MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
SECRETARIA DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

JOSÉ GUILHERME MONTEIRO DE MENEZES

GERÊNCIA DISTRIBUÍDA DE DADOS EM WORKFLOWS DE BIOINFORMÁTICA

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA

JOSÉ GUILHERME MONTEIRO DE MENEZES

GERÊNCIA DISTRIBUÍDA DE DADOS EM WORKFLOWS DE BIOINFORMÁTICA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadora: Profa. Maria Cláudia Reis Cavalcanti - D.Sc.,
Co-orientadora: Profa. Fernanda Araujo Baião - D.Sc.,

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

C2008 Menezes, J. G. M.
Gerência Distribuída de Dados em Workflows de Bioinformática/ José Guilherme Monteiro de Menezes. – Rio de Janeiro: Instituto Militar de Engenharia, 2008.
xxx p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2008.

1. Workflows Científicos.
2. Distribuição de Dados.
I. Título. II. Instituto Militar de Engenharia.

CDD 629.892

INSTITUTO MILITAR DE ENGENHARIA

JOSÉ GUILHERME MONTEIRO DE MENEZES

GERÊNCIA DISTRIBUÍDA DE DADOS EM WORKFLOWS DE BIOINFORMÁTICA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadora: Profa. Maria Cláudia Reis Cavalcanti - D.Sc.,

Co-orientadora: Profa. Fernanda Araujo Baião - D.Sc.,

Aprovada em 10 de julho de 2008 pela seguinte Banca Examinadora:

Profa. Maria Cláudia Reis Cavalcanti - D.Sc., do IME - Presidente

---

Profa. Fernanda Araujo Baião - D.Sc., da UNIRIO

Profa. Marta Lima de Queirós Mattoso - D.Sc., da COPPE/UFRJ

Profa. Raquel Coelho Gomes Pinto - D.Sc., do IME

Rio de Janeiro
2008

Dedico esta à Maria Helena, José Altivo e Eliane.

AGRADECIMENTOS

Agradeço as minhas orientadoras, Dra. Maria Cláudia Reis Cavalcanti e Dra. Fernanda Araujo Baião, por toda a atenção dada durante o desenvolvimento deste trabalho. Pela paciência e, sobretudo, pela confiança a mim depositada.

Aos meus pais José Altivo Barreto de Menezes e Maria Helena Monteiro de Menezes, que junto com meu tio Amaro Monteiro Júnior me acompanharam por toda trajetória.

À minha namorada Eliane Oliveira Yang, por ter me apoiado em tantos momentos difíceis que tive durante este trabalho.

Agradeço a todos os meus amigos que contribuíram com o desenvolvimento desta dissertação de mestrado, tenha sido por meio de críticas, idéias, apoio, incentivo ou qualquer outra forma de auxílio.

Por fim, agradeço a Sérgio Abramovitch, por ter acreditado no meu trabalho. Sem sua ajuda eu não teria chegado aqui.

José Guilherme Monteiro de Menezes

Existe uma teoria que diz que, se um dia alguém descobrir exatamente para que serve o Universo e por que ele está aqui, ele desaparecerá instantaneamente e será substituído por algo ainda mais estranho e inexplicável. Existe uma segunda teoria que diz que isso já aconteceu.

DOUGLAS ADAMS

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

BD	-	*Banco de Dados*
GUI	-	*Graphical User Interface*
GUS	-	*Genomic Unified Schema*
IDE	-	*Integrated Development Environment*
IME	-	*Instituto Militar de Engenharia*
LSID	-	*Life Science Identifier*
MoML	-	*Modeling Markup Language*
PAD	-	*Problema de Alocação de Dados*
SBDD	-	*Sistema de Bancos de Dados Distribuídos*
SCUFL	-	*Simple Conceptual Unified Flow Language*
SGWf	-	*Sistemas de Gerência de Workflows*
WDK	-	*Web Development Kit*

RESUMO

Workflows científicos têm sido utilizados para a realização de experimentos in silico. Esses experimentos são caracterizados por serem executados com o apoio de computadores. Em suas execuções, uma seqüência de programas computacionais é processada e os dados de saída de um programa são compostos como dados de entrada no programa seguinte. No sentido de prover workflows científicos mais interoperáveis e flexíveis, a tecnologia de serviços Web vem sendo adotada pela comunidade científica como um facilitador para a disponibilização e acesso a programas científicos. Isso possibilitou que programas científicos utilizados em ambientes distintos pudessem ser integrados para comporem workflows científicos. Porém, a natureza distribuída dos serviços Web resgatou algumas questões referentes à gerência dos dados gerados e processados durante a execução dos experimentos - dados intermediários. Onde e como disponibilizar esses dados de modo que fiquem acessíveis para análises e reutilizações em futuras execuções dos experimentos foram algumas questões levantadas com o uso de serviços Web na composição de workflows científicos. Muitas propostas para o gerenciamento de workflows científicos centralizam o armazenamento de todos os dados intermediários num único servidor. Essa abordagem pode impactar negativamente a execução do workflow, devido ao custo de repetidas transferências de dados para o sítio de armazenamento central. Este trabalho apresenta uma arquitetura denominada D-BioFlow para a gerência distribuída de dados em workflows de Bioinformática. A partir desta arquitetura foi implementado o protótipo e-Bioflow e foram realizados alguns testes para validação da abordagem proposta. A concepção desta arquitetura baseou-se fortemente no domínio da Bioinformática, porém acredita-se que possa ser aplicada também a outros domínios científicos. Um exemplo de uso do protótipo implementado foi realizado tomando-se por base um workflow real de Bioinformática chamado STINGRAY, que vem sendo utilizado pelo grupo de pesquisa BioWebDB da Fundação Oswaldo Cruz (FIOCRUZ) há alguns anos.

ABSTRACT

Scientific workflows have been used to perform in silico experiments. These experiments are characterized for being supported by computers. During their executions, a sequence of computational programs is processed, and the output data of a program is set as the input data of the following program. To provide interoperable and flexible scientific workflows, the scientific community have been adopting Web services technology as a facilitator to deploy and access scientific programs. Thus, scientific programs used in different scientific environments could be integrated, as part of the same workflow. However, considering the Web services distributed characteristic, some issues regarding data generated and processed during workflow executions - intermediate data - have to be revisited. Where and how to store these data to be available for analysis and reuse in future experiments (re)executions, are some of these issues. Many scientific proposals of scientific workflow management concentrate the storage of all the intermediate data in a single server. This approach may impact the performance of a workflow execution, because of the cost of repeated transferences of data (intermediate data) to the central data server. This work presents the D-BioFlow architecture, which aims at the distributed management of data in Bioinformatics workflows. Based on this architecture, the e-Bioflow prototype was implemented and some tests were performed for the proposed approach validation. The architecture idea was strongly based on the Bioinformatics domain, however, it can also be applied to other scientific domains. An example of the implemented prototype usage was carried out based on a real Bioinformatics workflow called STINGRAY, which have been in use by the research group BioWebDB of the Oswaldo Cruz Foundation (FIOCRUZ) for a few years.

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Na área científica, há uma crescente oferta de programas de simulação, transformação e análise de dados, mais conhecidos como programas científicos (SILVA, 2006). O desenvolvimento destes programas científicos fomentou a criação de um novo tipo de experimento científico, denominado experimento *in silico*, que é definido como um experimento científico realizado utilizando apoio computacional intensivo (CAVALCANTI, 2005). Tipicamente, os experimentos *in silico* são definidos e executados utilizando-se a abordagem de workflows. Segundo (AALST, 1999), um workflow é definido como uma coleção de tarefas organizadas para a realização de um processo, onde uma tarefa denota um conjunto de softwares que implementam atividades específicas. Nos workflows de experimentos *in silico*, as tarefas correspondem aos programas científicos.

Em ambientes científicos, no entanto, a gerência de workflows apresenta características especiais. A existência de múltiplas combinações possíveis de programas e dados dificulta a definição dos workflows. Um mesmo workflow científico costuma ser executado múltiplas vezes, com parametrizações diferentes, devendo todas as execuções serem documentadas, mesmo quando os resultados não são satisfatórios. Muitas vezes, estas execuções são na verdade, re-execuções totais ou parciais de um mesmo workflow. Além disso, todos os resultados devem ser armazenados, mesmo os não satisfatórios. Os metadados referentes às execuções também devem ser registrados (quem executou e quando, que parâmetros foram utilizados, quais os resultados intermediários gerados, qual a seqüência de programas executados). Assim, o volume de dados manipulados pelos programas científicos pode ser muito grande.

