outra tarefa é ordenar as mensagens de acordo com um conjunto de regras pré-definidas pelo pesquisador para um dado experimento. As informações do experimento são alcançadas durante sua execução. Após o recebimento das mensagens, o servidor de proveniência realiza a transformação dos dados e armazena os dados no banco de dados de acordo com o modelo de dados definido.

Tais tarefas atuam como organização e conhecimento da proveniência. Elas são organizadas de acordo com um modelo de dados pré-definido, onde os dados são armazenados no repositório de proveniência, permitindo que os pesquisadores possam realizar consultas sobre um dado experimento. O serviço desenvolvido no módulo servidor de proveniência fica localizado em um servidor centralizado (local).

3.2.1.2 Cliente de Proveniência

O módulo **cliente de proveniência** é o serviço que monitora a execução da tarefa remota do *workflow* que foi submetida ao ambiente distribuído. Este módulo fica localizado no servidor remoto (neste caso, no cluster de PC) aguardando até que seja iniciada uma nova execução do *workflow*.

A função do módulo cliente de proveniência é fazer o monitoramento das tarefas que estão sendo executadas no ambiente distribuído que foram enviadas pelo SGWfC sequencial e gerar as instâncias, ou seja, as notificações de eventos. Cada instância atua como um serviço mediador que roda em segundo plano (*background*), ao invés do controle ser feito pelo usuário, ele é iniciado por uma tarefa adicionada ao *workflow* que acompanha a execução remota através do disparo de uma mensagem de monitoramento.

As notificações de eventos consistem nos detalhes sobre os dados das tarefas que estão sendo executadas, como: status da execução da tarefa remota; tempos de execução; localização dos dados de saída e, mensagens de erros, quando houver. Estas informações são enviadas ao servidor de proveniência através de mensagens. O servidor de proveniência recebe as informações, trata e as armazena no banco de dados pertencente ao módulo repositório de proveniência.

Os serviços dos módulos servidor de proveniência e cliente de proveniência foram desenvolvidos utilizando a linguagem Java (SUN, 2008). Java foi escolhido por ser uma linguagem que apresenta facilidades para a implementação dos módulos definidos na arquitetura. A comunicação é feita sobre o protocolo TCP/IP. O processo de comunicação entre o servidor de proveniência e o cliente de proveniência é feito através de uma porta pré-definida, a qual fica aguardando conexões em um determinado sítio também pré-definido. Então o cliente de proveniência solicita uma conexão, se nenhum problema ocorrer, o servidor de proveniência aceita a conexão e cria um canal de comunicação entre eles. Tipicamente, o comportamento do cliente de proveniência é ficar em um *loop* aguardando novas conexões para atender as solicitações do servidor de proveniência.

3.2.1.3 Repositório de Proveniência

O módulo **repositório de proveniência** contém o banco de dados onde são armazenados os dados referentes aos usuários e aos *workflows*. Estes dados são enviados ao repositório pelo servidor de proveniência. O repositório de proveniência fica localizado em um repositório central. O tipo de armazenamento de dados utilizado, de acordo com o que foi definido no capítulo 2, é o centralizado, pois apesar dos dados serem capturados durante a execução das tarefas remotas do *workflow* no ambiente distribuído, eles são armazenados em um único repositório de dados.

Os dados de proveniência armazenados no repositório de dados de proveniência são referentes à proveniência **retrospectiva**, como por exemplo, os dados referentes a execução do *workflow*, como quem executou, quando executou, os parâmetros e arquivos de entrada utilizados, o *status* da execução da tarefa remota do *workflow* no ambiente distribuído, a data e hora de início e término da execução e os tempos de execução.

Por ainda não existir um modelo de dados padrão para a representação da proveniência de dados em geral e nem para os experimentos envolvendo os *workflows* de simulação de dinâmica molecular, foi definido um modelo de acordo com as necessidades levantadas junto aos pesquisadores do IBCCF/UFRJ para o armazenamento dos dados de proveniência referentes aos experimentos de simulações de dinâmica molecular.

A Figura 3.3 apresenta o modelo conceitual do banco de dados do repositório de proveniência. No modelo conceitual são definidas as classes, os tipos de dados, os relacionamentos e as operações. Vale salientar que o modelo conceitual pode ser implementado em qualquer SGBD.

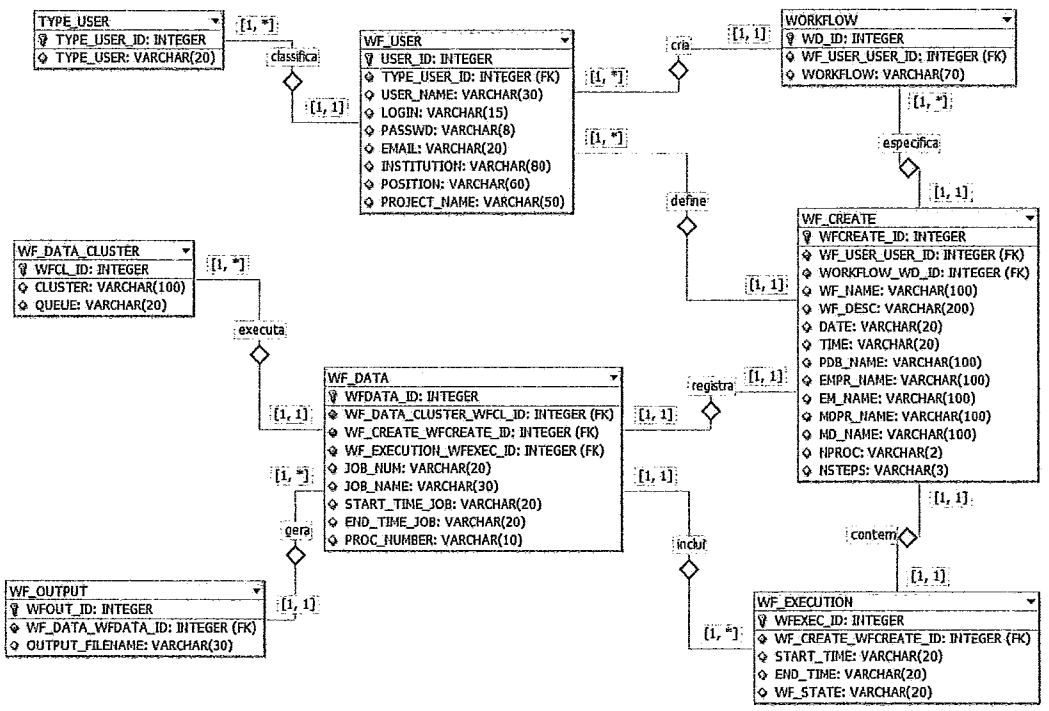

Figura 3.3 – Modelo conceitual do módulo repositório de proveniência.

A Figura 3.4 apresenta o modelo lógico. O modelo lógico reúne as informações referentes aos usuários e aos *workflows*. Estas informações possibilitam os pesquisadores a realizarem consultas sobre os dados armazenados.

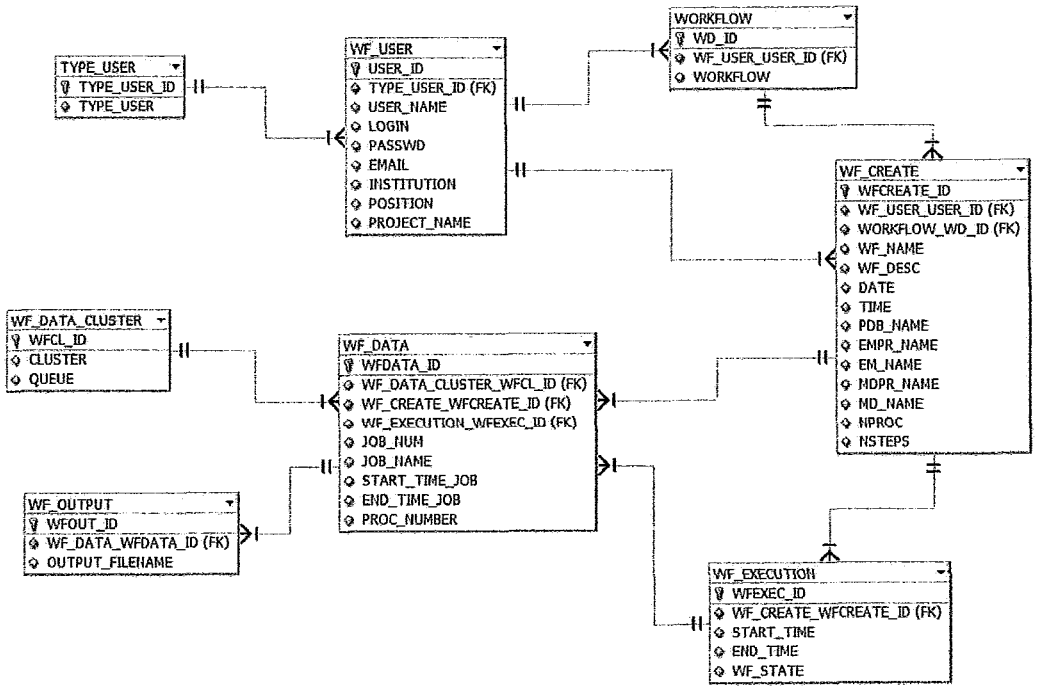

Figura 3.4 – Modelo lógico do módulo repositório de proveniência.

A classe **wf_user** representa as informações dos usuários cadastrados no banco de dados. A classe **type_user** através do atributo type_user classifica o tipo de usuário cadastrado na classe **wf_user**. Há três tipos de usuários que podem ser cadastrados – administrador, básico e avançado. O usuário do tipo administrador tem permissão de criar novos *workflows* e administrar todo o sistema. Os *workflows* criados pelo administrador são armazenados na classe **workflow**. Os usuários classificados como básico e avançado especifica o *workflow* utilizado através do atributo workflow e define os parâmetros utilizados no experimento. Os *workflows* definidos pelos usuários (básico e avançado) são armazenados na classe **wf_create**. Os atributos pertencentes a classe **wf_create** armazena as informações sobre a definição de um novo experimento, como nome do *workflow*, os dados de entrada que são utilizados e número de processadores, entre outros.

A classe **wf_execution** contém os atributos referentes ao tempo de execução do *workflow*, como data e hora de início e término da execução de um determinado

workflow. A classe **wf_data** registra através do atributo wf_name os dados gerados durante a execução de um determinado *workflow* no ambiente distribuído, como por exemplo, a data de inicio e término da execução da tarefa no ambiente distribuído, o nome da tarefa, o número da tarefa. A classe **wf_data_cluster** contém os atributos referentes ao ambiente distribuído, ou seja, em que nó do cluster a tarefa foi executada e em que fila. A classe **wf_output** armazena através do atributo wfdata_id as informações dos arquivos gerados durante a execução da tarefa remota de um determinado *workflow* no ambiente distribuído.

O processo de armazenamento dos dados de proveniência nesta arquitetura é muito importante, mesmo armazenando apenas alguns dados da definição do experimento e os dados gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído, pois somente através do registro destes dados é possível verificar o ocorrido durante a realização de um experimento e possibilitar ao pesquisador a realizar consultas aos dados referentes a um determinado experimento.

Capítulo 4

4 Experimentos de Dinâmica Molecular usando *Workflows* Científicos

No capítulo anterior foi apresentada uma arquitetura na qual o objetivo é auxiliar os pesquisadores de bioinformática a automatizar o processo de execução de seus experimentos de simulação de dinâmica molecular e possibilitar que seus experimentos sejam executados em ambientes distribuídos, assim como oferecer apoio à captura da proveniência dos dados da execução das tarefas remotas do *workflow* e também disponibilizar um ambiente para consulta aos dados armazenados.

Para validar a arquitetura proposta e os módulos desenvolvidos para a integração do SGWfC sequencial ao ambiente distribuído, foi definido um *workflow* para execução de experimentos de simulações de dinâmica molecular em ambientes distribuídos. O *workflow* faz parte da camada de aplicação apresentada na arquitetura.

O *workflow* definido apresenta uma forma genérica podendo ser integrado a qualquer um dos SGWfC apresentados no capítulo 2, entretanto para a modelagem do *workflow* foi escolhido o SGWfC Kepler (KEPLER, 2008). O Kepler foi escolhido por ser capaz de representar os modelos relacionados aos experimentos de bioinformática, apresentar uma solução para execução em ambientes distribuídos, como um componente que permite a conexão via SSH, além de possuir uma documentação clara e uma interface gráfica intuitiva.

Neste capítulo são definidas as etapas necessárias para a execução de um experimento de simulação de dinâmica molecular através de um *workflow* e também apresentado o *workflow* modelado no SGWfC Kepler.

4.1 GromDFlow – Um *Workflow* de Dinâmica Molecular

Os experimentos *in silico* geralmente são compostos pela execução de diversos programas encadeados. Os pesquisadores de bioinformática necessitam executar tais programas para realizarem seus experimentos. Para executar todos estes programas, os pesquisadores fazem uso de uma sequência de linhas de comando, ou seja, é necessário ficar com o terminal aberto executando etapa por etapa através de linhas de comando. A outra forma utilizada por eles são os *scripts*, geralmente desenvolvidos em Shell (SHELL, 2008) ou PERL (PERL, 2008), porém nem todos, necessariamente, têm conhecimentos em linguagens de programação para gerar *scripts*, fazendo com que eles gastem muito mais tempo para realizarem seus experimentos. A adoção da abordagem de *workflows* científicos é uma solução vantajosa para automatizar as tarefas realizadas pelos pesquisadores.

Uma típica simulação de dinâmica molecular envolve um conjunto de etapas que devem ser seguidas, tais como: a configuração de arquivos de entrada, definição de parâmetros, execução de programas locais, execução de programas remotos, monitoramento, coleta de dados de saída e de arquivos gerados durante a execução. Estas etapas levam o pesquisador a realizar diversas tarefas repetitivas, porém várias destas tarefas podem ser automatizadas. Uma estratégia utilizada nesta dissertação para automatizar e representar o experimento é encadear as tarefas através do desenvolvimento de um de *workflow* científico.

Para a definição das etapas a serem utilizadas no *workflow* de simulação de dinâmica molecular foi feito um levantamento junto aos pesquisadores do Laboratório de Física Biológica e do Laboratório de Modelagem e Dinâmica Molecular, ambos do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro, com o objetivo de verificar quais são as atividades desenvolvidas por eles e quais etapas de seus experimentos podem ser automatizadas. Para a realização da dinâmica molecular foi escolhido o pacote de programas GROMACS (GROMACS, 2008), que já é o utilizado pelo grupo. O GROMACS descrito no capítulo 2 contém uma série de programas para realizar simulações de dinâmica molecular.

Com o intuito de auxiliar os pesquisadores de bioinformática na automação de suas tarefas para realizarem os experimentos de simulações de dinâmica molecular, foi

definido um *workflow*, denominado **GromDFlow**. O *workflow* foi definido baseado na execução das etapas definidas pelos pesquisadores utilizando serviços de chamadas remotas para permitir sua execução em ambientes distribuídos.

4.1.1 Definição do GromDFlow

O GromDFlow (BARROS, 2008) foi concebido para detectar as mudanças conformacionais das proteínas, visando atender aos usuários que necessitam executar simulações de dinâmica molecular.

O GromDFlow está dividido em duas camadas, local e remota, conforme mostra Figura 4.1.

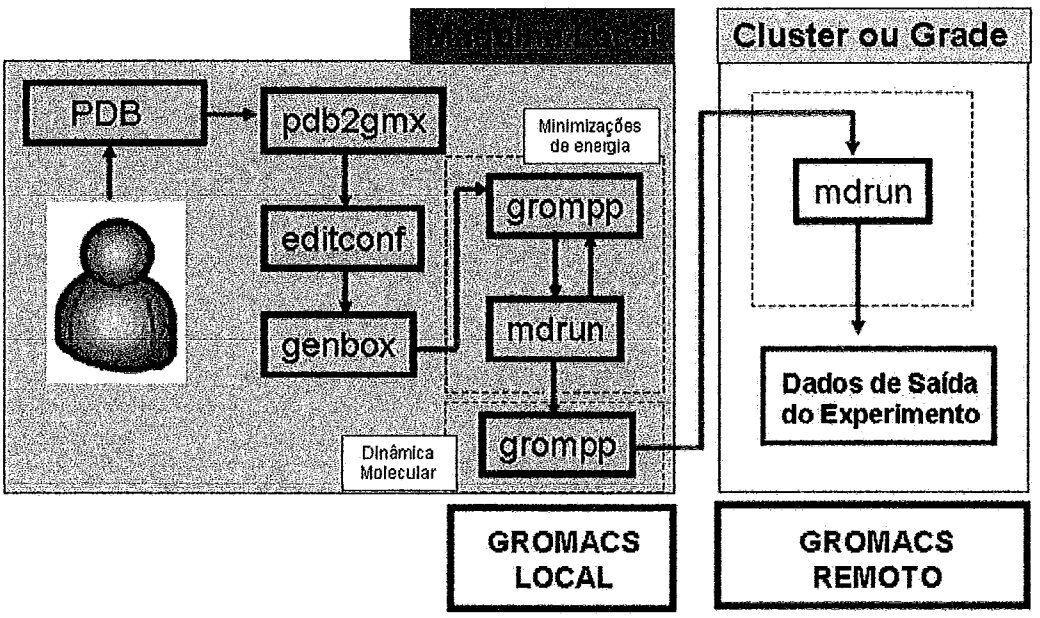

Figura 4.1 - Etapas do GromDFlow.