Outra característica dos workflows científicos é o alto custo de processamento. A execução de um programa científico pode ser muito custosa. Em alguns casos, uma execução pode levar dias e consumir muitos recursos em termos de processamento e de espaço em disco para armazenamento dos dados resultantes. Por isso, os cientistas procuram evitar a repetição de tarefas desnecessariamente, realizando re-execuções parciais de um workflow sempre que possível. Por fim, os programas que constituem as

tarefas de um workflow científico, em geral, não foram concebidos para funcionar de forma encadeada pois foram desenvolvidos para sistemas operacionais específicos e/ou usam formatos pré-definidos para os dados de entrada e de saída. Isto acarreta em problemas de integração entre as tarefas. Assim sendo, um sistema de gerência de workflows científicos precisa prover mecanismos para atender a tais necessidades.

1.2 CARACTERIZAÇÃO DO PROBLEMA

Como foi dito anteriormente, durante a execução de um workflow científico, um grande volume de dados é produzido, sejam dados intermediários, resultantes do processamento das tarefas que compõem o workflow, ou dados finais, que constituem o resultado final do workflow. No ambiente científico, em especial na área de Bioinformática, os dados intermediários têm grande importância visto que podem ajudar na validação dos experimentos, podem ser utilizados na re-execução dos mesmos experimentos ou, ainda, podem ser utilizados na execução de outros experimentos.

Num cenário em que os workflows científicos têm execução distribuída, os custos de comunicação entre os servidores de processamento tornam-se impactantes no processamento, visto os grandes volumes de dados manipulados pelos workflows científicos na sua execução. Desta forma, o gerenciamento dos dados intermediários é fator relevante na busca de uma execução eficiente de workflows científicos.

Dentre os possíveis paradigmas de armazenamento de dados intermediários, destacam-se o armazenamento centralizado e distribuído. No paradigma de armazenamento centralizado, todos os dados intermediários da execução dos workflows são armazenados em um único servidor de dados. Já o paradigma de armazenamento distribuído visa a distribuição dos dados intermediários em diferentes servidores, de acordo com as características do ambiente em que o workflow é executado, como por exemplo, as limitações de cada servidor ou o histórico das execuções anteriores.

Recentemente, diversos programas científicos vêm sendo disponibilizados na forma de serviços Web [1], por grandes centros na área de Bioinformática. Assim, os workflows científicos nesta área passaram a ser construídos a partir de uma coleção de tarefas mapeadas em serviços Web locais ou remotos. A execução remota das tarefas que compõem um workflow, em conjunto com a existência de servidores de armazenamento

[1] http://www.w3.org/TR/ws-arch/, visitado em 02/06/2008

disponíveis na Internet (Storage Area Networks (OSZU, 1999)), tornou possível o armazenamento remoto dos dados intermediários e finais de um workflow.

Considerando workflows científicos baseados em serviços Web, no paradigma de armazenamento centralizado de dados, os serviços Web podem estar hospedados em sítios de difícil acesso por parte do gerente central de execução, acarretando problemas para transferência de dados. Mesmo num ambiente de gerência distribuída de dados, um possível hiato entre os sítios de armazenamento e os sítios de processamento pode trazer as mesmas conseqüências da alternativa centralizada, já que o custo de transferência de dados entre os nós de um ambiente distribuído é responsável pela maior parte do custo do processamento de consultas (OSZU, 1999) (RUBERG, 2002).

Considerando as abordagens propostas na literatura para a gerência de workflows científicos, com aplicação em biologia, como por exemplo o Kepler (LUDÄSCHER, 2006), o Taverna (OINN, 2004), o InServices (SILVA, 2006) e o Vistrails (CALLAHAN, 2006), nota-se que oferecem solução para a distribuição do processamento das tarefas, mas o armazenamento dos dados intermediários - quando possível - se dá de forma centralizada. Nessas abordagens, para garantir o armazenamento de tais dados, é preciso que o cientista indique manualmente o local, ou locais, de armazenamento desses dados. Entretanto, esta solução transfere o fardo da gerência dos dados intermediários para o usuário, encarregando-o de incluir no workflow, tarefas que realizem o armazenamento ou a recuperação desses dados. Portanto, nenhuma dessas abordagens apresenta uma solução adequada para o armazenamento distribuído de dados intermediários.

Apesar da existência de trabalhos cujo foco é a execução em ambientes distriuídos (como ambientes de grade), como Triana (TAYLOR, 2005), Askalon (FAHRINGER, 2005) e Swift (FOSTER, 2003). No entanto, esses trabalhos focam principalmente na distribuição de processos e não são voltados para aplicações de bioinformática. Assim sendo, neste cenário de execução distribuída de workflows científicos, é necessário que exista uma política estabelecida para o armazenamento e recuperação dos dados que trafegam entre as tarefas de um workflow, uma vez que a estratégia de armazenamento pode interferir, significativamente, no desempenho da (re)execução de um workflow.

1.3 VISÃO GERAL DA PROPOSTA

Este trabalho busca melhorar o desempenho de workflows distribuídos de bioinformática, identificando oportunidades de modo a aproximar os dados produzidos por cada tarefa

de seus sítios consumidores. Para verificar esta hipótese, este trabalho propõe uma abordagem distribuída de gerenciamento de dados para workflows de Bioinformática chamada **D-Bioflow**. A abordagem proposta baseia-se em workflows constituídos de serviços Web. A idéia é que, no momento em que se aciona um workflow para execução dos serviços que o compõem, de forma transparente ao usuário, sejam inseridos serviços Web para realizar o armazenamento e recuperação de dados intermediários. Assim, este trabalho propõe a extensão da funcionalidade dos sistemas de gerência de workflows científicos, através da adição de um módulo de pré-execução, responsável pela tomada de decisão em relação aos locais de armazenamento mais adequados para cada tarefa do workflow. Além disso, outros módulos e bancos de dados são também incluídos na arquitetura, com o objetivo de apoiar o módulo de pré-execução.

Baseado na especificação dessa abordagem, foi construído um protótipo, **e-BioFlow**, que além de permitir o gerenciamento distribuído dos dados intermediários, facilita as atividades de definição e execução de workflows, permitindo a re-execução parcial de workflows com reaproveitamento de dados de outras execuções quando possível.

1.4 CONTRIBUIÇÕES

Dentre as principais contribuições deste trabalho podem ser destacadas:

- Especificação de uma arquitetura de gerenciamento de dados distribuídos em workflows para bioinformática;
- Especificação de um metamodelo de dados;
- Construção de um protótipo que facilita a definição e execução de workflows;

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho apresenta a seguinte organização: o capítulo 2 fornece uma visão geral sobre workflows e distribuição de dados, apresentando as principais características dos workflows científicos e algumas das abordagens clássicas para a gerência de dados distribuídos. Nesse capitulo são analisados também os principais projetos existentes para a gerência de workflows científicos. No capítulo 3 é apresentada uma abordagem para a gerência de dados distribuídos em workflows de bioinformática, incluindo a apresentação de uma arquitetura, de um metamodelo de dados e de um modelo de custos. O capítulo 4 descreve

o protótipo **e-BioFlow**, construído com base na especificação da abordagem proposta. O capítulo 5 apresenta um modelo de simulação criado para validar a abordagem proposta. O capítulo 6 apresenta um exemplo de uso do protótipo **e-BioFlow** e finalmente, o capítulo 7 apresenta as conclusões do trabalho, as contribuições obtidas e trabalhos futuros.

2 WORKFLOWS E DISTRIBUIÇÃO DE DADOS

Neste capítulo serão apresentados alguns conceitos básicos envolvendo workflows, workflows científicos e sistemas de gerência de workflows científicos. Além disso, o capítulo apresenta também uma breve revisão das técnicas de gerência de dados distribuídos e sobre a gerência de dados em ambientes de workflows científicos distribuídos que servirão de base para o restante deste trabalho.

2.1 WORKFLOWS

Encontram-se na literatura várias definições e formas de representação de workflows (AALST, 1999) (CAVALCANTI, 2005), bem como várias abordagens para a gerência de workflows como Kepler, Vistrails, Taverna e InServices. Algumas destas foram selecionadas e apresentadas aqui para o embasamento teórico deste trabalho.

Segundo (AALST, 1999), um workflow é definido como uma coleção de tarefas organizadas para a realização de um processo específico, onde o conceito de tarefa é utilizado para definir um programa de computador que implementa uma atividade específica. Uma formalização de workflow é proposta em (CAVALCANTI, 2005) onde um workflow é definido como W(T, V, Sf, Cf), onde:

- T: Conjunto de tarefas de W;
- V: Conjunto de variáveis de W definindo um fluxo de dados;
- Sf: Função sucessora associada a cada tarefa $t \in T$;
- Cf: Função de condição associada a cada tarefa $t \in T$;

Uma outra forma de representação de workflows é apresentada por (LEMOS, 2006). Neste trabalho, um workflow é representado por dois submodelos: um modelo de Aplicação e um modelo de Fluxo. Uma tarefa de workflow, normalmente, é mapeada num programa de computador. O modelo Aplicação é responsável pela modelagem das características dos programas que serão invocados durante a execução do workflow. O modelo de Fluxo possui uma abstração do controle de fluxo de dados e captura os padrões de workflow

já reconhecidos, como por exemplo, elementos que indicam uma execução seqüencial ou paralela de tarefas, ou ainda para representar quando múltiplas linhas de execução convergem para uma única seqüência de execução.

Em especial para este trabalho, cujo foco é a gerência de dados intermediários, são de grande interesse os modelos propostos, pois estes evidenciam os dados que fluem através das tarefas de um workflow. No modelo Aplicação há duas classes principais: Processo e Contêiner, cujas instâncias são, respectivamente, denominadas processos (p) e contêineres (c). Considere p um programa que implementa uma tarefa de um workflow; p possui parâmetros de configuração, entradas e saídas de dados; uma subclasse de Processo, Pr[p], será definida para que o programa p possa ser invocado na execução do workflow. Para cada parâmetro A de p, a classe Pr[p] terá uma propriedade Pa[p,A]. A classe Pr[p] terá também as seguintes meta-propriedades:

- input-ports: cujos valores formam uma lista (i_1,...,i_n) de nomes que identificam os dados de entrada de p, seus elementos são denominados portas de entrada;
- output-ports: cujos valores formam uma lista (o_1,...,o_n) de nomes que identificam os dados de saída de p, seus elementos são denominados portas de saída;
- input-ports-type: cujos valores formam uma lista (t_1,...,t_n) que identificam o tipo de cada porta de entrada de p;
- output-ports-type: cujos valores formam uma lista (v_1,...,v_n) que identificam o tipo de cada porta de saída de p;

No modelo de Aplicação foi necessário evidenciar se o comportamento das portas de saída e de entrada possuem comportamento gradual ou não-gradual. Numa porta gradual, o dado é consumido pelo processo à medida que está sendo disponibilizado. Numa porta não-gradual o dado só é consumido quando todo o conjunto de itens é disponibilizado.

O modelo de Fluxo é formalizado através de uma única classe Workflow. Ao ser instanciado, um workflow é apresentado na forma de um grafo bipartite rotulado (grafo cujo conjunto de vértices pode ser dividido em dois subconjuntos disjuntos), cujos nós são instâncias do modelo Aplicação. Um exemplo de instância da classe Workflow é ilustrado na figura 2.1, onde:

- O conjunto de nós é composto por instâncias de containers (c) e processes (p);

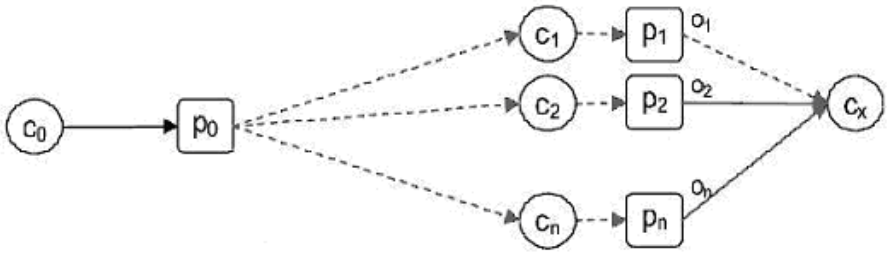

FIG. 2.1: Exemplo de instância da classe Workflow(LEMOS, 2006)

- O conjunto de arcos contém pares da forma (c,p) ou (p,c), onde p é nó do tipo processo e c é um nó do tipo container;
- Os arcos têm o mesmo comportamento das portas que ligam. Um arco tracejado representa um arco de comportamento gradual, enquanto um arco sólido representa um arco não-gradual;

2.2 WORKFLOWS EMPRESARIAIS X WORKFLOWS CIENTÍFICOS

Há trabalhos anteriores em que foram identificadas algumas características que diferenciam os workflows utilizados em ambientes empresariais dos workflows utilizados em ambientes científicos (MEDEIROS, 1995), (CAVALCANTI, 2003). Dentre os principais aspectos diferenciadores, pode-se destacar: a freqüência de redefinições, o tempo de execução das tarefas, os resultados considerados, a rastreabilidade, o volume de dados e a movimentação dos dados. Estes aspectos são detalhados a seguir.

No meio empresarial, nota-se relativa estabilidade dos workflows, quando comparados aos do meio científico. Normalmente, um workflow empresarial só tem sua definição modificada quando há mudança no processo que o mesmo automatiza. No meio científico, devido à natureza das atividades experimentais, os workflows são constantemente redefinidos. No meio científico, é comum que um workflow seja aproveitado para definição de outros, no contexto desse trabalho os novos workflows criados são denominados workflows derivados, e o workflow que serve de matriz é denominado workflow original.

Nota-se, em workflows empresariais, que as tarefas normalmente demandam pouco tempo de processamento, considerando-se que o volume de dados envolvido é tipicamente pequeno e que as tarefas são em geral pouco custosas. Nos workflows científicos o

comportamento costuma ser contrário. Normalmente as tarefas implementam algoritmos que demandam alto tempo de processamento.

Outra diferença importante diz respeito à análise dos resultados. Nos workflows científicos todos os resultados são importantes, indiferentemente se foram resultados corretos ou não. Todos os dados devem ser armazenados para posterior análise. Já no meio empresarial, só importa o resultado correto, os demais são normalmente descartados.

Em workflows empresariais o critério de rastreabilidade dos dados obtidos, é empregado para fins de auditoria de sistemas e acompanhamento da execução. Em workflows científicos, a rastreabilidade é importante para permitir que haja proveniência de dados. Segundo (BUNEMAN, 2001), proveniência de dados é a descrição das origens de um item de dado e do processo pelo qual foi produzido. Outro aspecto importante a ser levado em conta é que a proveniência possibilita re-execuções parciais de um workflow. Esse recurso permite que um workflow possa ser re-executado a partir de uma tarefa intermediária. Desta forma, caso um workflow tenha sua execução interrompida, esta poderá ser reiniciada a partir do ponto de interrupção, trazendo uma relevante economia de tempo, visto que as tarefas pregressas não precisarão ser re-executadas.

Os workflows científicos normalmente manipulam grandes volumes de dados. Tipicamente, como ocorre nos ambientes experimentais, um mesmo procedimento é repetido diversas vezes. De forma análoga, um workflow pode ser executado múltiplas vezes com diferentes parâmetros. E já que nenhum resultado deve ser descartado, multiplica-se a quantidade de dados gerada por um workflow pelo número de suas re-execuções. Já os workflows empresariais, normalmente manipulam pequenos volumes de dados, descartam-se os resultados incorretos, além de ocorrer, para cada caso, uma única execução do workflow. Assim, pode-se considerar que o volume de dados gerado é bem inferior ao gerado pelos ambientes científicos.

Deve-se considerar ainda, a importância dos dados intermediários. Como os workflows científicos demandam grande tempo de processamento, pode ser interessante para o cientista reutilizar os dados de forma a não ser necessário re-executar todas as tarefas. Segundo (LUDÄSCHER, 2006), uma re-execução parcial ocorre quando um workflow não é executado a partir de sua tarefa inicial, mas sim a partir de uma tarefa intermediária do encadeamento. A re-execução parcial só é possível via o armazenamento dos resultados produzidos por todas as tarefas que compõem o workflow. Outra razão para se manter os dados intermediários é que nos ambientes científicos, como foi dito anteriormente, há uma

demanda por parte da gerência do laboratório, não só para acompanhar os experimentos que têm sido realizados, como também para oferecer a possibilidade de chegar às origens dos dados, isto é, ter acesso à proveniência dos dados, seja do processo a partir do qual estes foram gerados, seja dos dados fonte e intermediários envolvidos neste processo.