A primeira camada, denominada **GROMACSLocal**, realiza a execução local dos programas em um ambiente centralizado. A segunda camada, **GROMACSRemoto** é executada em um ambiente distribuído com capacidade de processamento de alto desempenho, como um cluster de PC ou uma grade computacional.

Na primeira camada do GromDFlow, GROMACSLocal, uma sequência de programas são executados, no entanto, esses programas são de rápido processamento podendo ser executados em um único computador. Para a implementação do

GromDFlow foram utilizados apenas alguns programas oferecidos pelo pacote GROMACS.

Os programas utilizados na camada local são:

- ***pdb2gmx*** – o programa converte arquivos *pdb* em arquivos de topologias e coordenadas, ou seja, ele usa como entrada um arquivo *.pdb* que contém a estrutura molecular da proteína e gera como saída os arquivos *.gro* e *.top*. O arquivo *.gro* é um arquivo que contém todas as coordenadas do sistema. O arquivo *.top* é um arquivo de topologia que associa os parâmetros do campo de forças necessário para uma descrição completa das iterações dos átomos da proteína. Os arquivos de entrada *.pdb* podem ser criados pelos próprios pesquisadores ou podem ser obtidos na página do PDB (PDB, 2007).

- ***editconf*** – gera em torno da proteína uma caixa, que é uma célula unitária que será repetida preenchendo todo o espaço, garantindo assim as condições periódicas de contorno. Através do *editconf* é possível definir qual tipo de caixa será utilizada e suas dimensões. Usa como entrada o arquivo *.gro* gerado pelo *pdb2gmx* e gera como saída um novo arquivo *.gro* com as dimensões da caixa.

- ***genbox*** – solvata o sistema, incluindo na caixa as moléculas do solvente. Usa como entrada o arquivo *.gro* gerado pelo *editconf* e gera como saída novos arquivos *.gro* e *.top* referentes ao sistema completo, ou seja, a proteína mais o solvente dentro da caixa definida.

Após a execução destes programas, iniciam-se os cálculos. Podem ser realizados três tipos de cálculos, que são:

- **minimização de energia** – sua função é ajustar os átomos das moléculas de um sistema até que os comprimentos de ligação e os ângulos estejam em uma configuração que corresponde a mínima energia potencial. Dois métodos podem ser aplicados na minimização de energia, o do máximo declive, conhecido como *steepest descent* e o do gradiente conjugado, ambos descritos no capítulo 2 desta dissertação.

- **minimização de energia com restrição de posição** – tem a mesma função da minimização de energia, porém com algumas moléculas ou átomos fixos em uma posição específica.

- **simulação da dinâmica molecular** – consiste em estudar a evolução temporal de um dado sistema molecular.

Todos estes cálculos utilizam os programas *grompp* e *mdrun*.

- ***grompp*** – concatena os dados dos arquivos de parâmetros de execução (*.mdp*), coordenadas (*.gro*) e topologias (*.top*) do sistema gerando um único arquivo de saída *.tpr*, que será utilizado como entrada para o próximo programa. O arquivo *.mdp* é um arquivo onde estão definidos todos os parâmetros necessários, como o número de passos, o tipo de simulação, a temperatura, entre outros, que são utilizados para o cálculo a ser realizado. Este arquivo é configurado pelo próprio usuário. Para cada cálculo é utilizado um arquivo *.mdp* diferente. O arquivo *.tpr* gerado como saída contém toda a informação necessária para se iniciar os cálculos da simulação.

- ***Mdrun*** – é o programa que realiza os cálculos. Utiliza como entrada o arquivo *.tpr* gerado pelo *grompp* e gera como saída os arquivos *.trr, .xtc, .edr, .gro* e *.log* com todos os resultados dos cálculos. Os arquivos *.trr* e *.xtc* contém as trajetórias do sistema, o *.gro* as coordenadas dos últimos quadros gerados, o *.edr* possui as informações sobre as energias do sistema e o *.log* as informações sobre o andamento da simulação.

Os dois primeiros cálculos, minimização de energia com restrição da posição e minimização de energia, são executados na camada local – GROMACSLocal, pois suas execuções possuem um rápido processamento.

A segunda camada, denominada GROMACSRemoto, é onde é realizada a simulação de dinâmica molecular. Tal simulação consome muito tempo computacional, geralmente dias, sendo necessário executá-la em um ambiente com capacidade de processamento de alto desempenho, como um cluster de PC ou uma grade computacional.

Embora a implementação do GromDFlow através de *shell scripts* (SHELL, 2008) traga benefícios aos pesquisadores de bioinformática que têm conhecimentos em linguagens de programação para automatizar o processo de seus experimentos *in silico*, o uso desses *scripts* não permite o monitoramento da execução do experimento em ambientes distribuídos e nem apoio à captura dos dados de proveniência gerados durante o experimento. No entanto, o uso de um SGWfC possibilita a modelagem dos experimentos *in silico* de bioinformática e permite a execução coordenada das etapas do experimento, além de oferecer apoio à captura dos dados gerados durante o mesmo.

4.1.2 Modelagem do GromDFlow

Depois de definidas as etapas necessárias para a construção do *workflow* foi possível iniciar a modelagem do *workflow* em um SGWfC. O SGWfC utilizado para modelar os *workflows* definidos nesta dissertação foi o Kepler (KEPLER, 2008) pelos motivos já mencionados. Além disso, o Kepler também oferece mecanismos para a construção e inclusão de novos componentes que podem ser agregados à sua biblioteca de artefatos e utilizados nos experimentos.

O *workflow* foi modelado fazendo uso dos componentes básicos do Kepler. O diretor utilizado é o *SDFDirector*, sua função é supervisionar os *workflows*, determinando a ordem da invocação de um ator. Os atores são componentes que representam as etapas de um *workflow*. Outros componentes do Kepler também foram utilizados, dentre eles, estão o *SSHSession* que cria uma conexão SSH; *Parameter* que recebe os parâmetros passados ao *workflow* e *ExecutionCmd* que executa um comando. As portas de entrada e saída são responsáveis pela ligação de um *subworkflow* a outro.

Três atores iniciais foram definidos no GromDFlow, conforme Figura 4.2. O primeiro ator **Conexão** abre uma conexão SSH para o computador onde será executada a primeira camada do *workflow*, a GROMACSLocal, e para onde os arquivos necessários para a sua execução serão transferidos.

O segundo ator é um *CompositeActor* do Kepler, chamado **GROMACSLocal** (Figura 4.3). Um *CompositeActor* é uma agregação de atores, ou seja, ele permite modelar e executar *subworkflows* dentro de um fluxo de tarefas de um *workflow*. Este *subworkflow* contém um conjunto de etapas a serem executadas.

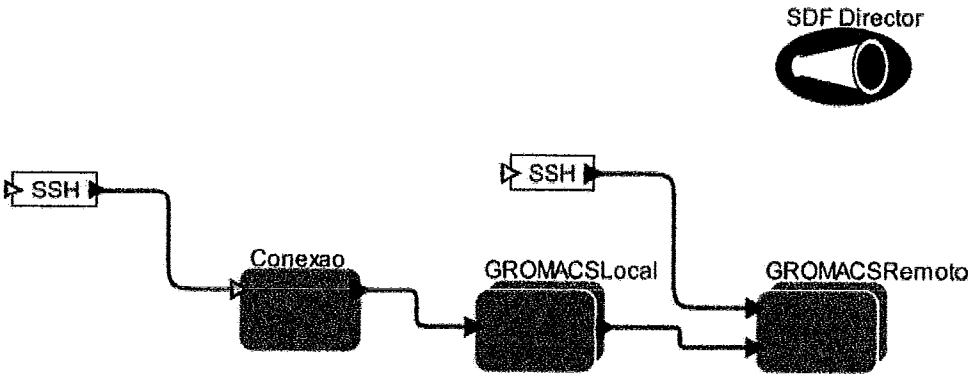

Figura 4.2 – GromDFlow no Kepler.

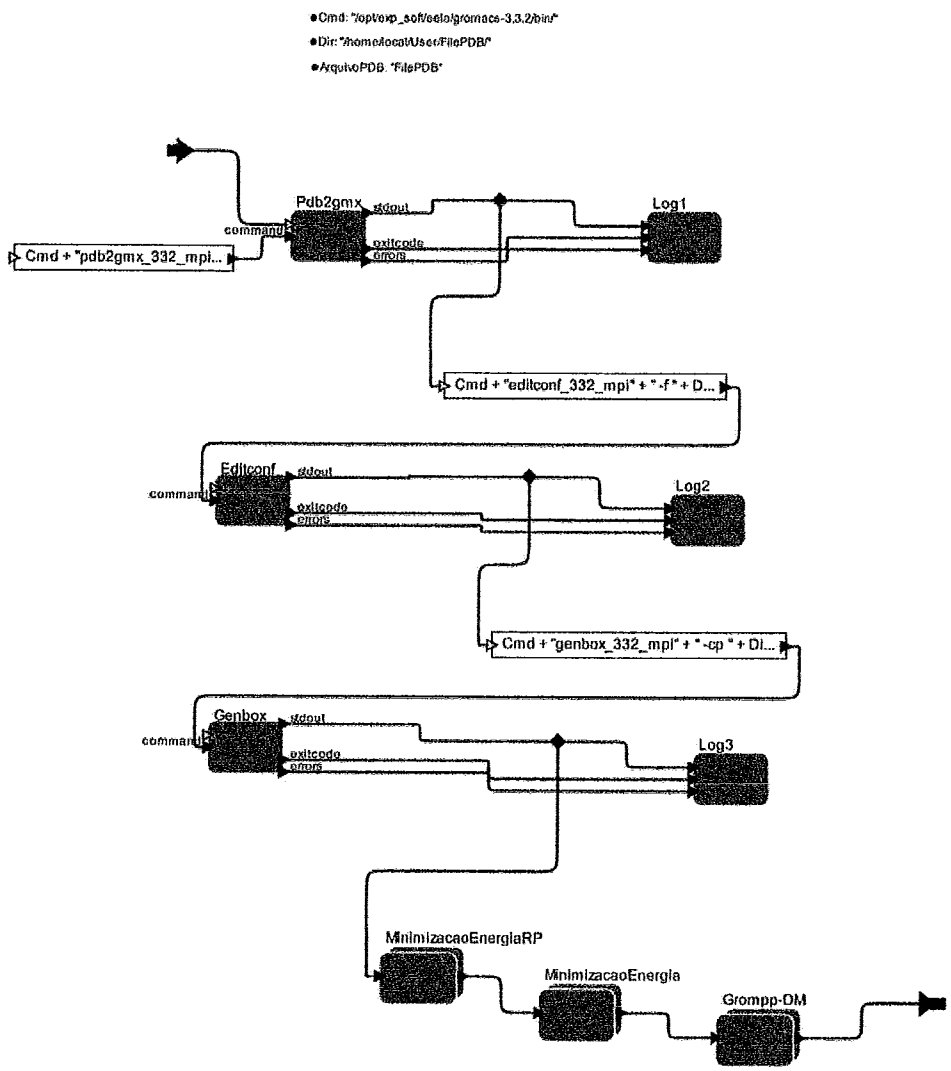

Figura 4.3 – *Subworkflow* GROMACSLocal.

O *subworkflow* **GROMACSLocal** consiste na execução dos programas do pacote GROMACS definidos na camada local. Este *subworkflow* prepara os arquivos para a simulação usando os programas *pdb2gmx*, *editconf* e *genbox*. Cada programa é representado por um ator do Kepler. Após estas etapas, iniciam-se os cálculos. Os *subworkflows MinimizacaoEnergiaRP* e *MinimizacaoEnergia* executam os cálculos de minimização de energia com restrição da posição e minimização de energia. O *subworkflow Grompp-DM* prepara o arquivo de entrada que será utilizado para a execução da simulação da dinâmica molecular.

As figuras (Figura 4.4 e Figura 4.5) apresentam os *subworkflows* com as etapas referentes à minimização de energia com restrição da posição e a minimização de energia, respectivamente. Os programas utilizados são o *grompp* e o *mdrun*. Estas etapas são executadas localmente.

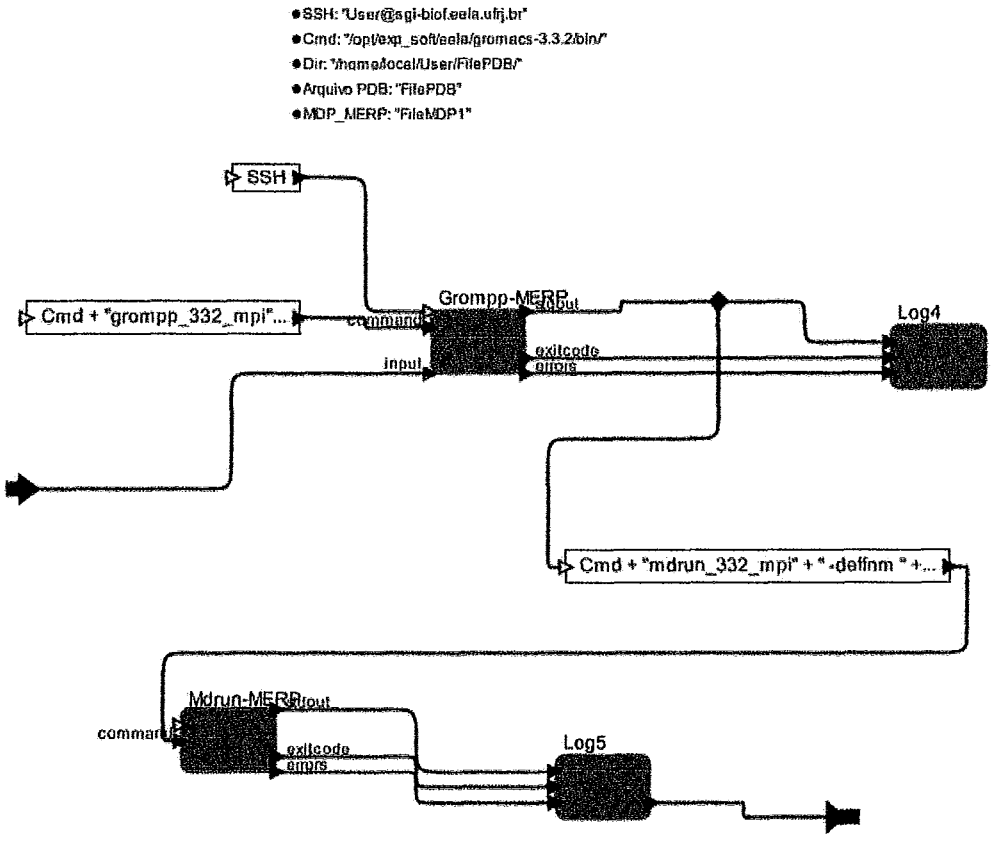

Figura 4.4 – *Subworkflow* da minimização de energia com restrição da posição.

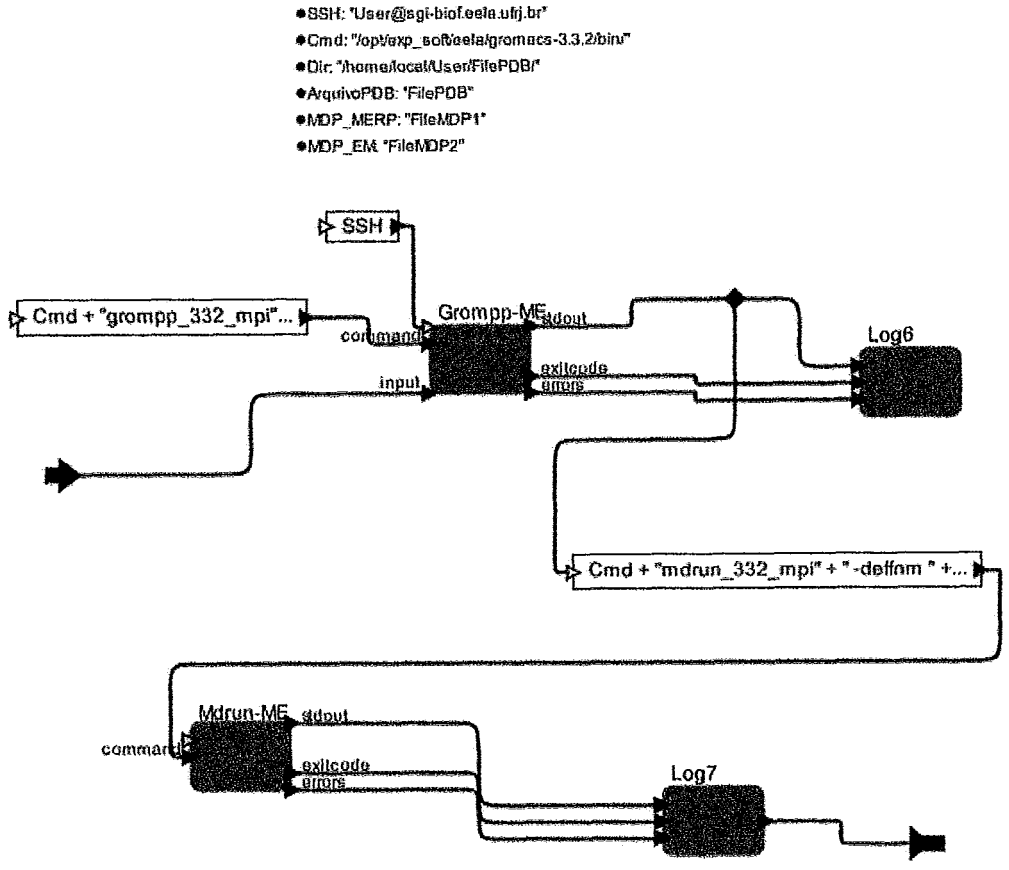

Figura 4.5 - *Subworkflow* da minimização de energia.

A Figura 4.6 apresenta o *subworkflow Grompp-DM*. Neste *subworkflow* é executado o programa *grompp,* ele gera o arquivo *.tpr* que será utilizado como entrada para a execução da simulação da dinâmica molecular.

Para execução do *workflow* é necessário que sejam informados alguns parâmetros. Alguns destes parâmetros possuem valores fixos sem a necessidade de intervenção do usuário, porém outros são necessários que sejam informados. As informações dos parâmetros utilizados no *workflow* são passadas através do portal *web*. Alguns dos parâmetros utilizados no GromDFlow são:

- *SSH* – endereço da máquina onde será executado.
- *Cmd* – diretório onde estão os arquivos executáveis dos programas.
- *Dir* – diretório onde estão os arquivos que serão utilizados na execução.

- *ArquivoPDB* – arquivo PDB que contém a estrutura da proteína a ser estudada.

- *MDP_MERP* – arquivo *.mdp* para execução da minimização de energia com restrição da posição.

- *MDP_ME* – arquivo *.mdp* para execução da minimização de energia.

- *MDP_DM* – arquivo *.mdp* para execução da dinâmica molecular.

- *NP* – número de processadores utilizado para executar a simulação.

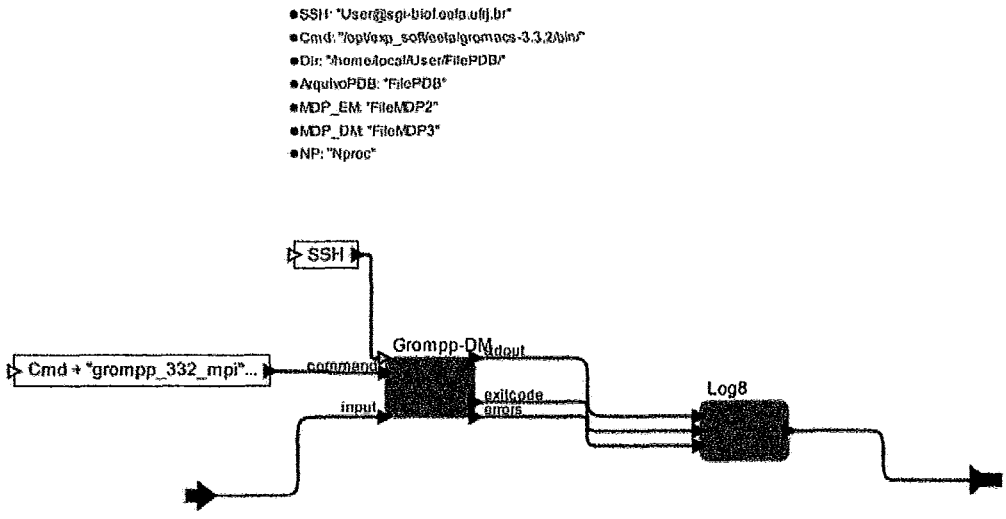

Figura 4.6 - *Subworkflow Grompp-DM.*

Nos atores *Log* são registrados os *logs* referentes a execução, ou seja, de todas as etapas que são executadas pelo *workflow*, assim como as mensagens de erro que possam ocorrer durante a execução do *workflow*.

O último *CompositeActor* do GromDFlow, é o *subworkflow* **GROMACSRemoto**. É nesta etapa do *workflow* que a simulação da dinâmica molecular é realizada. Esta simulação leva um longo tempo de processamento e requer uma capacidade computacional de alto desempenho, como um cluster de PC ou uma grade computacional. É a partir do *subworkflow* GROMACSRemoto que é disparada uma mensagem para realizar a comunicação entre os módulos desenvolvidos na arquitetura, apresentada no capítulo 3 desta dissertação.

Para a realização desta etapa houve a necessidade da criação de um novo ator no Kepler. O Kepler permite a construção de novos componentes que podem ser agregados à sua biblioteca de artefatos e utilizados nos experimentos. O ator foi desenvolvido utilizando a linguagem Java, tendo em vista que o SGWfC Kepler é baseado em Java. Para o ator desenvolvido foi gerado um arquivo .kar que contém as definições do ator. Este arquivo é importado para a biblioteca de componentes do Kepler e fica disponível para ser utilizado. O ator desenvolvido foi chamado de ***JobExecutor***, conforme mostra a Figura 4.7.

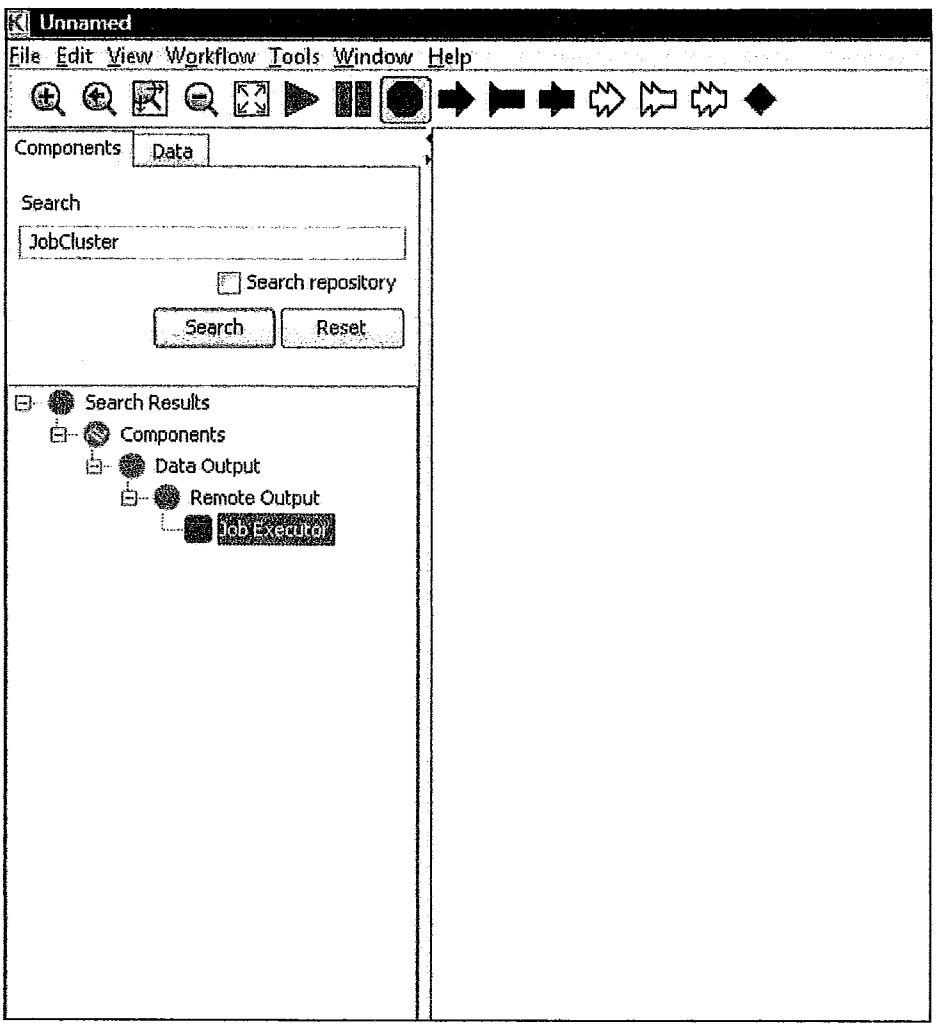

Figura 4.7 – Interface do Kepler com o ator JobExecutor.

O ***JobExecutor*** é o ator responsável por enviar as tarefas remotas do *workflow* ao ambiente distribuído e efetuar a comunicação entre os módulos. Após o inicio da execução da tarefa remota do *workflow* em um ambiente distribuído, o controle tanto da

execução da tarefa que foi submetida quanto o seu monitoramento é perdido, com isso os pesquisadores ficam sem saber o que está acontecendo com a tarefa que foi submetida e de terem todas as informações que são geradas durante a execução do *workflow* no ambiente distribuído, por isso a necessidade dos módulos servidor de proveniência, cliente de proveniência e repositório de proveniência, desenvolvidos na arquitetura. O cliente de proveniência monitora a execução da tarefa no ambiente distribuído e envia as informações para o servidor de proveniência. Eles fazem tanto o monitoramento da tarefa quanto a captura dos dados gerados no ambiente distribuído. Os dados capturados são armazenados no repositório de proveniência.

A Figura 4.8 apresenta o *subworkflow* GROMACSRemoto. Através do ator *JobExecutor* são passados os parâmetros necessários para a execução do *workflow* no ambiente distribuído.

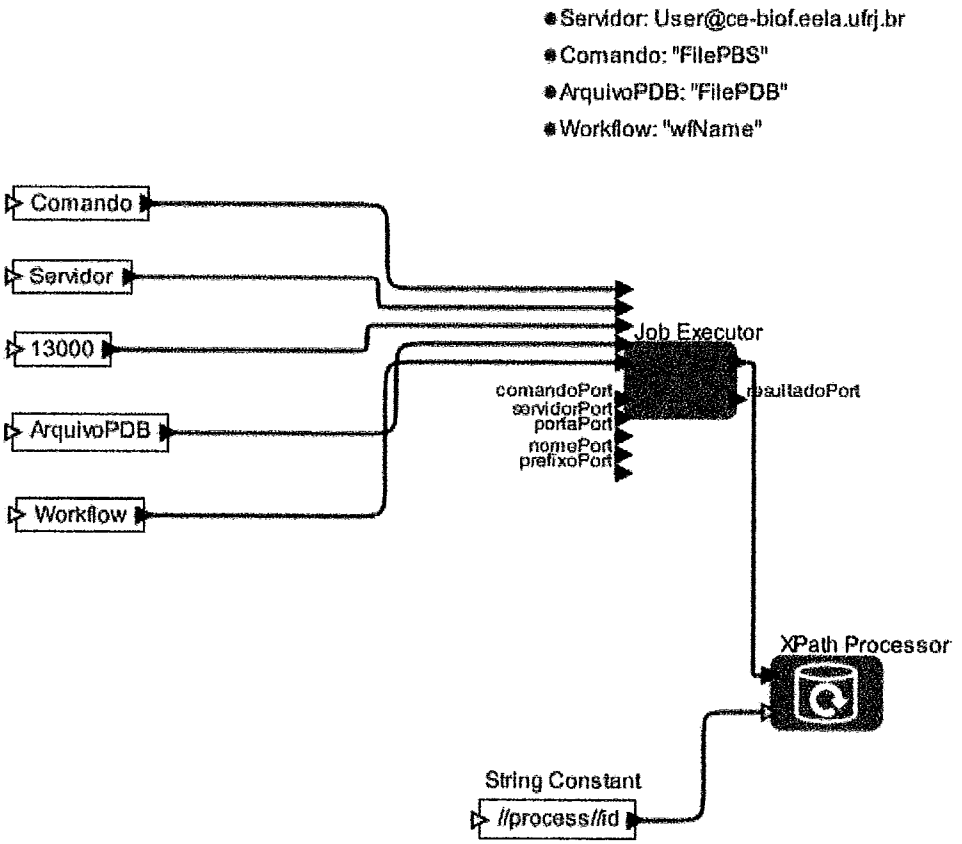

Figura 4.8 - *Subworkflow* GROMACSRemoto.

Os parâmetros passados para o *JobExecutor* são:

- Nome do servidor o qual será conectado;
- Número da porta para a conexão remota, neste caso, foi utilizado a porta 13000 para fazer a conexão. Vale ressaltar, que a porta 13000 foi escolhida aleatoriamente, podendo ser utilizada outra.
- Nome do comando submetido ao ambiente distribuído.
- Nome do arquivo PDB submetido pelo usuário, sendo através dele que é feita a identificação dos arquivos que são gerados ao longo da execução do *workflow* no ambiente distribuído, pois todos os arquivos gerados apresentam o prefixo do nome do arquivo PDB.

O ator *JobExecutor* retorna, após a sua execução, informações codificadas no padrão XML (W3C, 2008). Este formato foi escolhido devido ao Kepler possuir um ator chamado *XPathProcessor*, que é capaz de processar informações vindas no padrão XML.

O *subworkflow* GROMACSRemoto poderia ser executado tanto em um ambiente de cluster de PC quanto em um ambiente de grade computacional, porém por motivos de disponibilidade para testes e configuração dos programas utilizados no GromDFlow, ele foi executado apenas no ambiente de cluster de PC.

A submissão de uma tarefa para ser executada em um ambiente de cluster é feita através de um sistema de fila. O PBS (*Portable Batch System*) é um sistema de filas utilizado para gerenciar o escalonamento de tarefas (*jobs*) remotas em um cluster (TORQUE, 2009).

No GromDFlow, a submissão é feita através de um *script* PBS, onde nele são informados os dados necessários para a execução da tarefa remota, como por exemplo em qual fila do cluster a tarefa será executada. O *script* é criado com base nas informações pré-definidas e nos parâmetros definidos pelo usuário. Após a submissão da tarefa ao cluster, o cliente de proveniência recebe uma notificação de que uma nova tarefa está sendo executada e então inicia o monitoramento da execução e a captura das informações que são geradas durante a execução. As informações são enviadas ao

servidor de proveniência, onde são tratadas e então armazenadas no repositório de proveniência.

4.1.3 Variações do GromDFlow

Com o objetivo de permitir que os pesquisadores que possuem mais experiência em realizar simulações de dinâmica molecular, foi também modelado o *workflow* GromDFlow Avançado.

GromDFlow Avançado visa atender os pesquisadores que necessitam executar simulações de dinâmica molecular mais complexas. O que diferencia o GromDFlow Avançado do GromDFlow é o número de etapas que são realizadas ao longo da execução do experimento.

A Figura 4.9 mostra trecho do *subworkflow* da camada local do GromDFlow Avançado.

Na camada local, o GromDFlow Avançado executa os programas: *pdb2gmx*, *editconf*, *genbox*, *grompp* e *genion*. Os programas *pdb2gmx*, *editconf* e *genbox* são os mesmos utilizados no GromDFlow. Na etapa de preparação dos arquivos para a simulação, o programa ***grompp*** adiciona íons para equilibrar a carga do sistema, utiliza como entrada os arquivos *.mdp*, *.gro* e *.top* e gera como saída um arquivo *.tpr* com as informações dos íons. O programa ***genion*** gera os íons, usa como entrada o arquivo *.tpr* gerado pelo *grompp* e gera novos arquivos de coordenadas e topologias com os íons gerados.

Além dos três *subworkflows* do GromDFlow que executam os cálculos de minimização de energia com restrição da posição, minimização de energia e a simulação da dinâmica molecular, há também no GromDFlow Avançado o *subworkflow DinamicaMolecularRP* que executa a simulação da dinâmica molecular com restrição da posição com o objetivo de ajustar o solvente em torno da proteína em estudo.

A camada remota, GROMACSRemoto, do GromDFlow Avançado possui as mesmas etapas e funções do GromDFlow já apresentadas na seção 4.2.