É comum encontrar tanto no meio empresarial quanto no científico, um ambiente de distribuição de processamento. Em ambos os meios, os sítios de processamento podem estar geograficamente dispersos. Porém, no meio empresarial, encontra-se um cenário denominado intra-organizacional, onde os dados são transferidos entre a matriz e filiais obedecendo a critérios hierárquicos. No meio científico o cenário é de colaboração entre instituições de pesquisa independentes e autônomas, caracterizando-se então como um cenário de movimentação de dados interorganizacional. A tabela 2.1 resume os itens apresentados nessa seção.

TAB. 2.1: Comparação entre os workflows empresariais e científicos

Características	Empresariais	Científicos
Redefinições	Poucas	Constantes
Tempo de Execução das Tarefas	Tipicamente Baixo	Alto
Resultados	Resultado Certo	Todos os Resultados são Importantes
Rastreabilidade	Auditoria/ Controle de Execução	Proveniência de Dados
Volume de Dados	Tipicamente Pequeno	Muito Grande
Movimentação de Dados	Intra-Organizacional	Interorganizacional

Num cenário de distribuição de processamento em workflows científicos, onde as tarefas encontram-se implementadas em serviços Web, e podem estar publicadas em diversos servidores remotos, faz sentido pensar, também, na distribuição do armazenamento dos dados intermediários produzidos durante as execuções dos workflows. A seção seguinte traz uma breve revisão sobre a distribuição de dados, ainda dissociada do contexto de workflows, e discute os principais problemas envolvidos no seu gerenciamento.

2.3 DISTRIBUIÇÃO DE DADOS

O paradigma da gerência distribuída de dados em workflows e em Sistemas de Bancos de Dados Distribuídos envolve o problema da alocação de dados, que consiste em encontrar a distribuição "ótima" dos dados através dos sítios de armazenamento do ambiente de

forma a minimizar o custo das aplicações sobre esses dados (OSZU, 1999). Nos Sistemas de Bancos de Dados Distribuídos (SBDD) a alocação dos dados é um aspecto crítico, visto que uma alocação ineficiente dos dados pode levar a um aumento significativo do custo de acesso (WILDEMBERG, 2003). Este fato também ocorre nos workflows com armazenamento distribuído de dados intermediários. Na alocação de dados, busca-se aumentar a proximidade entre os dados e os sítios de processamento que os utilizam, minimizando assim o custo de comunicação.

Para aumentar o desempenho das aplicações via distribuição de dados, deve-se realizar um planejamento prévio, denominado Projeto de Distribuição de Bases de Dados, que aproveite a localidade de acesso das informações (BAIÃO, 2004).

2.3.1 PROJETO DE DISTRIBUIÇÃO DE BASES DE DADOS

Segundo (OSZU, 1999), um Projeto de Distribuição define como será o armazenamento dos dados entre os sítios da rede de um sistema distribuído. Basicamente, há duas estratégias para projetar a distribuição de bases de dados (OSZU, 1999): ascendente (*bottom-up*) e descendente (*top-down*).

A estratégia ascendente parte dos esquemas conceituais locais (que contêm a descrição do modelo de dados particular de cada sítio), e integra-os para formar um esquema conceitual global (com a descrição do modelo de dados da base como um todo) através da criação de uma camada de integração. Esta estratégia é adequada num processo de integração de bases de dados já existentes, especialmente se os sistemas envolvidos apresentarem algum tipo de heterogeneidade, quando então a camada de integração se responsabilizará pela resolução dos conflitos existentes(OSZU, 1999).

A estratégia descendente propõe a construção dos esquemas locais a cada sítio a partir de um esquema conceitual global, através da fragmentação e da alocação dos dados, que são as suas duas etapas principais. A estratégia descendente é adequada em situações como no desenvolvimento de uma base de dados (*from scratch*), ou em sistemas em que os SGBDs componentes são fortemente integrados (OSZU, 1999).

As etapas de um projeto de distribuição descendente são: a fragmentação e a alocação (OSZU, 1999). A etapa de fragmentação consiste no particionamento do conjunto de dados em subconjuntos disjuntos, denominados fragmentos de dados. Tradicionalmente, há três tipos de fragmentação: horizontal, vertical e híbrida, detalhadas a seguir. A etapa de alocação pode também envolver a replicação de fragmentos em mais de um sítio.

2.3.2 FRAGMENTAÇÃO

Na fragmentação horizontal, os dados são agrupados aplicando-se predicados de seleção sobre o conjunto de dados. No modelo relacional, subconjuntos de tuplas de uma relação são agrupados de acordo com um critério de seleção. Na fragmentação vertical, os dados são agrupados seguindo um critério de projeção. Desta forma, há o particionamento das estruturas de dados. A fragmentação híbrida é composta pela combinação de critérios de fragmentação horizontal e vertical.

2.3.3 ALOCAÇÃO

Na etapa de alocação, busca-se encontrar a melhor distribuição dos fragmentos, pelos possíveis sítios de armazenamento disponíveis no ambiente, de forma a aproximar os dados dos sítios que os acessam, seguindo critérios de disponibilidade, custos de comunicação e de armazenamento, etc.

O principal aspecto que determina a qualidade da solução encontrada para o problema de alocação de dados é a eficiência do projeto de alocação, ou seja, a alocação dos dados nos nós deve minimizar, o quanto possível, o custo das consultas executadas. O processo de tomada de decisão presente no Problema de Alocação de Dados (PAD) em um SBDD, de uma forma geral, pode ser definido como: escolher dentre todos os possíveis modelos de alocação de fragmentos nos sítios de uma rede, ou seja, entre todas as possíveis distribuições dos fragmentos, aquela que minimize o custo de acesso aos dados alocados considerando um conjunto de restrições do problema, como por exemplo, a capacidade dos nós onde os dados são alocados (WILDEMBERG, 2003). O PAD inclui ainda a decisão de replicação dos fragmentos em outros sítios. Este tipo de problema é classificado na literatura como Problema de Otimização Combinatória.

Para atingir seu objetivo, o Problema de Alocação de Dados deve levar em consideração fatores do ambiente, tais como (OSZU, 1999):

- **Informações do Banco de Dados:** precisa-se observar a seletividade de um fragmento F_j quando aplicada uma query q_i. Isto é, o número de tuplas de F_j acessadas para o processamento de q_i. O valor pode ser obtido através de uma função definida como $sel_i(F_j)$. Outra informação necessária diz respeito ao tamanho do fragmento. Pode-se definir uma função $size(F_j) = card(F_j) * length(F_j)$, onde $card(F_j)$ retorna a cardinalidade do fragmento e $length(F_j)$ retorna o tamanho em

bytes de uma tupla do fragmento F_j;

- **Informações da Aplicação:** duas importantes informações devem ser observadas quanto às aplicações: número de acessos de leitura que uma query q_i, realiza em um determinado fragmento F_j e o número de acessos de escrita (atualização) realizado num fragmento. Deve-se também determinar duas matrizes UM e RM, cujos elementos u_{ij} e r_{ij}, são definidos da seguinte forma:

$$u_{ij} = \begin{cases} 1, & se \quad a \quad query \quad q_i \quad atualiza \quad F_j \\ 0, & caso \quad contrário \end{cases}$$

$$\mathrm{r}_{ij} = \begin{cases} 1, & se \quad a \quad query \quad q_i \quad acessa \quad dados \quad do \quad fragmento \quad F_j \\ 0, & caso \quad contrário \end{cases}$$

- **Informações sobre os Sítios:** para cada sítio do ambiente, deve-se levantar informações quanto a capacidade de armazenamento e processamento. Esses valores podem ser levantados por simples estimativas ou através de funções específicas;

- **Informações sobre a Rede:** os custos envolvidos na comunicação entre os sítios do ambiente devem ser avaliados. Normalmente, define-se uma unidade padrão de custo, por exemplo por frame de dado trafegado;

Para realizar a alocação dos dados de forma a minimizar os custos de comunicação, diversos modelos de custos são encontrados na literatura de banco de dados (OSZU, 1999). Normalmente os modelos de distribuição baseiam-se em funções de custo que levam em conta restrições do ambiente, custos de comunicação, custos de processamento, armazenamento e recuperação de dados, permissionamento e disponibilidade dos servidores, e que visam encontrar o melhor local de armazenamento. Segundo (WILDEMBERG, 2003), o problema de encontrar uma alocação ótima de fragmentos em um SBDD é NP-Difícil. Segundo (WILDEMBERG, 2003), dado um problema com n fragmentos e m nós, $(2m-1)^n$ é o número de combinações possíveis para o problema. Desta forma, o uso de metodologias que buscam solucionar o problema analisando todas as possibilidades são inviáveis devido à complexidade computacional do problema. Assim, justifica-se o uso de heurísticas para obtenção de soluções boas, porém não-ótimas num tempo viável.