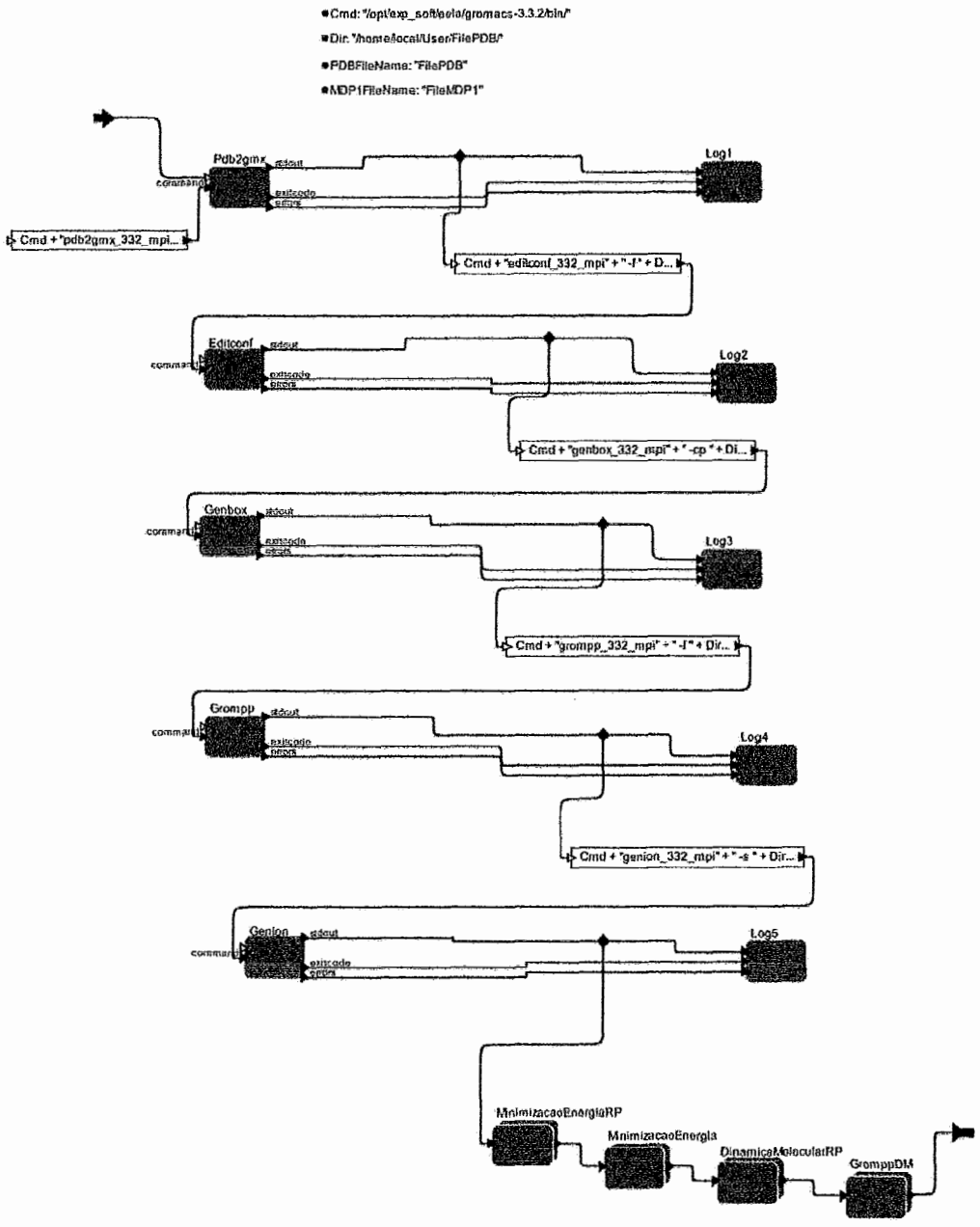

Figura 4.9 – Trecho do *subworkflow* GROMACSLocal do GromDFlow Avançado.

Capítulo 5

5 Experimentos

Neste capítulo são apresentados os experimentos que foram realizados durante o desenvolvimento desta dissertação. Foram realizados três tipos de experimentos, o primeiro utilizando um ambiente de cluster de PC, o segundo utilizando um ambiente de grade computacional e o terceiro utilizando a arquitetura proposta nesta dissertação.

5.1 Execução distribuída em um Cluster de PC

O primeiro experimento foi realizado com o objetivo de avaliar o *workflow* de dinâmica molecular GromDFlow, modelado no SGWfC Kepler (apresentado no capítulo 4). Também fez parte do experimento a análise de desempenho do pacote de programas GROMACS através do GromDFlow, com o intuito de verificar qual o desempenho apresentado por ele.

O experimento foi executado através do *workflow* GromDFlow, porém quando foi realizado, o GromDFlow ainda não possuía os módulos de controle da execução do *workflow* no ambiente distribuído e nem da captura dos dados de proveniência.

5.1.1 Ambiente Computacional

O ambiente computacional utilizado para realizar os experimentos consiste em um computador local e um cluster de PC. O computador local é composto de um processador quad-core Intel Xeon 2.33 GHz com 8GB de memória RAM e disco rígido de 160GB. O cluster de PC é composto de onze nós de processamento, onde cada nó possui quatro processadores Intel Xeon 2.33GHz duo processados, totalizando oito, com 8GB de memória RAM cada nó, conectados através de uma rede Gigabit Ethernet. O cluster utilizado para realização dos experimentos pertence ao Laboratório de Física Biológica do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro.

O sistema operacional utilizado no cluster é o *Scientific* Linux 4. A versão do MPI (*Message Passing Interface*) instalado nos nós do cluster é o MPICH2-1.0.6. Os experimentos foram realizados utilizando a versão 3.3.2 do pacote de programas GROMACS. A versão do Kepler utilizada para modelar o GromDFlow foi a 1.0.0.

5.1.2 Estudo de caso

Para a realização do experimento através do *workflow* GromDFlow foi utilizado um conjunto de seis proteínas, que são: a toxina peptídica w-Aga-IVB que consiste no estudo de uma toxina isolada do veneno de uma espécie de aranha (YU *et al.*, 1993) e o conjunto de proteínas disponíveis para testes na página do GROMACS (GROMACS, 2008). O conjunto consiste nas seguintes proteínas: d.villin, d.dppc, d.poly-ch2, d.lzm/cut, d.lzm/pme. Cada uma destas proteínas apresenta diferentes estruturas moleculares, complexidade e quantidade de átomos e é representada por um arquivo PDB distinto. Estas proteínas foram escolhidas para realização dos experimentos iniciais, por serem proteínas pequenas e de rápido processamento (CRUZ *et al.*, 2008).

As simulações de dinâmica molecular dependem do número de átomos e dos passos que são calculados. O experimento realizado calcula 5000 interações, que é o equivalente a 10 picosegundos em uma temperatura de 300K, utilizando o campo de força GROMOS (VAN GUNSTEREN e BERENDSEN, 1987). Todos os cálculos foram feitos usando a precisão simples do GROMACS. Vale ressaltar que este estudo de caso foi realizado utilizando um número pequeno de iterações, pois um dos objetivos é apenas analisar o *workflow* GromDFlow, atualmente os cálculos são efetuados na ordem de nanosegundos.

A Figura 5.1 apresenta parte do *workflow* GromDFlow para a execução da simulação da dinâmica molecular da proteína d.villin utilizada no experimento.

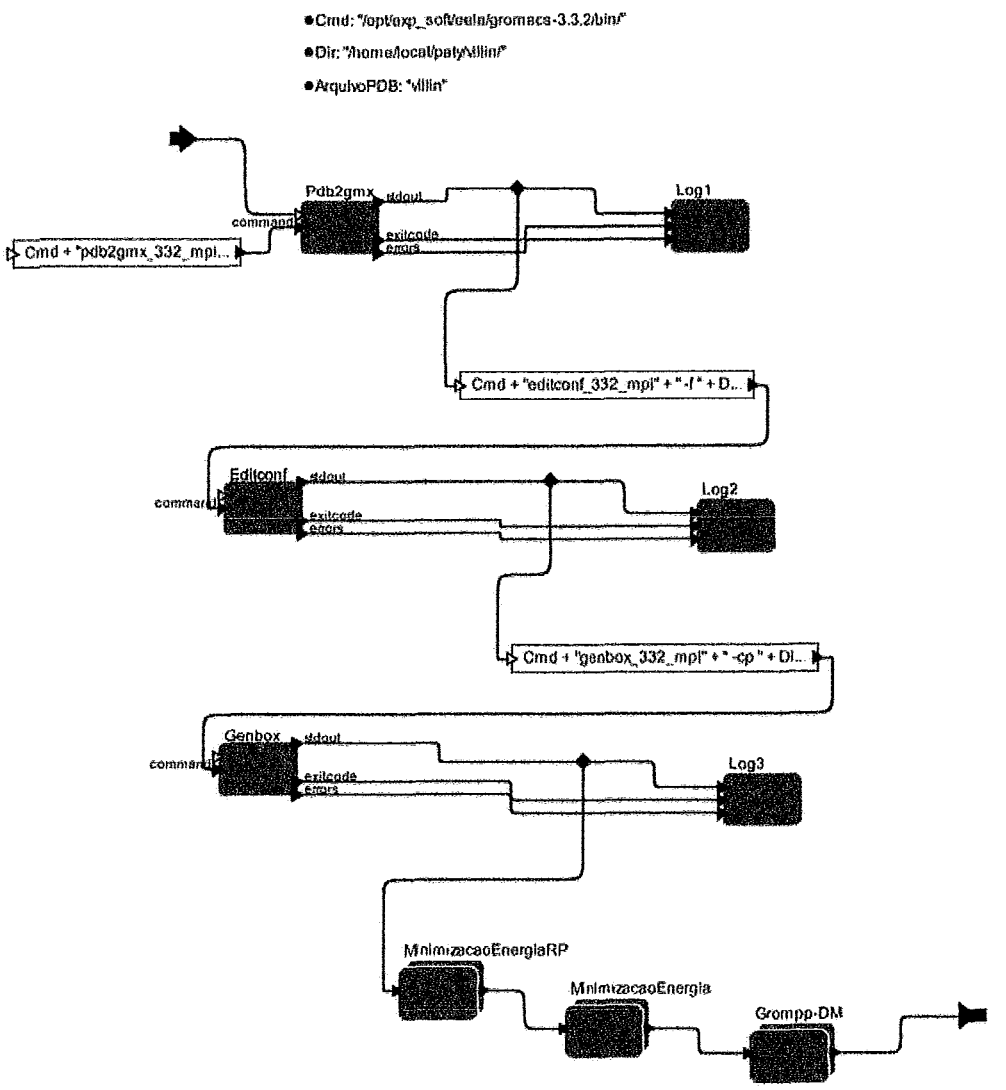

Figura 5.1 – Parte do *workflow* GromDFlow com a proteína d.villin.

5.1.3 Resultados

Parte da execução do *workflow* GromDFlow, que consiste na camada local, GROMACSLocal, foi executada em um ambiente centralizado, ou seja, em um único computador, pois sua execução apresenta um rápido processamento. A segunda camada do *workflow* GromDFlow, a GROMACSRemoto, foi executada em um cluster de PC.

A avaliação consiste na execução de tarefas independentes, onde cada nó do cluster executa uma tarefa em paralelo. As tarefas foram executadas em paralelo usando 1, 2, 4 e 8 processadores por nó, a comunicação entre os processadores é feita através do

uso do MPI (MPICH2, 2008). Cada nó é responsável pela execução de uma tarefa de um experimento de simulação de dinâmica molecular de uma dada proteína. As tarefas foram invocadas através do *workflow* GromDFlow e enviadas à fila do cluster. Para uma melhor compreensão, as proteínas foram numeradas de um a seis, sendo: toxina da aranha w-Aga-IVB (*ptn1*), d.villin (*ptn2*), d.polly-ch2 (*ptn3*), d.lzm/cut (*ptn4*), d.dppc (*ptn5*) e d.lzm/pme (*ptn6*).

A Tabela 5.1 apresenta a média dos tempos de execução obtidos com sucessivas invocações do *workflow* GromDFlow. Os tempos são apresentados em segundos.

Tabela 5.1 - Tempo de execução em segundos por processador.

Número de processadores /nó	**ptn1**	**ptn2**	**ptn3**	**ptn4**	**ptn5**	**ptn6**
1	1894,0	545,0	661,0	4164,0	2499,0	5900,0
2	1022,0	541,0	359,0	2337,0	2043,0	3419,0
4	601,0	478,0	211,0	1355,0	1470,0	1959,0
8	417,0	411,0	170,0	1001,0	1212,0	1478,0

Os resultados são considerados muito satisfatórios. Para uma melhor visualização da redução dos tempos de processamento, a Figura 5.2 apresenta o gráfico com os tempos normalizados das execuções para cada proteína apresentada na Tabela 5.1. Através do gráfico podemos observar uma queda significativa nos tempos de execução, conforme o número de processadores vai aumentando.

A Figura 5.3 apresenta o gráfico com a curva de aceleração linear obtida para as mesmas proteínas da Tabela 5.1. A maioria das execuções obtém ganhos significativos. O melhor caso é para a *ptn1*, onde o tempo de execução baixou de 31,57 minutos para 6,59 minutos com oito processadores.

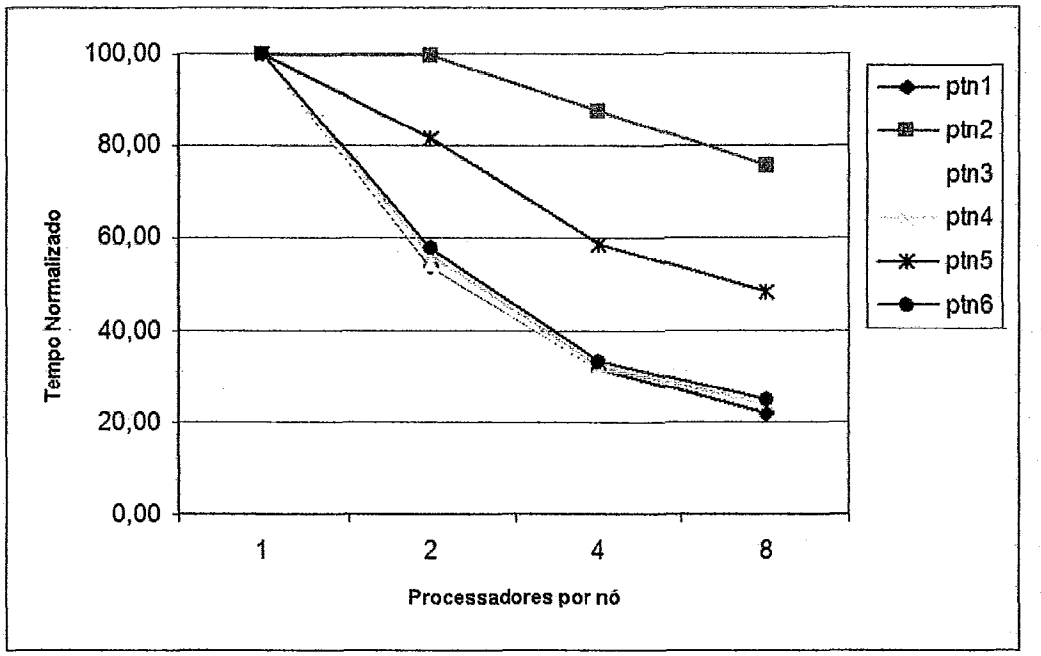

Figura 5.2 – Tempo normalizado das execuções dos experimentos.

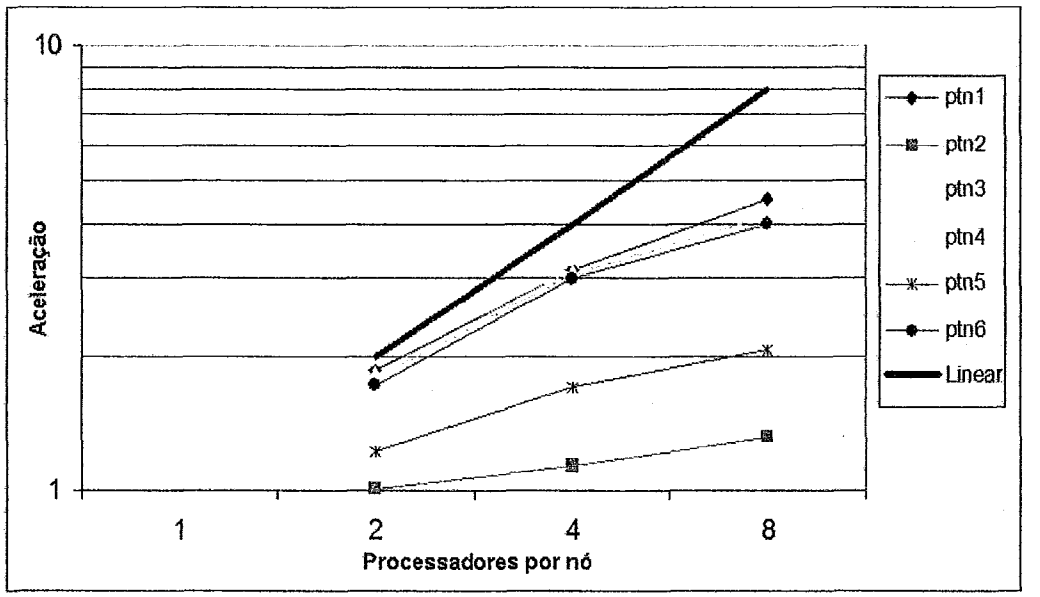

Figura 5.3 – Aceleração dos tempos das execuções dos experimentos.

Através do experimento executado pelo *workflow* GromDFlow, foi possível perceber o ganho significativo no tempo de execução, pois o tempo total de execução para as seis proteínas (*ptn1*, *ptn2*, *ptn3*, *ptn4*, *ptn5* e *ptn6*) diminuiu de aproximadamente de 4,35 horas para 1,30 horas.

Após análise dos experimentos, pode se observar que as execuções realizadas utilizando acima de oito processadores apresentaram problemas de desempenho. De acordo com KUTZNER *et al.* (2007), o GROMACS apresenta problemas de desempenho em conexões Gigabit Ethernet com experimentos realizados utilizando acima de oito processadores. O problema está relacionado ao congestionamento de rede na troca de mensagens entre os nós. Porém, segundo KUTZNER *et al.* (2007), o GROMACS escala bem em clusters de PC que possuem conexões do tipo Myrinet ou Infiniband devido às características de alta taxa de transmissão e baixa latência. Como no momento da realização dos experimentos, não possuíamos nenhum cluster de PC com conexões Myrinet ou Infiniband, não foi possível verificar tal afirmação. Considerando os resultados obtidos, foi possível verificar que a submissão de tarefas utilizando *workflows* científicos em um ambiente distribuído, como um cluster de PC, reduz altamente o tempo de execução das simulações de dinâmica molecular.