No domínio de workflows com armazenamento distribuído de dados, pode-se aplicar as técnicas de alocação de fragmentos de dados para definir o local de armazenamento dos

dados intermediários produzidos durante a execução dos workflows. Pode-se ainda utilizar os conceitos apresentados das matrizes de acesso de leitura e atualização de dados, visto que as tarefas dos workflows podem ser vistas como aplicações que acessam os dados. No contexto de workflows, pode-se observar a repetida associação de pares de tarefas produtoras e consumidoras. Neste trabalho, define-se como tarefa produtora aquela que produz um determinado conjunto de dados, e tarefa consumidora aquela que utiliza o dado, produzido pela tarefa anterior, para executar o seu processamento.

Pode-se acrescentar novos aspectos a serem analisados como:

- Identificação de tarefas consumidoras: dada uma tarefa t_i, deve-se construir uma matriz de freqüência de ocorrência da tarefa t_{i+1}, nos workflows existentes, no intuito de identificar as prováveis tarefas consumidoras de t_i. Desta forma, pode-se alocar os dados produzidos por t_i, junto aos sítios das tarefas consumidoras;

- Workflows derivados: deve-se identificar a existência de workflows derivados, buscando oportunidades de reaproveitar dados gerados por execuções anteriores de workflows semelhantes. Considere como exemplo o workflow W1: $T1- > T2- > T4- > T5$ e o workflow W2: $T1- > T2- > T6- > T7$. W2 é derivado de W1, visto que eles se diferenciam a partir da terceira tarefa. Desta forma, o workflow W2 pode aproveitar dados produzidos por parte da execução de W1(até o ponto de derivação) e vice-versa;

- Freqüência de execução de workflows: deve-se levar em consideração a freqüência de execução dos workflows, de forma a alocar os dados produzidos junto aos sítios de processamento utilizados com maior freqüência;

2.3.4 FORMALIZAÇÃO DE UM PROBLEMA DE ALOCAÇÃO DE DADOS

Um exemplo de PAD é mostrado em (WILDEMBERG, 2003), cuja meta é encontrar uma matriz de alocação **FAT** (Fragment Allocation Table), que representa a distribuição dos fragmentos nos nós de forma a minimizar o custo da execução das transações. Considere $S = s_1, s_2, ..., s_m$ o conjunto de nós, onde m é o número de nós da rede, $T = t_1, t_2, ..., t_q$ o conjunto de transações que são executadas em S, onde q é o número de transações, e $F = f_1, f_2, ..., f_n$ o conjunto de fragmentos resultantes da fase de fragmentação do projeto de distribuição, onde n é o número de fragmentos, o custo total para a execução de cada t_k disparada de cada s_j sobre os diversos fragmentos envolvidos em t_k deve ser o menor

possível. A modelagem do problema é baseada no conjunto de parâmetros discutidos anteriormente:

a) Parâmetros do banco de dados: os tamanhos dos fragmentos, representados por um vetor TAM, onde *tam(i)* é o tamanho do fragmento f_i;

b) Parâmetros das transações: incluem quatro matrizes:

- A matriz RM representa os pedidos de leitura, onde rm_{ki} representa o número de vezes que a transação t_k faz acesso de leitura ao fragmento f_i, a cada vez que é executada. Este valor é inteiro e positivo podendo ser 0 caso a transação t_k não acesse o fragmento f_i para leitura;
- A matriz UM representa os pedidos de atualização, onde um_{ki} representa o número de vezes que a transação t_k atualiza o fragmento f_i, a cada vez que é executada. Como no caso anterior, um_{ki} é um valor inteiro e positivo podendo ser igual a 0;
- A matriz SEL representa a seletividade das transações, onde $sel(k_i)$ representa o percentual do volume de dados de f_i que é acessado durante a execução da transação t_k;
- A matriz FREQ representa a freqüência de execução de cada transação em cada nó, onde $freq_{kj}$ é o número de vezes que o nó s_j dispara a transação t_k;

c) Parâmetros da rede: incluem o custo de transferir uma unidade de dado entre os nós da rede e o custo de construção do circuito virtual entre dois nós. O primeiro parâmetro é representado por uma matriz CTR, onde ctr_{jl} é o custo de transferir um dado do nó s_j para o nó s_l. Para simplificar o problema, CTR é uma matriz simétrica onde ctr_{ii} é igual a 0 para $1 < i < m$. O segundo parâmetro, VC_{ini}, diz respeito ao circuito virtual que é criado entre o nó que dispara a transação e o nó que possui um fragmento acessado por esta transação. Com o fim da transação este circuito é fechado. Além deste, C_{ini} é um custo constante de iniciar a transmissão de um pacote de dados de tamanho P_SIZE.

De posse de tais parâmetros, define-se uma função de custo a partir da qual pode-se estimar o custo da execução de um conjunto de transações sobre uma base de dados distribuída. No exemplo em questão, pressupõe-se uma alocação estática dos fragmentos.

A função de minimização do custo que foi utilizada nesse exemplo é definida pela equação I:

$$Min(CC_{proc} = \sum_{j=1}^{m} \sum_{k=1}^{q} freq_{kj} * (TR_k + TU_k + VC_{ini})) \text{ (I)}$$

Onde TR é o custo de recuperação relacionado às transações de leitura, detalhado a seguir. Supondo uma transação t_k que realiza um acesso de leitura ao fragmento f_i e é disparada a partir do nó s_j, TR_k (definido na equação II) representa o custo de executar a transação t_k. O nó que será lido durante a execução da transação t_k é o que apresenta o menor custo de comunicação (CC_{com}) para transferir os dados consultados para o nó s_j, dentre todos os nós que alocam uma réplica do fragmento f_i. Equação II:

$$TR_k = \sum_{i=1}^{n} rm_{ki} * min(CC_{com}(ctr_{js} \quad _{where} \quad _{fat_{i,s}=} \quad _{1}, \quad sel_{ki} * tam_i))\text{(II)}$$

Onde fat_{ij} é igual a 1 se o fragmento f_i está alocado no nó s_j, senão é igual a 0 e CC_{com} é uma função definida na equação III:

$$CC_{com}(ctr_{jl}, m\_size) = Cini * \frac{m\_size}{p\_size} + ctr_{jl} * m\_size \text{ (III)}$$

Onde m_size é o tamanho do fragmento e p_size é a capacidade de transmissão da rede. Para o cálculo do custo relacionado às transações de atualização (TU), supondo uma transação t_k que atualiza o fragmento f_i e é disparada a partir do nó s_j, TU_k, definido na equação 4, indica a soma dos custos de comunicação para transferir os dados atualizados pela transação t_k a partir de s_j em todos os nós que alocam o fragmento f_i de forma a garantir a consistência de todas as réplicas de f_i no sistema distribuído. Equação IV:

$$TU_k = \sum_{i=1}^{n} um_{ki} * (\sum_{l=1}^{m} fat_{i,l} * CC_{com}(ctr_{jl}, sel_{ki} * tam_i)) \text{ (IV)}$$

2.4 GERÊNCIA DE DADOS EM WORKFLOWS CIENTÍFICOS DISTRIBUÍDOS

Conforme dito anteriormente, nos ambientes científicos os dados intermediários têm grande importância e não devem ser descartados. Destaca-se ainda a importância da proveniência de dados, onde deve-se armazenar - além dos dados - as informações de como os dados foram obtidos. Com o advento dos serviços Web, tornou-se possível utilizar programas científicos distribuídos pela internet que, via compartilhamento de recursos

computacionais, permitiu maior colaboração entre os centros de pesquisa. Porém, nesse cenário de distribuição de processamento, surge a questão: como gerenciar adequadamente o grande volume de dados intermediários e finais produzidos pelas execuções dos Workflows? A literatura de banco de dados propõe dois paradigmas para armazenamento de dados: **centralizado** e **distribuído**. Paradigmas estes que serão detalhados nas seções a seguir.

2.4.1 ARMAZENAMENTO CENTRALIZADO

No paradigma de armazenamento centralizado de dados, os dados produzidos durante a execução dos workflows são armazenados num único repositório do ambiente. Esse repositório pode estar hospedado no gerente de execução, ou em qualquer servidor da arquitetura que possua grande capacidade de armazenamento. Desta forma, o dado produzido por cada tarefa de um workflow em execução é enviado para o repositório central, e deste para a tarefa subseqüente para dar continuidade à execução. A figura 2.2 ilustra o esquema de funcionamento centralizado.

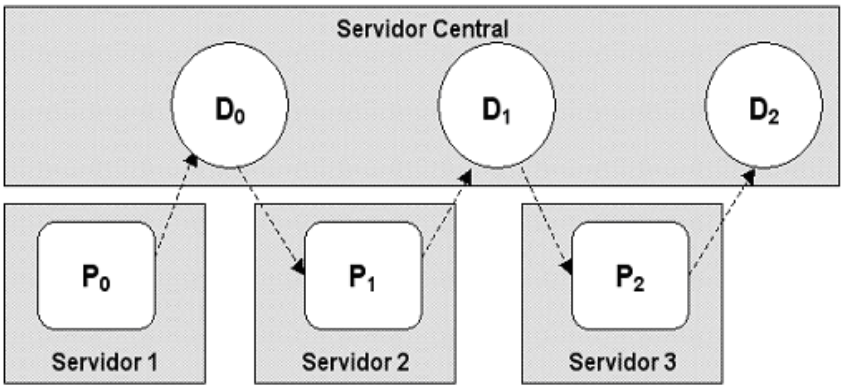

FIG. 2.2: Esquema de armazenamento centralizado

Devido ao seu modo de funcionamento, o processo de armazenamento e recuperação dos dados gerados por cada tarefa, em múltiplas execuções de diversos workflows, pode apresentar baixo desempenho, prejudicando as execuções dos workflows. Num ambiente de workflows científicos, em que tipicamente os sítios de processamento estão dispersos geograficamente, a infra-estrutura de rede torna-se um ponto impactante na execução do workflow, visto que os sítios de processamento podem estar localizados em locais de acesso dificultado por parte do sítio central de armazenamento.

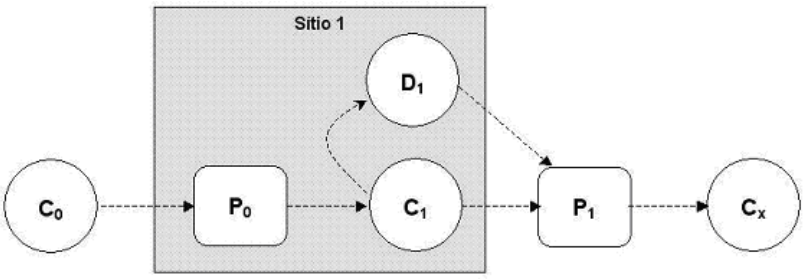

FIG. 2.3: Armazenamento no sítio produtor

2.4.2 ARMAZENAMENTO DISTRIBUÍDO

No paradigma de armazenamento distribuído de dados, vários repositórios de dados encontram-se disponíveis no ambiente. Os dados produzidos pelas tarefas dos workflows que estão em execução podem ser armazenados em qualquer um dos repositórios de dados disponíveis.

Considere uma instância w' de um workflow w composto por i tarefas $t_1, t_2, ..., t_i$. Considere um conjunto de n sítios de processamento, onde $t_{i,j}$ significa que a tarefa i está hospedada no sítio j. Suponha que $t_{i,p}$ e $t_{i+1,r}$, $p \neq r$ represente um par de tarefas subseqüentes em w', hospedadas nos sítios p e r, respectivamente. Suponha que o dado d produzido por $t_{i,p}$ seja consumido por $t_{i+1,r}$. Considere que todos os sítios de processamento possuam repositórios de dados. Neste cenário, uma possível estratégia de armazenamento seria persistir d sempre no sítio de t_i, neste caso o sítio p, no intuito de armazená-lo junto ao sítio produtor, como ilustra a figura 2.3; outra estratégia seria sempre armazenar d no sítio de t_{i+1}, neste caso o sítio r, buscando armazená-lo junto ao processo consumidor, ilustrado na figura 2.4. Pode-se pensar ainda em armazenar d em um outro sítio de armazenamento qualquer do ambiente, ilustrado na figura 2.5. No entanto, esta decisão não é tão simples, considerando que deve ser tomada para cada dado intermediário produzido durante a execução do workflow, e levando-se em conta restrições do ambiente que devem ser observadas, como limitação no espaço de armazenamento, políticas de administração do sítio remoto, autorização de acesso entre os sítios, dentre outras.

Armazenar o dado intermediário sempre no sítio onde o mesmo foi produzido, ou consumido, pode não ser uma boa estratégia de armazenamento distribuído em todos os casos, visto que a comunicação entre os sítios produtor e consumidor pode ser deficiente, causando lentidão no armazenamento e transmissão de dados. Pode-se analisar

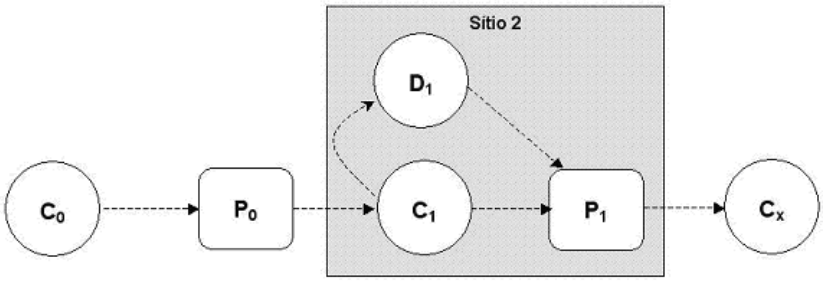

FIG. 2.4: Armazenamento no sítio consumidor

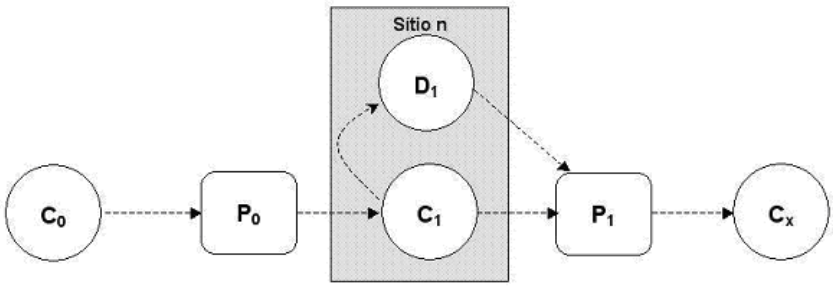

FIG. 2.5: Armazenamento num sítio qualquer do ambiente

ainda o aspecto de aproveitamento de dados produzidos por execuções anteriores de um workflow. Desta forma, o dado produzido por uma instância w' do workflow w poderia ser aproveitado em outra instância w". Sob esse prisma, deve-se escolher um repositório de dados levando em conta a possível utilização futura do mesmo.

O problema de armazenamento distribuído de dados é estudado, há bastante tempo, na área de sistemas de banco de dados distribuídos. Técnicas de distribuição têm sido empregadas com sucesso no incremento de desempenho de aplicações que manipulam grandes volumes de dados. No cenário de workflows distribuídos, as técnicas de distribuição de dados possibilitam adequar o armazenamento dos dados intermediários de acordo com a necessidade de processamento dos mesmos pelas instâncias de tarefas durante a execução de workflows científicos. Entretanto, o domínio dos workflows acrescenta novos complicadores não tratados adequadamente por bancos de dados distribuídos, como o encadeamento de execuções e a interdependência de dados e tarefas. No domínio de workflows - conforme já apresentado - as tarefas possuem regras de encadeamento, onde uma tarefa precisa do dado produzido pela sua antecessora para poder executar, esse aspecto é conhecido como interdependência de dados e tarefas. Mas o dado produzido por uma tarefa pode também ser utilizado por diversas outras tarefas. Assim, um projeto de distribuição de dados intermediários de workflows deve considerar

o aspecto de tarefas prováveis consumidoras. Como no paradigma de armazenamento distribuído os dados encontram-se dispersos nos vários repositórios do ambiente, há a necessidade de um mecanismo eficiente para identificar os dados no ambiente distribuído como um todo, a fim de viabilizar o seu armazenamento e recuperação durante a execução dos workflows. Um mecanismo que tem sido adotado por sistemas científicos é o LSID (CLARK, 2004), que será detalhado na subseção a seguir.