5.2 Execução distribuída em uma Grade Computacional

O segundo experimento foi realizado com o objetivo de conhecer quais etapas são necessárias para executar um experimento de simulação de dinâmica molecular em um ambiente de grade computacional e de verificar a comunicação entre o SGWfC sequencial e o ambiente de grade computacional. Os experimentos foram realizados sem a utilização dos módulos desenvolvidos na arquitetura, pois o objetivo era apenas verificar como seria a comunicação entre o SGWfC e o ambiente de grade computacional.

5.2.1 Ambiente Computacional

O ambiente computacional utilizado para realizar os experimentos foi a grade computacional pertencente ao projeto EELA (*E-Infrastructure shared between Europe and Latin America*) (EELA, 2009). O projeto EELA consiste na implantação de uma infra-estrutura de grade computacional compartilhada por países da Europa e América Latina. O objetivo é criar uma rede de colaboração para compartilhar uma infra-estrutura de grade computacional entre as instituições que fazem parte do projeto.

Uma das características é a utilização do *middleware* Glite. O Glite é um conjunto integrado de componentes destinados a permitir o compartilhamento de recursos. Foi criado para facilitar a alocação e escalonamento de recursos distribuídos (GLITE, 2009).

O ambiente de grade computacional EELA foi escolhido por ser um dos projetos em que o Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro está envolvido. O Instituto disponibiliza um conjunto de máquinas que fazem parte da grade computacional do EELA. Os experimentos foram realizados utilizando a versão 3.0 do *middleware* glite e a versão 3.3.2 do pacote de programas GROMACS. A versão do Kepler utilizada para modelar o *workflow* foi a 1.0.0.

5.2.2 Estudo de caso

Para a realização dos experimentos de simulação de dinâmica molecular em um ambiente de grade computacional foi utilizado o arquivo PDB da proteína d.villin. A proteína faz parte do conjunto de proteínas disponíveis para testes na página do GROMACS (GROMACS, 2008). Esta proteína foi escolhida por ser uma proteína pequena e de rápido processamento, pois o objetivo era de apenas conhecer as etapas necessárias para executar um experimento na grade.

Primeiramente, para ter acesso a grade computacional é necessário atender alguns requisitos. A primeira etapa consiste em passar por um processo de certificação. A solicitação do certificado é feita através de um formulário a uma autoridade certificadora (CA). O certificado é então validado por uma autoridade de registro (RA), que após validar, o envia para o usuário. Após o recebimento do certificado, é necessário fazer um cadastro em uma Organização Virtual e solicitar uma conta em uma interface de usuário (máquina de acesso à grade).

Para submeter uma tarefa na grade é preciso criar um arquivo com a descrição da tarefa. A Figura 5.4 mostra o arquivo jdl (*job description language*) para submissão de uma tarefa de simulação de dinâmica molecular com a proteína d.villin. O processo de execução de uma tarefa na grade computacional é feito através da execução de algumas linhas de comando, como submeter uma tarefa, monitorar o andamento da execução e obter os resultados.

```
Executable   = "gromacs.sh";
StdOutput    = "gromacs.out";
StdError     = "gromacs.err";
InputSandbox = {"gromacs.sh", "villin.tgz"};
OutputSandbox = {"gromacs.out","gromacs.err",,"topol.edr","topol.gro","topol.log","topol.trr"};
```

Figura 5.4 – Descrição do arquivo jdl

O arquivo jdl contém o nome do arquivo executável (*Executable*), o nome dos arquivos que são gerados contendo as mensagens de saída ou as de erro (*StdOutput e StdError*), os nomes dos arquivos de entrada que são utilizados (*InputSandbox*) e os nomes dos arquivos de saída que são gerados (*OutputSandbox*). O experimento utilizou os mesmos parâmetros definidos no ambiente de cluster de PC. Todos os cálculos foram feitos usando a precisão simples do GROMACS.

5.2.3 Resultados

Os experimentos foram realizados com o objetivo de verificar a comunicação entre o SGWfC sequencial Kepler e o ambiente de grade computacional e de conhecer quais etapas são necessárias para execução de um experimento de simulação de dinâmica molecular no ambiente de grade computacional. Primeiramente, os testes foram realizados executando as linhas de comando através de um terminal do sistema operacional Linux, com o intuito de conhecer todos os comandos e parâmetros necessários para submeter e executar uma tarefa na grade. Após esta etapa, foi possível definir o que é necessário para que o experimento possa ser executado na grade. Foi então, criado um *script shell* (SHELL, 2008) com todos os comandos e parâmetros utilizados.

O experimento foi realizado utilizando somente parte do *workflow* GromDFlow. Apenas a camada local do GromDFlow, a GROMACSLocal foi executada no ambiente centralizado. Para executar a camada remota foi necessário modelar um novo *workflow*. O *workflow* foi modelado fazendo uso dos componentes básicos do Kepler. Os atores utilizados foram o *Parameter* que recebe os parâmetros passados ao *workflow*, o *ExecutaScript* que executa um comando e o *Display* que exibe os resultados da execução.

No ator *Proxy*, é passado a linha de comando para iniciar um processo na grade computacional. Através do ator *ExecutionCmd* é submetido o *script* criado para execução dos comandos necessários para submeter uma tarefa remota à grade. A Figura 5.5 mostra o *script* criado para submissão da tarefa remota à grade computacional. O *script* contém a localização dos diretórios onde estão os arquivos executáveis (comandos) e a linha de comando de submissão de uma tarefa à grade.

```
#!/bin/bash
EDG_WL_LOCATION=/opt/edg
export EDG_WL_LOCATION
$EDG_WL_LOCATION/bin/edg-job-submit -o teste01 gromacs.jdl
```

Figura 5.5 – *Script* para submissão da tarefa à grade computacional.

A Figura 5.6 apresenta o *workflow* modelado no SGWfC Kepler para executar a simulação da dinâmica molecular com a proteína d.villin no ambiente de grade computacional.

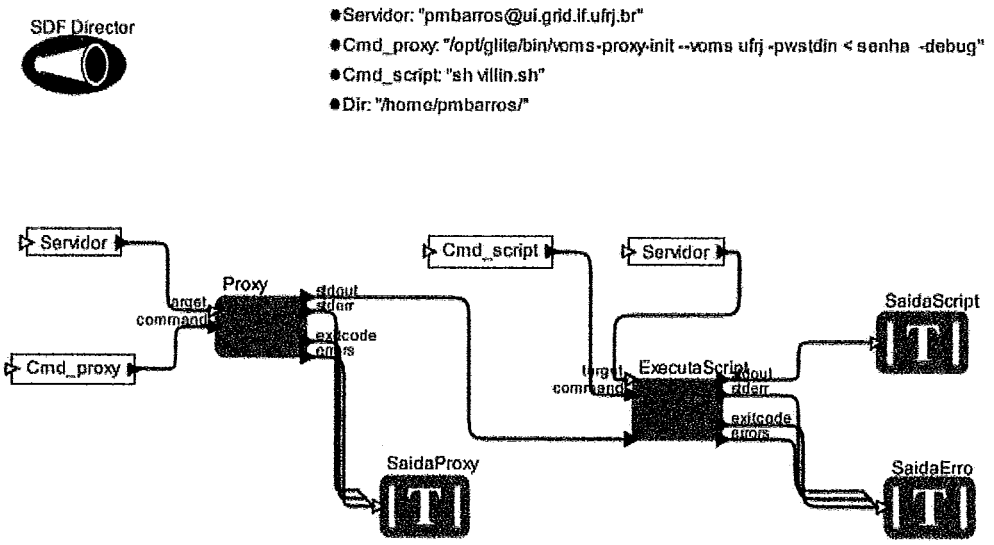

Figura 5.6– *workflow* para execução no ambiente de grade computacional.

O *workflow* consiste na execução de um experimento de simulação de dinâmica molecular no ambiente de grade computacional. Alguns dos parâmetros necessários para a execução do *workflow* são identificados através dos parâmetros:

- *Servidor* – identificação do usuário e nome da máquina que é utilizada como interface de usuário.
- *Cmd_proxy* – comando para iniciar um processo na grade.
- *Cmd_script* – comando para submissão do *script* e o nome do *script* que será executado.
- *Dir* – diretório onde estão localizados os arquivos necessários para execução.

Os arquivos utilizados para a realização do experimento foram transferidos manualmente para a máquina que é usada como interface de usuário no ambiente de grade computacional. Após a modelagem do *workflow* no Kepler, foi possível concluir que a submissão de tarefas à grade através de um SGWfC ainda apresenta algumas barreiras. A submissão da tarefa, só foi possível com a execução de um *script* enviado através do Kepler, porém o objetivo dos sistemas de gerência de *workflows* científicos é a não utilização de *scripts* para realização de experimentos. Apesar de o SGWfC Kepler dispor de alguns atores para submissão de tarefas na grade, não foi possível utilizá-los, pois ainda estão em fase de desenvolvimento. A utilização de um SGWfC na grade ainda é considerado muito complexo, devido aos recursos heterogêneos e dispersos. Esta utilização ainda é considerada uma questão em aberto que está sendo pesquisada pelas comunidades científicas que desenvolvem os sistemas de gerência de *workflows* científicos.

5.3 Execução distribuída na Arquitetura Proposta

Para a realização dos experimentos utilizando a arquitetura proposta no capítulo 3, foi desenvolvido um portal *web* com o objetivo de oferecer aos pesquisadores uma interface gráfica que permitisse a definição e execução de *workflows* para realização de experimentos de simulações de dinâmica molecular em ambientes distribuídos e, também disponibilizar um ambiente para consultas aos dados de proveniência armazenados referentes à definição e execução do experimento no ambiente distribuído.

A construção do portal se fez necessária para auxiliar os pesquisadores de bioinformática na automação de seus experimentos de dinâmica molecular em ambientes distribuídos. O portal está representado pela camada de interface na arquitetura. É através da camada de interface que é feita a integração das camadas de aplicação, de serviços de proveniência e de persistência. Pelo portal, é possível definir os parâmetros e arquivos necessários para realizar os experimentos de simulações de dinâmica molecular.

O *workflow* é executado pela máquina de execução do *workflow*. O processo de execução do *workflow* é feito de forma transparente ao pesquisador, o que permite que mesmo os pesquisadores que possuem poucos conhecimentos nos programas necessários para realizar seus experimentos, em como executar o *workflow* e sobre o funcionamento de um cluster de PC ou uma grade computacional possam utilizar. O portal também disponibiliza aos pesquisadores interfaces para consultas aos dados de proveniência referentes a definição e execução dos experimentos no ambiente distribuído. O principal objetivo do portal é oferecer uma camada intermediária entre o usuário, o SGWfC e o ambiente distribuído.

Também são objetivos do portal:

- Disponibilizar uma interface gráfica simples e intuitiva ao pesquisador de bioinformática para realizar seus experimentos de simulação de dinâmica molecular;
- Permitir a utilização do pacote de programas GROMACS para realização de experimentos de simulação de dinâmica molecular de forma simplificada;
- Permitir a definição e execução de *workflows* científicos em ambientes distribuídos;
- Atender tanto o usuário de nível mais básico como o usuário de nível mais avançado;
- Possuir versatilidade e simplicidade na agregação de recursos computacionais;

- Disponibilizar ao usuário um ambiente de consulta aos dados de proveniência armazenados.

Um grande diferencial na utilização de um portal *web* para execução de experimentos de bioinformática, como os de simulações de dinâmica molecular é a não necessidade de o usuário ter conhecimentos em como instalar e configurar todos os programas necessários para sua pesquisa em sua máquina local, e também de não ficar restrito a sua máquina local para executar seus experimentos. Além disso, o portal disponibiliza os programas de forma centralizada podendo ser acessados de qualquer lugar através de um navegador de páginas *web*, e de qualquer sistema operacional que o usuário estiver acostumado a utilizar.

No caso das simulações de dinâmica molecular, os experimentos levam dias, semanas ou meses para serem executados, daí a necessidade da utilização de um ambiente distribuído, o que faz com o que o uso de um portal *web* facilite os pesquisadores a realizarem seus experimentos sem a necessidade de ficar esperando a execução de cada etapa em sua máquina local e possibilitando a sua execução em um ambiente distribuído.

Visando a não utilização de linguagens ou ferramentas de origem proprietária, foi utilizada para a construção do portal a linguagem PHP (PHP, 2008) e como o servidor *web* para o portal foi o Apache (APACHE, 2008). O SGDB (Sistema Gerenciador de Banco de Dados) escolhido para armazenar os dados de proveniência dos experimentos foi o MySQL (MYSQL, 2008). Todos os programas escolhidos atenderam as necessidades do portal. O portal desenvolvido nesta dissertação foi chamado de **GromDExp**.

5.3.1 GromDExp

O portal GromDExp oferece um ambiente amigável para auxiliar os pesquisadores de bioinformática a realizar seus experimentos de simulação de dinâmica molecular em ambientes distribuídos e a consultar os dados de proveniência armazenados. O Portal GromDExp é composto por um conjunto de páginas HTML (HTML, 2008) e PHP (PHP, 2008) que permite sua utilização através de uma interface simples e intuitiva. A Figura 5.7 apresenta a página principal do GromDExp. O portal

GromDExp pode ser acessado através do endereço *web* http://146.164.20.135/pmbarros/GromDExp.

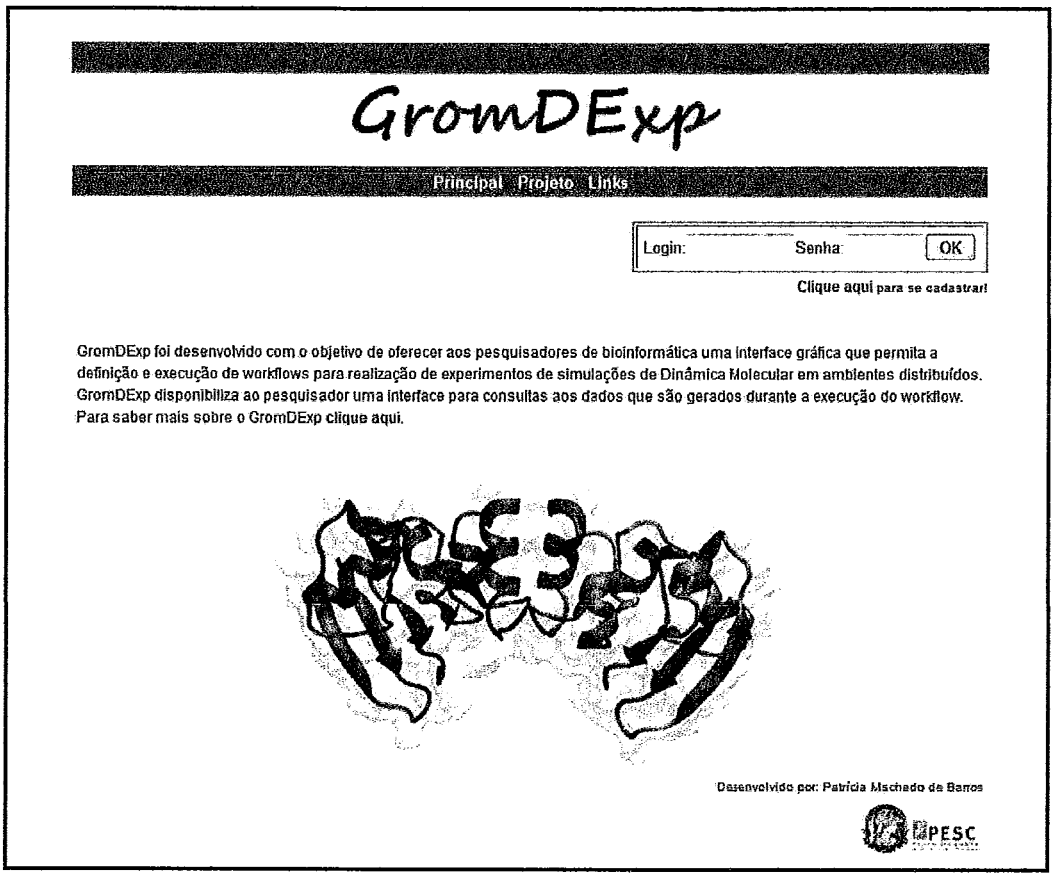

Figura 5.7 – Página principal do GromDExp.

Para um pesquisador (usuário) ter acesso ao portal, é necessário que ele faça um cadastro informando seus dados pessoais, assim como seu perfil. O perfil do usuário consiste de três níveis, básico para pesquisadores que possuem pouca experiência em realizar experimentos de dinâmica molecular; avançado para pesquisadores que necessitam realizar simulações mais complexas e, administrador, que permite a criação de novos workflows. O cadastro do usuário é enviado ao administrador do portal, onde é feita a validação de seus dados. Após a validação dos dados, o usuário recebe uma notificação informando que seu acesso foi liberado para utilizar os serviços oferecidos pelo portal. As informações dos dados de cada usuário cadastrado são armazenadas na classe wf_user, conforme modelo de dados apresentado no capítulo 3. É criado um diretório para cada usuário, onde são armazenados os arquivos referentes a cada experimento.

Para acessar os serviços oferecidos pelo portal, o usuário deve se autenticar com o nome do usuário e senha cadastrados. Após se autenticar, a página é direcionada para a página de definição do *workflow* de acordo com o perfil do usuário. Caso o usuário esteja cadastrado no nível básico, ele tem um único *workflow* disponível, o GromDFlow. Caso contrário, tem duas opções de *workflows*, o GromDFlow e o GromDFLow Avançado (ambos apresentados no capítulo 4 desta dissertação).

Na Figura 5.8 é possível verificar os *workflows* disponíveis para um usuário com o perfil avançado.

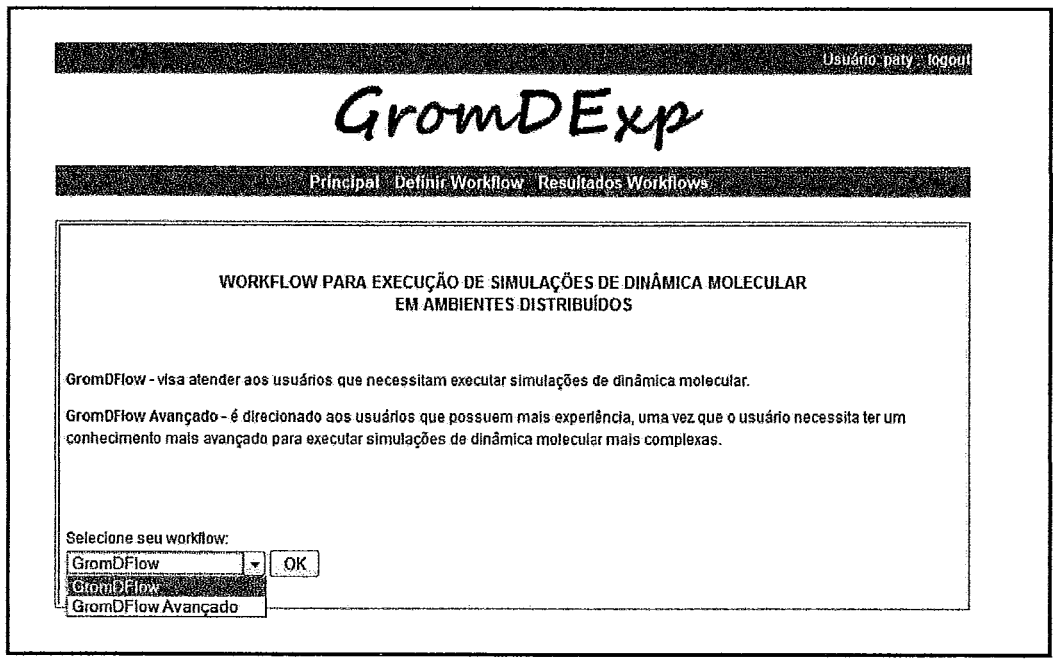

Figura 5.8 – Página com os workflows disponíveis.

Após o usuário selecionar qual *workflow* pretende executar, a página é direcionada para a página de definição dos parâmetros e arquivos de entrada que são utilizados no *workflow*. A página consiste na transferência dos arquivos de entrada, como o arquivo PDB e os arquivos de parâmetros de configuração do GROMACS e também do preenchimento dos parâmetros que são utilizados para a execução do *workflow*. A Figura 5.9 apresenta a página de definição do *workflow* GromDFlow. Vale ressaltar, que os *workflows* GromDFlow e GromDFlow Avançado podem ser executados com qualquer arquivo PDB.

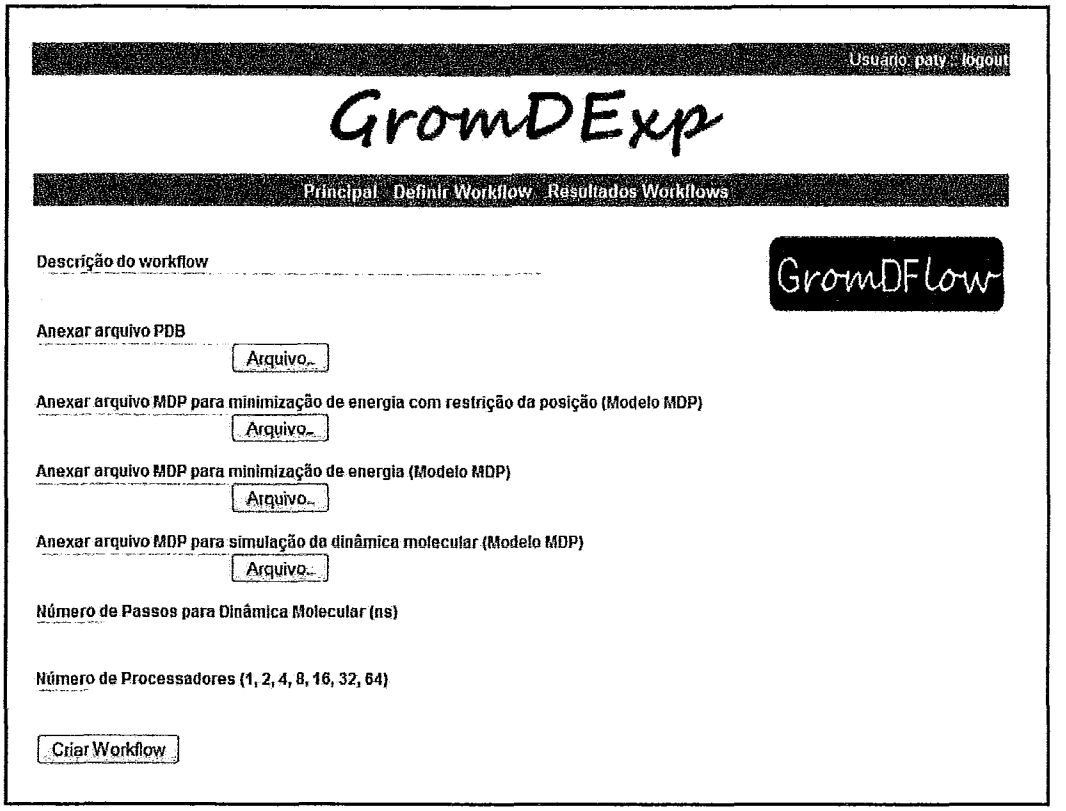

Figura 5.9 – Página de definição dos parâmetros do GromDFlow.

Uma das funções disponíveis no portal é identificar os parâmetros e arquivos de entrada definidos pelo usuário e traduzi-los para o documento XML baseado na descrição do *workflow*. Após esta etapa, o *workflow* está pronto para ser executado pelo SGWfC. As informações sobre os parâmetros e arquivos de entrada são armazenadas na classe *wf_create* do modelo de dados.

O *workflow* é executado de forma transparente ao pesquisador, ele não precisa ter conhecimentos da ferramenta de *workflow* que está sendo utilizada e nem de como e onde será executado. O *workflow* inicia parte da execução no ambiente centralizado utilizado pela camada local do *workflow* GromDFlow (*subworkflow* GROMACSLocal). Após o término dos programas que fazem parte do *subworklfow* GROMACSLocal, inicia-se a execução do *subworkflow* GROMACSRemoto no ambiente distribuído.

Os dados gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído são armazenados no banco de dados. Logo, que uma tarefa é submetida, um registro é feito na classe *wf_execution*. Os dados que são gerados ao longo da execução são armazenados na classe *wf_data*, como o número da tarefa que foi

submetida ao ambiente distribuído, o nome da tarefa, a quantidade de processadores utilizados, a data e hora do início e término da execução. Na classe *wf_data_cluster* são armazenadas as informações referentes ao ambiente distribuído, onde a tarefa está sendo executada, por exemplo, em qual máquina, em qual fila. As informações dos arquivos de saída gerados são armazenados na classe *wf_output*.

5.3.2 Ambiente Computacional

O ambiente computacional utilizado para realizar os experimentos consiste de um servidor *web* onde estão hospedadas as páginas HTML e PHP do GromDExp e parte dos módulos desenvolvidos na arquitetura. O ambiente centralizado, ou seja, a máquina local onde é executado o *subworkflow* GROMACSLocal e o cluster de PC onde é executado o *subworkflow* GROMACSRemoto foram os mesmos já definidos na seção 5.1 deste capítulo. Os experimentos foram realizados utilizando a versão 3.3.2 do pacote de programas GROMACS. A versão do Kepler utilizada para modelar o *workflow* GromDFlow foi a 1.0.0.

5.3.3 Estudo de Caso

Para a realização dos experimentos através do portal GromDExp foi utilizado o arquivo PDB de uma proteína chamada Shethna (URETA e NORDLUND, 2002). O objetivo deste experimento é estudar os aspectos apresentados pela proteína. A análise faz parte do projeto "Estudos do mecanismo de proteção conformacional através de técnicas de modelagem computacional". Este projeto está em desenvolvimento no Laboratório de Física Biológica do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro.

A *Gluconacetobacter diazotrophicus* é uma bactéria encontrada em associação com a cana de açúcar, o café, o abacaxi e a batata-doce, dentre outros, sendo capaz de ajudar no processo de crescimento e desenvolvimento do vegetal através de mecanismos de fixação de nitrogênio atmosférico (ROJAS e MELLADO, 2003). Para que esta fixação do nitrogênio ocorra, a bactéria necessita de uma enzima chamada nitrogenase. Esta enzima pode ser inativada pela presença de oxigênio. Para prevenir sua inativação, uma outra proteína, chamada Shethna, pode associar-se à nitrogenase. Para melhor entender este mecanismo de interação, é necessário estudar os aspectos apresentados

pela proteína Shethna, como a possibilidade dela ser um dímero (duas proteínas idênticas unidas) ou um monômero (uma única proteína). Uma das etapas deste estudo é feito através da técnica de simulação de dinâmica molecular.

As simulações de dinâmica molecular realizadas com a proteína Shethna calcularam o equivalente a 10 nanosegundos de iterações moleculares em uma temperatura de 300K utilizando o campo de força GROMOS (VAN GUNSTEREN e BERENDSEN, 1987). Todos os cálculos foram feitos utilizando a precisão simples do GROMACS. Para minimização de energia foi utilizado o método do gradiente conjugado (descrito no capítulo 2).

A Figura 5.10 apresenta parte do *workflow* GromDFlow para a execução da simulação da dinâmica molecular com a proteína Shethna.

5.3.4 Resultados

Os experimentos foram executados através do portal GromDExp, utilizando o *workflow* GromDFlow. A execução da camada GROMACSLocal do GromDFlow foi executada no ambiente centralizado, pois sua execução apresenta um rápido processamento. A segunda camada, a GROMACSRemoto, foi executada no ambiente de cluster de PC. As tarefas foram enviadas à fila do cluster através do *workflow* GromDFlow. As tarefas foram executadas em paralelo utilizando 1, 2, 4 e 8 processadores por nó. Cada nó é responsável pela execução de uma tarefa de um experimento de simulação de dinâmica molecular da proteína Shethna. A comunicação entre os processadores é feita através do uso do MPI (MPICH2, 2008).

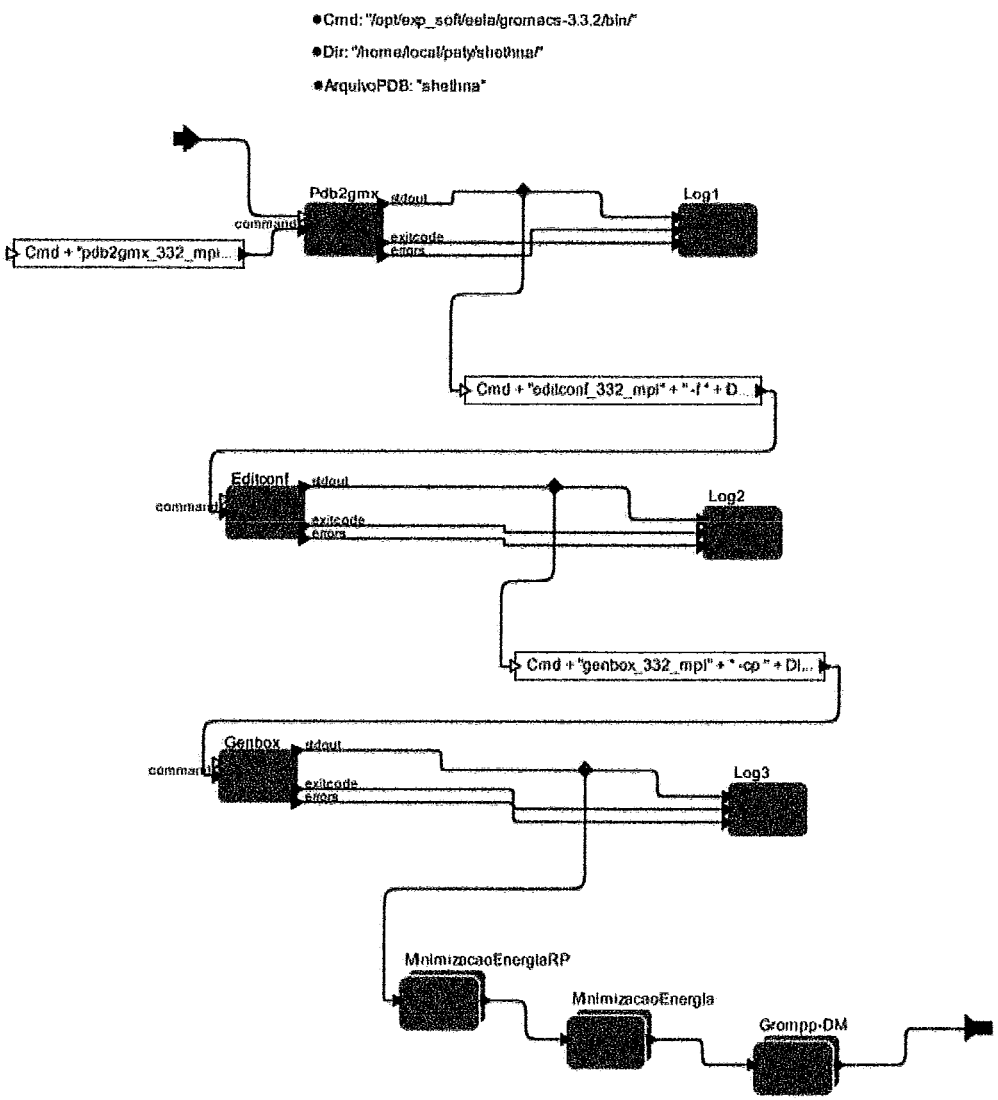

Figura 5.10 - Parte do *workflow* GromDFlow com a proteína Shethna.

A Tabela 5.2 apresenta a média dos tempos de execução do GromDFlow na camada local – GROMACSLocal (ambiente centralizado) e na camada remota - GROMACSRemoto (ambiente distribuído). Os tempos são apresentados em minutos.

Tabela 5.2 - Tempo de execução em minutos por processador.

Número de processadores /nó	Tempo execução camada local (ambiente centralizado)	Tempo execução camada remota (ambiente distribuído)	Tempo total de execução
1	12,0	14820,0	14832,0
2	12,0	8940,0	8952,0
4	12,0	4755,0	4767,0
8	12,0	3116,0	3128,0

Os tempos de execução do experimento realizado com a proteína Shethna apresentados na Tabela 5.2 são referentes ao equivalente a 10 nanosegundos de cálculos (número de passos). Através do gráfico apresentado Figura 5.11 é possível observar uma redução significativa no tempo de execução do experimento conforme o número de processadores aumenta. O gráfico apresenta os tempos normalizados das execuções da proteína Shethna utilizando 1, 2, 4 e 8 processadores.

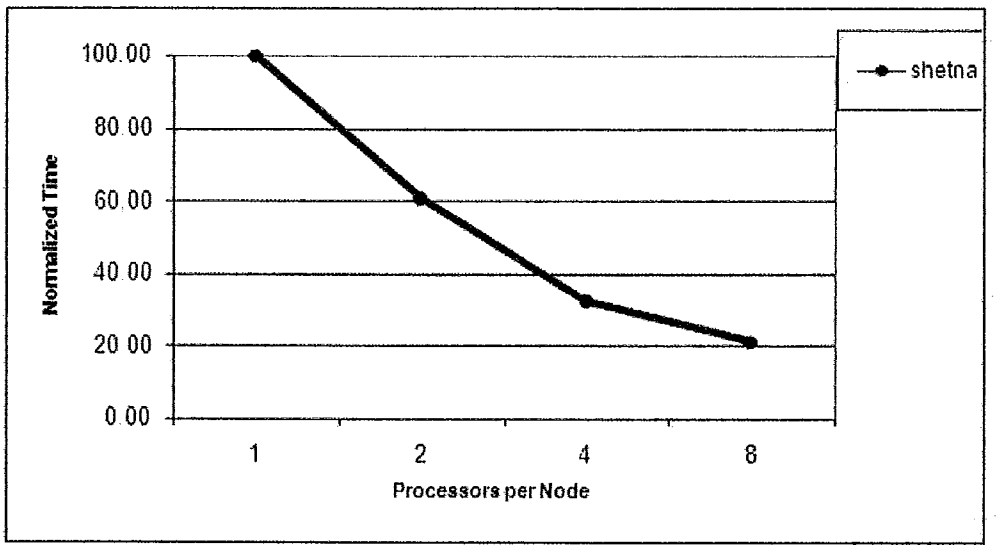

Figura 5.11 – Tempo normalizado das execuções dos experimentos com a proteína Shethna.