2.4.3 LSID

O LSID (Life Science Identifier) é um mecanismo de identificação de dados baseado em URNs (Universal Resource Names) que faz parte da OMG (Object Management Group) desde 2004. A estrutura de um LSID é demonstrada a seguir:
$< LSID >::='urn : lsid :'< AuthorityID >':'< AuthorityNamespaceID >':'< ObjectID > [':'< RevisionID >]$
Alguns exemplos de LSID são demonstrados a seguir:

- URN:LSID:rcsb.org:PDB:1D4X:22
- urn:lsid:kepler-project.org:director:1:1

Observa-se que no formato do LSID, o AuthorityID é constituído pelo domínio do proprietário da informação na internet. Esse mecanismo é utilizado para localização e recuperação da informação. AuthorityNamespaceID destina-se a informações complementares do proprietário do dado, no caso de URN:LSID:rcsb.org:PDB:1D4X:22 o AuthorityNamespaceID PDB refere-se ao nome do banco de dados. O ObjectID é um identificador para o dado, o RevisionID - parte opcional - é um identificador de versionamento da informação.

2.5 SISTEMAS DE GERÊNCIA DE WORKFLOWS CIENTÍFICOS

Para apoiar as etapas de definição e execução de workflows científicos, foram desenvolvidos sistemas de gerência de workflows científicos. Dentre os projetos analisados, destacam-se os sistemas de gerência de workflows científicos Kepler (LUDÄSCHER, 2006), VisTrails (CALLAHAN, 2006), InServices (SILVA, 2006) e Taverna (OINN, 2004). De forma geral os sistemas de gerência de workflows apresentam funcionalidades para apoiar a definição dos workflows, como editores gráficos, suportam a chamada de serviços Web.

2.5.1 KEPLER

O Kepler é um sistema de gerência de workflows científicos em geral, baseado no projeto PtolomeuII [2]. O projeto Ptolomeu vem sendo desenvolvido por um grupo de pesquisa da Universidade de Berkeley que há mais de vinte anos pesquisa e desenvolve ferramentas de modelagem. O PtolomeuII refere-se à terceira geração de software produzida pelo grupo.

A figura 2.6 mostra a arquitetura do Kepler, evidenciando o Ptolomeu como um de seus módulos. No Kepler, um workflow é composto por um diretor e um conjunto de atores. O diretor é responsável por controlar a execução do workflow seguindo um modelo computacional específico. Desta forma, por exemplo, um diretor do tipo PN (Process Network) permite que as tarefas sejam executadas em trilhas de execução independentes, já um diretor SDF (Synchronous Dataflow) implementa uma execução serial das tarefas.

Os atores modelam as tarefas propriamente ditas. Na versão avaliada para elaboração desse trabalho, versão 1.0 beta3, o Kepler dispunha de uma coleção de mais de duzentos atores responsáveis por tarefas que vão desde manipulação de strings à interação com Serviços Web. Os atores podem ainda ser classificados em simples ou compostos. Um ator do tipo simples realiza uma tarefa pontual no workflow, um ator do tipo composto encapsula um conjunto de atores simples, representando um subworkflow. Assim, um workflow é definido no Kepler como um ator do tipo composto. Os atores interagem entre si através de um mecanismo de portas, cada ator possui uma ou mais portas que podem ser dos tipos: entrada, saída ou entrada e saída - de acordo com a necessidade da atividade que o respectivo ator implementa.

O Kepler estendeu a linguagem de modelagem MoML (Modeling Markup Language), originária do PtolomeuII. Todos os atores e diretores possuem um identificador único no projeto, baseado no mecanismo de LSID (Life Science Identifier) descrito anteriormente. Por exemplo, um diretor SDF possui o LSID urn:lsid:kepler-project.org:director:1:1. O interessante de se notar aqui é que o uso de LSID foi concebido inicialmente para identificar dados, mas no caso do Kepler passou a ser usado para identificar outro tipo de recurso.

O módulo Vergil é a GUI (Graphical User Interface) herdada do projeto PtolomeuII que teve algumas funcionalidades acrescentadas tais como a integração com o repositório central de atores e a geração de Kepler ARchives (KAR). Os módulos de autenticação e proveniência, até o momento da escrita deste trabalho, ainda estavam em fase de desenvolvimento. O módulo de autenticação será responsável por prover um mecanismo de

[2] http://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm

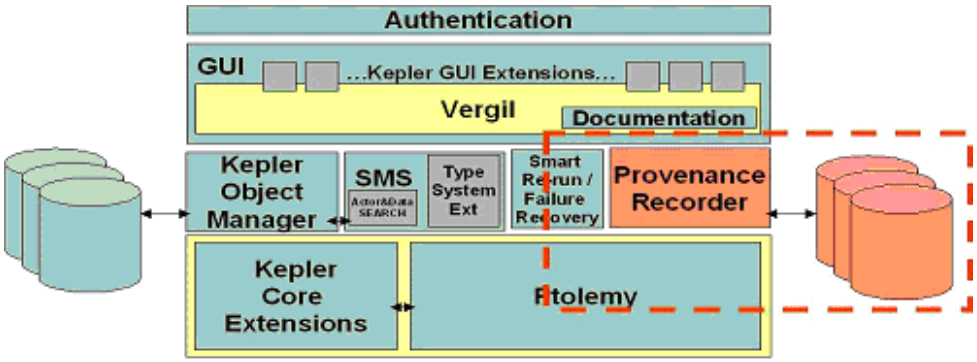

FIG. 2.6: Arquitetura do Kepler (ALTINTAS, 2006)

autenticação/autorização com uso de certificados digitais. O módulo de proveniência (em destaque na figura 2.6) será composto pelos componentes Provenance Recorder e Smart Rerun Manager. O componente Provenance Recorder é inserido num workflow, como os demais atores e diretores, e captura as informações sobre a execução do workflow. Os dados serão persistidos em arquivos texto. O componente Smart Rerun Manager, será responsável por permitir re-execuções parciais dos workflows. A persistência dos dados deve ser planejada durante a montagem do workflow. Pode-se inserir atores responsáveis pela persistência dos dados desejados em SGBDs ou em arquivos. Apesar de prover recursos para persistência dos resultados tais como arquivos locais, transmissão por FTP ou mesmo inserção em SGBDs, o Kepler não provê qualquer mecanismo para a identificação dos mesmos, usando por exemplo o LSID para este fim. O Kepler, assim como seu projeto original PtolomeuII, possui código-fonte aberto. Novos componentes de softwares podem ser criados de forma a extender as suas funcionalidades.

2.5.2 VISTRAILS

Desenvolvido pela Universidade de Utah, o Vistrails é um sistema de gerência de workflows científicos, com enfoque na visualização gráfica dos resultados. No Vistrails, as tarefas são implementadas por módulos - componentes de software - conectáveis ao ambiente. O suporte a serviços Web ocorre via módulo específico. O Vistrails é o único, dentre os projetos avaliados, a prover um mecanismo que registra as múltiplas versões dos workflows, documentando assim as suas evoluções. O Vistrails armazena, junto à definição do workflow, um registro das suas evoluções. Desta forma, é possível resgatar e executar todas as versões que um determinado workflow teve. Como os workflows científicos - devido a sua natureza experimental - evoluem constantemente, o registro das suas múltiplas versões

é de grande importância para o cientista. O Vistrails oferece ainda a possibilidade de identificar as diferenças entre dois workflows. A arquitetura do Vistrails é mostrada na figura 2.7.