Vale ressaltar que o portal GromDExp está disponível somente para execução de experimentos utilizando um cluster de PC. Para execução em uma grade computacional é necessário realizar algumas alterações.

5.3.5 Módulo de Consulta do GromDExp

O módulo de consulta disponibilizado pelo portal GromDExp permite aos pesquisadores ter acesso aos dados da proveniência retrospectiva referentes a cada experimento. Os dados são armazenados de forma centralizada, ou seja, em um único repositório de dados.

Os dados gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído são capturados, tratados e armazenados pelos módulos desenvolvidos na camada de serviços de proveniência – são eles: servidor de proveniência, cliente de proveniência e repositório de proveniência. O módulo cliente de proveniência captura os dados que são gerados no ambiente distribuído e envia para o servidor de proveniência, que recebe os dados, trata e armazena no repositório de proveniência com base no modelo de dados definido no capítulo 3. A captura da proveniência dos dados gerados durante a execução do *workflow* no ambiente distribuído é importante, tendo em vista, que o pesquisador precisa ter conhecimento sobre o que está acontecendo durante a execução do seu experimento, como por exemplo, o andamento da execução.

O módulo de consulta disponibilizado pelo portal GromDExp se fez necessário, pois através dele, o pesquisador tem um acesso centralizado a todas as informações e dados de seus experimentos, como os arquivos de entrada que foram utilizados e os arquivos de saída que foram gerados no ambiente distribuído de um determinado experimento. Sem este módulo, o pesquisador teria seus resultados espalhados, pois a execução é realizada em ambientes computacionais distintos, dificultando o acesso às informações.

Por não haver ainda na literatura, um formato padrão de consultas para os experimentos de simulação de dinâmica molecular, as consultas disponíveis através do portal GromDExp foram pré-definidas baseada no modelo de dados definido no capítulo 3 e de acordo com as necessidades dos pesquisadores do Laboratório de Física Biológica do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro. As consultas aos dados estão disponíveis aos pesquisadores através de uma interface gráfica do GromDExp. Também é possível, através do GromDExp fazer o *download* dos arquivos com os resultados dos experimentos.

O GromDExp permite que o pesquisador veja todos os *workflows* que foram executados por ele. Os *workflows* podem ser visualizados em uma lista, que contém as seguintes informações: nome do *workflow*, descrição do *workflow*, data de criação do *workflow*, e os dados do *workflow*, que consiste nos dados de entrada e de execução. O nome do *workflow* consiste no nome do *workflow* selecionado, o nome do usuário que o executou, e da data e hora da criação do *workflow*. A Figura 5.12 mostra a página do portal GromDExp com os resultados da consulta dos *workflows* executados pelo usuário "paty". Através desta interface, o pesquisador tem acesso a todos os *workflows* que foram executados por ele e os dados correspondentes a cada um. Os arquivos de entrada e de saída de cada experimento são armazenados no servidor dentro do diretório de cada usuário.

O portal também permite que o pesquisador consulte quais os parâmetros e arquivos de entrada foram utilizados para cada experimento realizado, assim como a visualização das informações e dos dados gerados durante a execução das tarefas remotas do *workflow* no ambiente distribuído. A Figura 5.13 apresenta a página com o resultado da consulta dos dados de execução do *workflow* "gromdflow_paty_05121008_221042". Através desta consulta, o pesquisador tem acesso às informações sobre o início e término da execução do *workflow* como um todo, início e término da execução da tarefa remota do *workflow* no ambiente distribuído, o andamento (*status*) da execução da tarefa remota. E, também, acesso a consulta dos arquivos de saída gerados e aos dados sobre o ambiente remoto onde foi executado.

A Figura 5.14 mostra a página com o resultado da consulta dos dados do ambiente remoto referente à execução da tarefa remota do *workflow* selecionado no ambiente distribuído. A consulta mostra o nome do *workflow*, o número da tarefa que foi executada (este número é criado automaticamente pelo sistema de filas do cluster), o número de processadores que foi utilizado para executar a tarefa, em que nó do cluster de PC a tarefa foi executada e em qual fila do cluster de PC.

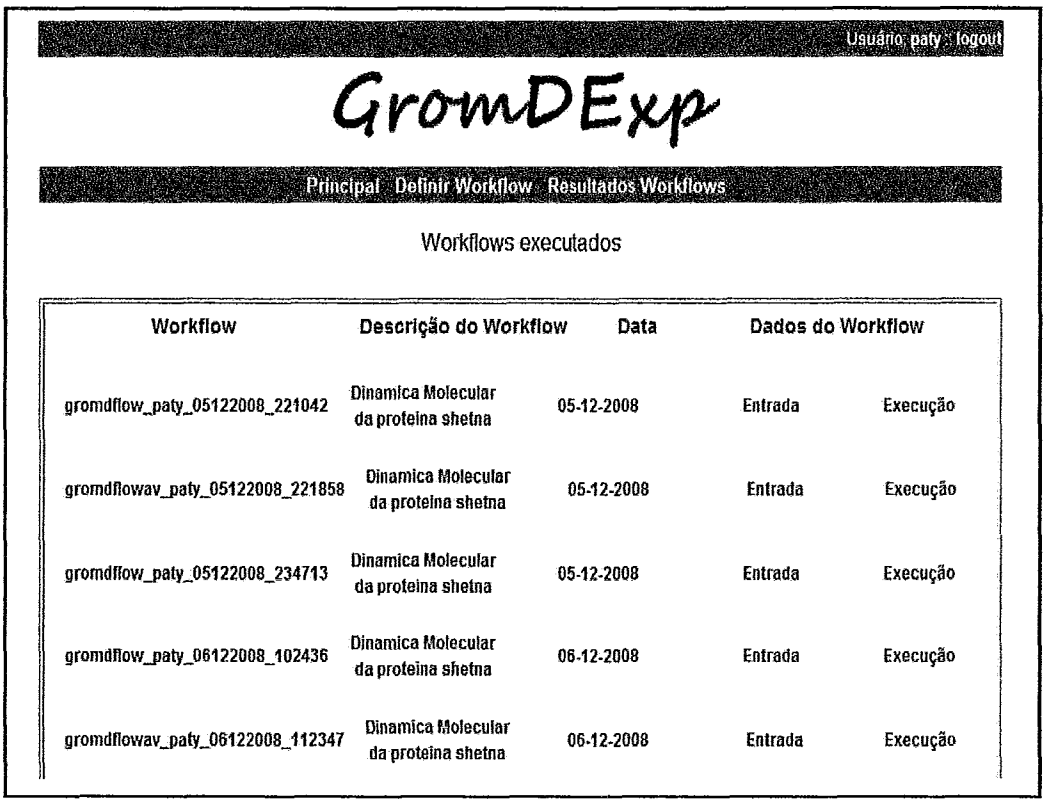

Figura 5.12 – Página do portal GromDExp com os resultados dos *workflows* executados.

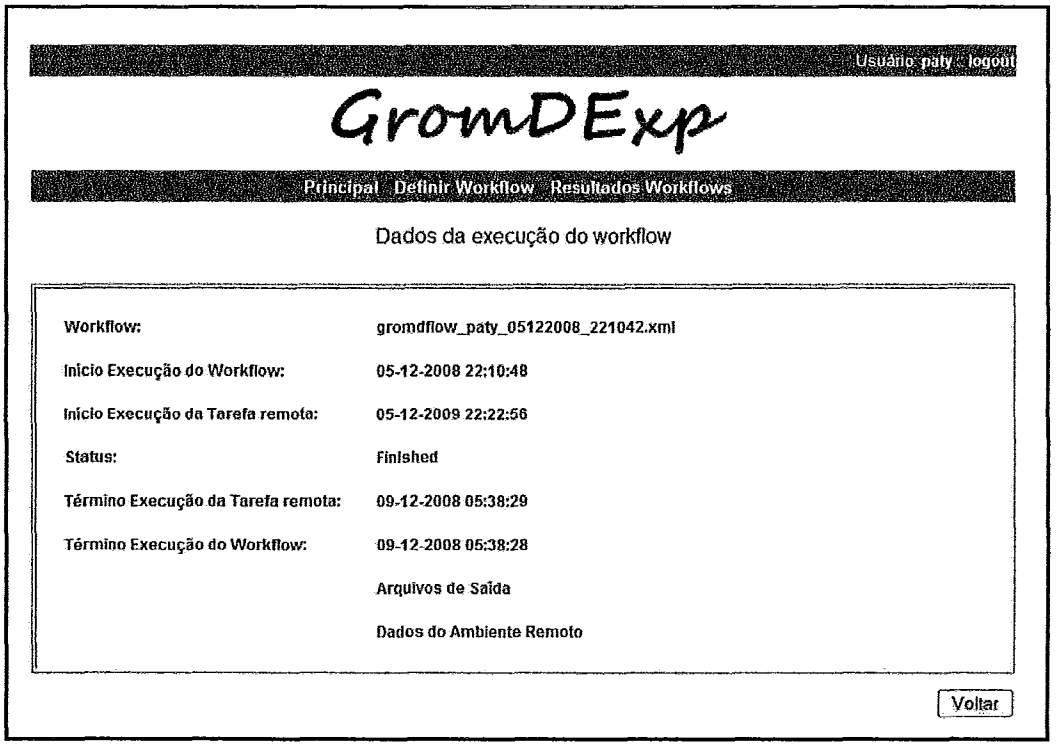

Figura 5.13 – Página do GromDExp com os dados referentes a execução do workflow selecionado.

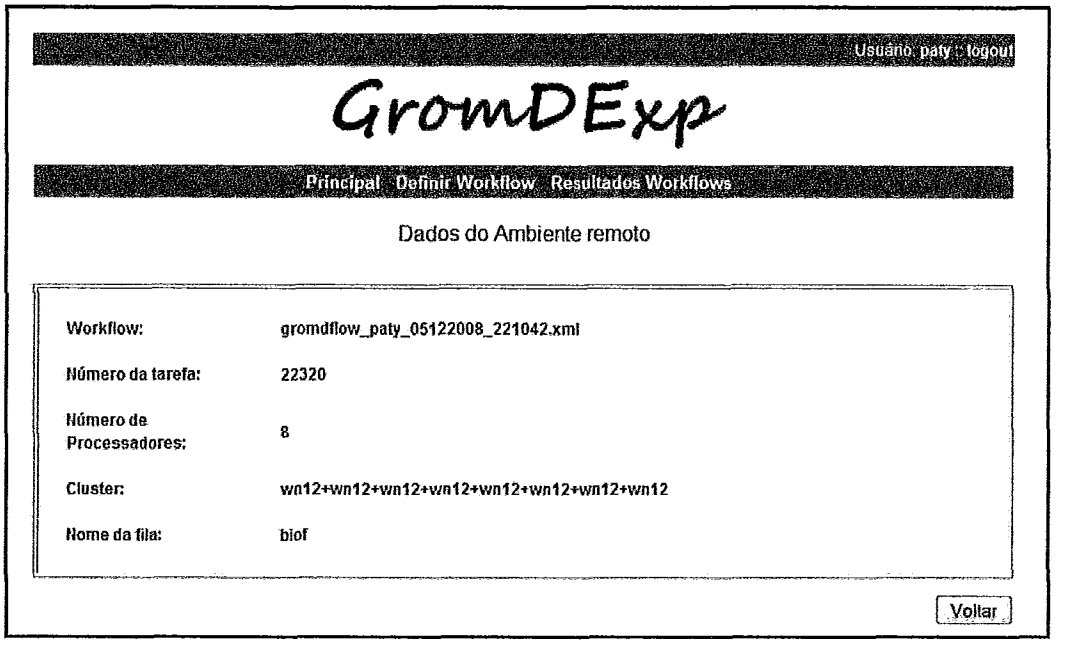

Figura 5.14 – Página do GromDExp com os dados do ambiente remoto.

As consultas foram pré-definidas de acordo com o modelo de dados definido no capítulo 3 desta dissertação. A disponibilização de uma interface para consultas é muito valiosa para o pesquisador, pois é através dela que ele tem acesso a todas as informações e dados referentes a cada experimento, além de poderem ser acessadas de qualquer lugar, pois estão disponíveis através do portal GromDExp. Por exemplo, sem a disponibilização da consulta aos dados do ambiente remoto, o pesquisador teria dificuldade em verificar em qual nó e fila do cluster de PC as tarefas foram executadas.

Capítulo 6

6 Conclusões

Os pesquisadores de bioinformática estão cada vez mais utilizando recursos computacionais de alto desempenho para realização de seus experimentos *in silico*. A simulação de dinâmica molecular é um dos métodos da bioinformática que consiste em estudar a evolução temporal de um dado sistema molecular. Os experimentos de dinâmica molecular são compostos pela execução de uma série de tarefas encadeadas, assim o uso de um *workflow* auxilia naturalmente os pesquisadores na modelagem de seus experimentos. A utilização de sistemas de gerência de *workflows* científicos para executar experimentos tem sido gradativamente adotada pela comunidade científica de bioinformática.

Parte do *workflow* de dinâmica molecular requer um ambiente com capacidade de PAD para ser executado, tendo em vista que algumas de suas etapas consomem muito recurso computacional, necessitando de um ambiente distribuído (cluster de PC ou grade computacional). A execução de *workflows* científicos em uma grade computacional ainda é um problema em aberto, apesar dos sistemas de *workflows* voltados exclusivamente para esses ambientes. O uso e a configuração de um ambiente de grade computacional ainda são muito complexos. Um dos complicadores é a captura dos dados gerados durante a execução do experimento no ambiente tanto centralizado quanto distribuído e manter estes dados armazenados em repositórios torna-se uma tarefa complexa.

Os sistemas de gerência de *workflows* científicos (SGWfC) permitem que a execução de um *workflow* seja automatizada ao modelar tarefas encadeadas. Os SGWfC destacados por popularidade na execução de *workflows* e reconhecimento científico foram o Kepler, o VisTrails, o Taverna, o GriDFlow e o GriddLes. Cada SGWfC apresenta abordagens distintas para controlar a execução do *workflow* e registrar os dados de proveniência. Nenhum dos SGWfC analisados apresentou uma solução adequada para tratar o problema do monitoramento da execução do *workflow* ora no ambiente centralizado, ora no ambiente distribuído e na captura dos dados de

proveniência independente do ambiente, além de não possuírem mecanismos para consultas aos dados armazenados.

Considerando a necessidade de oferecer uma solução que auxilie os pesquisadores de bioinformática na automação e execução de seus experimentos *in silico*, esta dissertação propôs uma arquitetura, complementar aos SGWfC sequenciais, que permite a gerência da execução de *workflows* científicos em ambientes distribuídos, bem como apoio à captura da proveniência dos dados referentes a execução remota do *workflow*. A solução apresentada visa atender aos pesquisadores de bioinformática na realização de seus experimentos de simulações de dinâmica molecular.

A arquitetura está dividida em quatro camadas, que são a camada de interface, que disponibiliza uma interface gráfica que tem como objetivo fazer a interação direta com o pesquisador, oferecendo interfaces para definição, execução e monitoramento do *workflow*, além de possuir um ambiente para consultas aos dados de proveniência armazenados; a camada de aplicação, que é responsável pela execução do *workflow*; a camada de serviços de proveniência, que tem como função monitorar a execução das tarefas remotas do *workflow* no ambiente distribuído, capturar e armazenar os dados de proveniência; e a camada de persistência, que é onde estão armazenados os dados de proveniência.

Para oferecer estas funcionalidades à arquitetura, foram desenvolvidos os módulos descritos na camada de serviços de proveniência, que são o servidor de proveniência, cliente de proveniência e repositório de proveniência. Os módulos fazem parte da arquitetura Matrioshka, onde provêem a integração entre o SGWfC sequencial e o ambiente distribuído e são capazes de gerenciar a execução do *workflow* no ambiente distribuído, capturar e registrar os dados de proveniência gerados pelo experimento.

Para demonstrar a viabilidade da arquitetura proposta e os módulos desenvolvidos, foi definido um *workflow*, denominado GromDFLow para execução de experimentos de simulações de dinâmica molecular em ambientes distribuídos. GromDFlow é dividido em duas camadas, a local, que realiza a execução local dos programas em um ambiente centralizado e a remota que é executada em um ambiente distribuído. Para modelagem do *workflow* GromDFlow, foi utilizado o SGWfC Kepler.

Também foi modelado uma variação do *workflow* GromDFlow, denominado GromDFlow Avançado, que visa atender aos pesquisadores que necessitam executar experimentos mais complexos. Junto aos componentes do Kepler também foi desenvolvido um ator, chamado JobExecutor, que é o responsável por enviar as tarefas remotas do *workflow* ao ambiente distribuído e realizar a comunicação entre os módulos desenvolvidos. O Kepler se mostrou eficiente na definição e execução dos *workflows*, bem como na inclusão de novos componentes.

O portal GromDExp pertencente a camada de interface da arquitetura proposta, foi desenvolvido baseado no uso de tecnologias de distribuição livre. A implementação do GromDExp contribuiu para facilitar os pesquisadores a definir e executar seus *workflows* referentes aos experimentos de simulações de dinâmica molecular de forma automatizada, bem como monitorar a execução do *workflow* no ambiente distribuído, capturar e armazenar os dados de proveniência. O portal GromDExp também disponibiliza um ambiente para consultas aos dados de proveniência armazenados.

Na Tabela 6.1 podemos comparar as características atendidas pelo GromDExp em relação as características apresentadas pelos SGWfC analisados na Tabela 2.1 apresentada no capítulo 2 desta dissertação.

Tabela 6.1 – Comparação entre os SGWfC analisados e o GromDExp

	KEPLER	**VISTRAILS**	**TAVERNA**	**GRIDFLOW**	**GRIDDLES**	**GROMDEXP**
Domínio da Aplicação	Científico em geral	Científico em geral	Bioinformática	Científico em geral	Científico em geral	Dinâmica Molecular
Execução remota de uma tarefa	Sim	Não	Não	Sim	Sim	Sim
Execução paralela de uma tarefa remota	Não	Não	Não	Sim	Sim	Sim
Utilização de serviços de grade computacional	Não	Não	Não	Sim	Sim	Sim
Armazenamento de dados	Sim	Sim	Sim	Não	Não	Sim
Proveniência de dados	Não	Sim	Sim	Não	Não	Sim

Proveniência de dados distribuída	Não	Não	Não	Não	Não	Sim
Interface para consultas	Não	Não	Não	Não	Não	Sim

Os experimentos foram divididos em três estudos de casos distintos. No primeiro, o GromDFlow foi executado utilizando um conjunto de proteínas no ambiente de cluster de PC. No segundo, foi realizado um estudo com o objetivo de conhecer quais etapas são necessárias para executar um experimento no ambiente de grade computacional. No terceiro, e último, os experimentos foram submetidos a partir do GromDExp, onde parte do experimento foi executado em um cluster. A proposta foi validada utilizando um experimento real de simulação de dinâmica molecular com o GromDFlow. O estudo realizado permitiu demonstrar a viabilidade da proposta em um *workflow* real.

Com os experimentos realizados, podemos concluir que a arquitetura proposta atendeu os objetivos desta dissertação, permitindo que os pesquisadores de bioinformática realizassem seus experimentos de dinâmica molecular de forma automatizada quanto à definição e execução de workflows, permitindo a execução de tarefas remotas do *workflow* no ambiente distribuído, monitoramento da execução remota no ambiente distribuído, captura e armazenamento dos dados de proveniência.

Como trabalhos futuros, podemos apontar algumas possibilidades de evolução desta dissertação, como estender os módulos servidor de proveniência, cliente de proveniência e repositório de proveniência, desenvolvidos na arquitetura para o ambiente de grade computacional e realizar os experimentos a partir do GromDExp no ambiente de grade computacional do projeto EELA.

A solução apresentada mantém o foco na captura dos dados de proveniência da definição do experimento e da execução das tarefas remotas do *workflow* no ambiente distribuído, um possível trabalho futuro seria fornecer também a captura dos dados intermediários que são gerados durante a execução das tarefas do *workflow* no ambiente centralizado.

Outro trabalho futuro que pode ser destacado seria a definição de novos *workflows* de dinâmica molecular, incluindo também outros programas do pacote GROMACS, para atender as necessidades de um número maior de pesquisadores. Além de uma melhoria na interface gráfica do portal GromDExp, permitindo que novos parâmetros possam ser adicionados para a definição dos experimentos.

Além dos trabalhos futuros mencionados acima, também pode ser apontada a expansão do GromDExp para suportar outras linguagens de definição de *workflows* e também a integração com outros SGWfC, além do Kepler.

Referências Bibliográficas

ALBERTS, B., JOHNSON, A., WALTER, P., *et al.*, 2004, "*Biologia Molecular da Célula*", 4 ed., São Paulo, SP, Brasil, Artmed.

ABRAMSON, D., KOMMINEMI, J., ALTINTAS, I., 2005, "Flexible IO Services in the Kepler Grid *Workflow* System". In: *Proceedings of the First International Conference on e-Science and Grid Computing*, pp.255-262, Washington, DC, USA.

ALTINTAS, I., BERKLEY, C, JAEGER, E., *et al.*, 2004, "Kepler: An Extensible System for Design and Execution of Scientific *Workflows*". In: *Proceeding of 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04)*, pp. 21-23, Santorini Island, Greece.

APACHE, 2008, "The Apache Software Foundation". Disponível em: <http://www.apache.org>. Acesso em: 18 out. 2008.

AYOAMA, Y., NAKANO, Y., 1999, "Practical MPI Programming". *International Technical Support Organization*, IBM.

BARROS, P.M., CRUZ, S. M. S, BITAR, M., *et al.*, 2008, "GromDFlow: Um Workflow para Dinâmica Molecular em Ambientes Distribuídos". *Brazilian Symposium on Bioinformatics*, Santo André, SP, Brasil.

BAXEVANIS, A. D., OUELLETTE, B. F. F., 2005, "*Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*", 3 ed. New York, John Wiley & Sons.

BERMAN, H. M., WESTBROOK, J., FENG, Z., *et al.*, 2000, "The Protein Data Bank", *Nucleic Acids Research*, v. 28, n. 1 pp. 235-242.

BROOKS, B. R., BRUCCOLERI R. E., OLAFSON, B. D., *et al.*, 1983, "CHARMM – A Program for Macromolecular Energy Minimization and Dynamics Calculations", *Journal of Computational Chemistry*, v. 4, n. 2, pp. 187-217.

BUNEMAN, P., KHANA, S., TAN, W. C., 2001, "Why and Where: A Characterization of Data Provenance". In: *Proceedings Intl. Conf. on Data Theory*, v. 1973 of LNCS, pp. 316-330, Springer.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., *et al.*, 2006a, "VisTrails: Visualization meets Data Management". In: *Proceedings ACM SIGMOD*, Chicago, Illinois.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., *et al.*, 2006b "*Using Provenance to Streamline Data Exploration through Visualization*", SCI Institute Technical Report, n. UUSCI-2006-016, University of Utah.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., *et al.*, 2006c, "Managing the Evolution of Dataflows with VisTrails". In: *IEEE Workshop on Workflow and Data Flow for Scientific Applications*, pp. 71-71, Atlanta, GA, USA.

CAMPBELL, M. K., FARREL, S. O., 2006, "*Bioquímica*", 1 ed., São Paulo, SP, Brasil, Thomson.

CAO, J., JARVIS, S., SAINI, S., *et al.*, 2003, "GridFlow: Workflow Management for Grid Computing". In: *Proceedings of the 3rd International Symposium on Cluster Computing and Grid*", pp. 198-206, Tokyo, Japan.

CRUZ, S. M. S., BARROS, P. M., BISCH, P. M., *et al.*, 2008, "Provenance Services for Distributed Workflows". In: *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 526-533, Washington, DC, USA.

DALAL, S., BALASUBRAMANIAN, S., REGAN, L., 1997, "Protein Alchemy: Changing β-sheet into α-helix", *Nature Structural Biology*, v. 4, n. 7.

EELA, 2009, "E-Infrastructure shared between Europe and Latin America". Disponível em: <http://www.eu-eela.eu>. Acesso em: 06 jan. 2009.

FOSTER, I. KESSELMAN C., TUECKE S., 2001, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *Lecture Notes in Computer Science*, v. 2150, pp. 1-4.

FOSTER, I., KESSELMAN, C., 1999, "*The Grid: Blueprint for a Future Computing Infrastructure*", Morgan Kaufmann, USA.

FREEFLUO, 2008, "FreeFluo Workflow Enactor". Disponível em: <http://freefluo.sourceforge.net>. Acesso em: 02 dez. 2008.

FREIRE, J., KOOP, D., SANTOS, E., *et al.*, 2008, "Provenance for Computational Tasks: A Survey", In: Computing in Science & Engineering, v. 10 (3), pp. 11-21.

GIBAS, C., JAMBECK, P., 2001, "*Developing Bioinformatics Computer Skills*", Associates, Sebastopol, O'Reilly.

GLITE, 2009, "Lightweight Middleware for Grid Computing". Disponível em: <http://glite.web.cern.ch/glite>. Acesso em: 06 jan. 2009.

GOBLE, C., WROE, C., STEVENS R., 2003, "The myGrid Project: Services, Architecture and Demonstrator". In: *Proceedings UK e-Science All Hands Meeting 2003*, pp. 595-603.

GOODMAN, N., 2002, "Biological Data Becomes Computer Literature: New Advances in Bioinformatics", *Current Opinion in Biotechnology*, v. 13, n. 1, pp. 68-71.

GROMACS, 2008, "GROMACS: Fast, Free and Flexible MD". Disponível em: <http://www.gromacs.org>. Acesso em: 18 jan. 2008.

HTML, 2008, "HyperText Markup Language". Disponível em: <http://www.w3c.org/TR/html401>. Acesso em: 02 dez. 2008.

KEPLER, 2008. Disponível em: <https://code.kepler-project.org/code/kepler-docs/trunk/outreach/documentation/shipping/getting-started-guide.pdf>. Acesso em: 18 out. 2008.

KPF, 2008, "Kepler Provenance Framework". Disponível em: <http://kepler-project.org/Wiki.jsp?page=KeplerProvenanceFramework>. Acesso em: 02 dez. 2008.

KUTZNER, C., VAN DER SPOEL, D., FECHNER M., *et al.*, 2007, "Speeding up Parallel GROMACS on High-Latency Networks", *Journal of Computational Chemistry*, v. 38-12, pp. 2075-2084.

LEMOS, M., 2004, "*Workflow para Bioinformática*", Tese de Dsc., PUC-Rio, Rio de Janeiro, RJ, Brasil.

LEHNINGER, A. L., NELSON, D. L., COX, M. M., 2006, "*Princípios de Bioquímica*", 4 ed., São Paulo, SP, Brasil, Sarvier.

LENGAUER, T., 2002, "*Bioinformatics – from genomes to drugs*", Wiley-VCH Verlag GmbH, Weinheim.

MENEZES, J. G. M., 2008, "*Gerência Distribuída de Dados em Workflows de Bioinformática*". Dissertação de Msc. Instituto Militar de Engenharia, Rio de Janeiro, RJ, Brasil.

MORGON, N. H., COUTINHO, K., 2007, "*Métodos de Química Teórica e Modelagem Molecular*", 1 ed., São Paulo, SP, Brasil, Livraria da Física.

MORO, M., 2006, "Workflow". Disponível em: <http://www.inf.ufrgs.br/~mirella/workflow/work.html>. Acesso em: 02 dez. 2006.

MPICH2, 2008. Disponível em: <http://www.mcs.anl.gov/research/projetcts/mpich2>. Acesso em: 29 dez. 2008.

MUNDIM, K. C., 2002, "Modelagem Molecular Aplicada a Sólidos e Biomoléculas". *IV Escola de Inverno CBPF*, Rio de Janeiro, RJ, Brasil.

MYGRID, 2008. Disponível em: <http://www.mygrid.org.uk/tools/taverna>. Acesso em: 18 out. 2008.

MYSQL, 2008. Disponível em: <http://www.mysql.com>. Acesso em: 18 out. 2008.

OINN, T., ADDIS, M., FERRIS, J., *et al.*, 2004, "Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows", *Bioinformatics, Oxford University Press*, v. 20 (17), pp. 3045-3054.

PASCUTTI, P. G., 2007, "Introdução à Modelagem e Dinâmica Molecular". Disponível em: <http://www.nacad.ufrj.br/~amit/DinamicaMolecular.pdf>. Acesso em: 15 mai. 2007.

PDB, 2007, "Protein Data Bank". Disponível em: <http://www.pdb.org/home/home.do>. Acesso em: 12 nov. 2007.

PERL, 2008, "Practical Extraction and Report Language". Disponível em: <http://www.perl.com>. Acesso em: 13 dez. 2008.

PHILLIPS, J. C., BRAUN, R., WANG, W., *et al.*, 2005, "Scalable molecular dynamics with NAMD", *Journal of Computational Chemistry*, v. 26, pp. 1781-1802.

PHP, 2008, "Hypertext Preprocessor". Disponível em: <http://www.php.net>. Acesso em: 18 out. 2008.

PROSCIDIMI, F., CERQUEIRA, G. C., BINNECK, E, *et al.*, 2003, "Bioinformática: Manual do Usuário", In: *Revista BIO Tecnologia*, v. 29.

PTOLEMY, 2008. Disponível em: <http://ptolemy.eecs.berkeley.edu/ptolemyII>. Acesso em: 18 out. 2008.

PYTHON, 2008. "Python Programming Language". Disponível em: <http://www.python.org>. Acesso em: 02 dez. 2008.

ROJAS, J. M., MELLADO, J. C., 2003, "Population Dynamics of Gluconacetobacter Diazotrophicus in Sugarcane Cultivars and Its Effect on Plant Growth". *Microbial Ecology*, v. 46, pp. 454-464.

SHELL, 2008, "Introdução ao Shell Script". Disponível em: <http://aurelio.net/shell/apostila-introducao-shell.pdf>. Acesso em: 12 dez. 2008.

SILVA, A. W. S., 2003, *Estudo por Modelagem e Dinâmica Molecular da protease de variantes do vírus da imunodeficiência humana tipo 1 resistentes a drogas antivirais*. Tese de Dsc., IBCCF/UFRJ, Rio de Janeiro, RJ, Brasil.

SILVA, M. L., 2007, "*Planejamento de inibidores seletivos da serina hidroximetiltransferase de Plasmodium falciparum por Modelagem Molecular*". Dissertação de Msc. Instituto Militar de Engenharia, Rio de Janeiro, RJ, Brasil.

STEVENS, R., ZHAO, J., GOBLE, C., 2007, "Using Provenance to Manage Knowledge of in Silico Experiments". *Briefings in Bioinformatics, Oxford Journals*, v. 8, pp. 183-194.

SUN, 2008, "Java Technology". Disponível em: <http://java.sun.com>, Acesso em: 02 dez. 2008.

TAYLOR, I., SHIELDS, M., WANG, I., et al., 2003, "Triana Applications within Grid Computing and Peer to Peer Environments". *Journal of Grid Computing*, v. 1 (2), pp. 199-217, Kluwer Academic Press.

TORQUE, 2009, "Torque Resource Manager". Disponível em: <http://www.clusterresources.com/pages/products/torque-resource-manager.php>. Acesso em: 06 jan. 2009.

URETA, A., NORDLUND, S., 2002, "Evidence for Conformational Protection on Nitrogenase against Oxygen in Gluconacetobacter Diazotrophicus by a Putative FeSII Protein". *Journal of Bacteriol*, v.184(20), pp. 5805-5809.

VAN DER AALST, W. M. P., HOFSTEDE, A. H. M., KIEPUSZEWSKI, B., *et al.*, 2003, "Workflow Patterns", *Distributed and Parallel Databases*, v. 14, pp. 5-51, Hingham, USA.

VAN DER SPOEL, D., LINDAHL, E., HESS, B., *et al.*. 2004, "Gromacs User Manual Version 3.2". Disponível em: <http://www.gromacs.org>. Acesso em: 18 jan. 2008

VAN GUNSTEREN, W. F., BILLETER, S. R., EISING. A. A., *et al.*, 1996, "Biomolecular Simulation: The GROMOS96 Manual and User Guide", *VdF: Hochschulverlag AG and der ETH Zurich and BIOMOS b.v.*, Zurich, Groningen.

VAN GUNSTEREM, W. F., BERENDSEN, H. J. C., 1990, "Computer Simulation of Molecular Dynamics: Methodology, Applications and Perspectives in Chemistry." *Angewandte Chemie International Edition in English*, v.29, pp. 992-1023.

VERLET, L., 1967, "Computer Experiments on Classical Fluids I. Thermodynamical Properties of Lennard-Jones Molecules". *Physical Review*, v. 159, pp. 98-103.

VISTRAILS, 2008. Disponível em: <http://www.vistrails.org/index.php/Main_Page>. Acesso em: 18 out. 2008.

W3C, 2008, "World Wide Web Consortium - Extensible Markup Language (XML)". Disponível em: <http://www.w3.org/XML>. Acesso em: 02 dez. 2008.

WFMC, 2008, "The Workflow Reference Model". Disponível em: <http://www.wfmc.org.> Acesso em: 02 dez. 2008.

WIECZOREK, M., PRODAN, R., FAHRINGER, T., 2005, "Scheduling of Scientific Workflows in the Askalon Grid Environment". In: *ACM SIGMOD*, pp. 56-62.

YU, J., BUYYA, R., 2005, "A Taxonomy of Scientific Workflows Systems for Grid Computing", In: *SIGMOD Record Archive*, v. 34 (3), pp. 44-49.

YU, H., ROSEN, M. K., SACCOMANO, N. A., *et al*, 1993, "Sequential Assigment and Structure Determination of Spider Toxin Omega AgaIIVB", *Biochemistry*, v. 32, pp. 13123.

ZHAO, Y., HATEGAN, M., CLIFFORD, B., *et al.*, 2007, "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In: *IEEE Congress on Services*, v.(9-13), pp. 199-206.