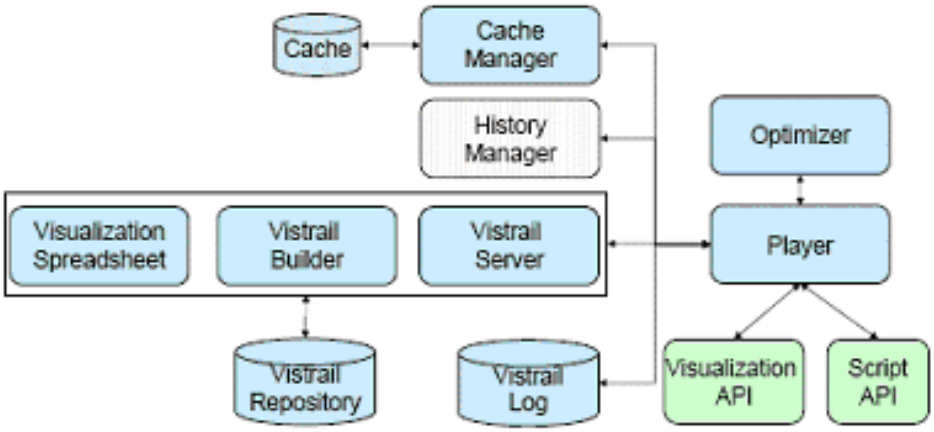

FIG. 2.7: Arquitetura do Vistrails (CALLAHAN e al, 2006)

O módulo Vistrail Builder faz a interface com o usuário, auxiliando-o na construção e modificação dos workflows. Uma vez construídos, os workflows podem ser salvos e resgatados localmente ou em repositórios na Web através do módulo Vistrail Server. O módulo Visualization Spreadsheet provê uma interface para visualização dos resultados. O Spreadsheet é dividido em células, onde cada célula pode conter o resultado de uma instância do workflow, facilitando assim análises comparativas dos resultados. Quando submetidos para execução, os workflows são analisados pelo Cache Manager que registra todas as tarefas envolvidas com suas respectivas parametrizações. Apesar da arquitetura prever que apenas as combinações de tarefas e parâmetros não encontradas em cache são enviadas para efetiva execução, a versão avaliada ainda não dispunha desta funcionalidade. O módulo Player gerencia as execuções das tarefas necessárias. O Player interage também com o módulo Optimizer que otimiza o workflow definido. O log da execução do workflow é capturado e armazenado pelo Vistrail Log. Até o momento da elaboração deste trabalho, a versão corrente do VisTrails, não apresentava um mecanismo formal para persistir os resultados em um SGBD. Os resultados podem ser armazenados em arquivos locais, através do componente FileSink, ou simplesmente salvar o resultado como um arquivo local. Na versão avaliada para elaboração deste trabalho, versão 1.0, o VisTrails não dispunha de mecanismos para identificação dos resultados. Desenvolvido

em Python, o VisTrails possui código-fonte aberto, e portanto, pode-se acrescentar novas funcionalidades, com o desenvolvimento de novos módulos. Os workflows construídos em VisTrails são definidos em linguagem própria, baseada em XML.

2.5.3 INSERVICES

O InServices é um sistema de gerência de workflows, desenvolvido pelo IME (Instituto Militar de Engenharia), especificamente para o domínio da Bioinformática. Dentre seus destaques cita-se a preocupação com a questão da filtragem dos dados e a adoção de padrões tecnológicos de mercado: linguagem BPEL [3] e serviços Web.

Para tratar a questão de filtragem de dados, o InServices se baseia em uma arquitetura voltada para a gerência de dados intermediários. Durante a execução do workflow, os dados produzidos por uma tarefa são utilizados como entrada para a tarefa seguinte, mas, freqüentemente, a tarefa seguinte não utiliza todo o conjunto de dados em seu processamento. Para filtrar estes dados, normalmente o bioinformata precisa tratar os dados intermediários manualmente. Assim, para evitar esta interferência, torna-se interessante a existência de filtros entre as tarefas. No entanto, a inserção destes filtros não é uma tarefa simples, pois a localização e configuração destes filtros podem variar conforme o experimento em questão. Desta forma, seria necessário que cada bioinformata adequasse o workflow para suas necessidades de filtragem. Para facilitar esta tarefa, o InServices oferece uma interface amigável, que se baseia na existência de workflows baseados em serviços Web.

No InServices cada tarefa é mapeada num serviço Web; a filtragem de dados se dá através da inserção de serviços Web (serviços de dados) no workflow previamente definido. Na arquitetura do InServices, os serviços de dados podem ser de três tipos: filtragem de dados, inserção e recuperação de dados num SGBD GUS. Além disso, os workflows passam por três etapas: etapa de definição (onde o workflow é efetivamente definido através da composição de serviços web, referentes aos programas científicos), etapa de redefinição (momento em que os serviços de dados são inseridos no workflow) e etapa de execução/re-execução (quando o workflow está disponível para ser re-executado várias vezes).

Cada tipo de serviço de dados segue um modelo específico: modelo de inserção, modelo de recuperação e modelo de filtragem. Como as informações sobre os dados que irão manipular só estarão disponíveis na etapa de redefinição do workflow, os modelos possuem

[3] http://www.oasis-open.org

apenas instruções básicas de como realizar as suas atividades. Assim, por exemplo, um modelo de filtragem possui apenas as operações básicas de comparação de valores, sem implementação.

Diferentemente dos demais projetos analisados nesse trabalho, o InServices é baseado na linguagem BPEL (Business Process Execution Language), que tornou-se um padrão adotado por grandes fornecedores - há engines BPEL fornecidos pela IBM, Oracle, BEA entre outros - permitindo que os workflows gerados sejam executados em diversas máquinas de execução. No projeto foi utilizado o Oracle BPEL Process Manager. A arquitetura do InServices, ilustrada na figura 2.8, é subdividida nos seguintes componentes:

- A - Gerência de Workflows;
  - A1 - Construção de Workflow;
  - A2 - Execução de Workflow;
- B - Serviços de Dados;
- C - Banco de Dados e Metadados;

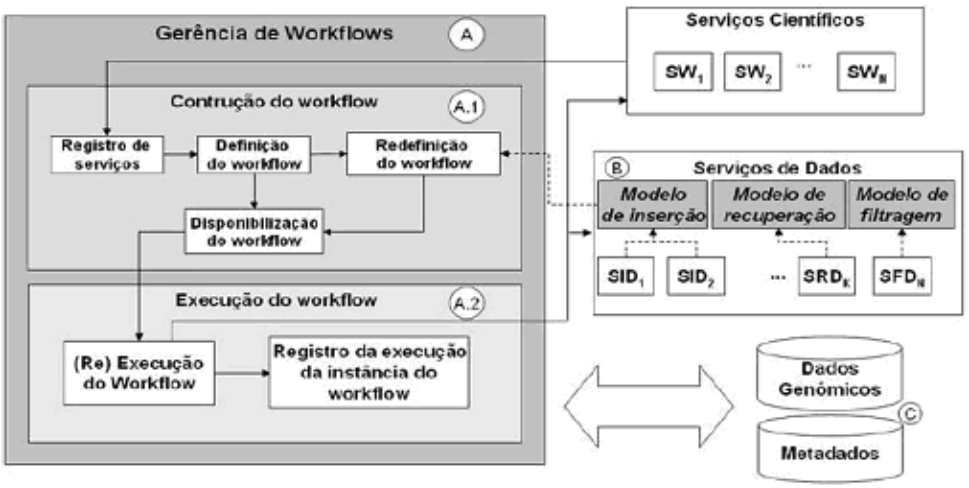

FIG. 2.8: Arquitetura do InServices.(SILVA, 2006)

O componente de Gerência de Workflows é responsável pela construção, execução e controle de execução dos workflows; é subdividido em dois subcomponentes: Construção de Workflows e Execução de Workflows. O componente de Construção de Workflows permite o registro de serviços Web científicos disponíveis, definição, redefinição e

disponibilização dos workflows científicos. Os workflows são construídos a partir dos serviços Web registrados. Na redefinição dos workflows construídos, são inseridos os serviços de dados (2.8.B) seguindo os respectivos modelos. No componente Execução de Workflow o workflow, já redefinido, é então interpretado pela máquina BPEL, seus módulos Re-execução do workflow e Registro de execução da instância do workflow são responsáveis, respectivamente, pela execução/re-execução do workflow definido e o registro dos dados da execução do workflow, assim como os seus passos, na base de metadados.

O InServices utiliza o GUS (Genomic Unified Schema) para persistir os dados. O GUS é um esquema extensível para bancos de dados relacionais associado a um framework de aplicação, concebido para armazenar, integrar, analisar e apresentar dados genômicos. O framework de aplicação provê uma camada objeto-relacional e uma API Plugin (desenvolvida em Perl) que facilita o desenvolvimento de aplicações para carga de dados. O GUS provê ainda o WDK (Web Development Kit), que permite a construção de páginas web para pesquisa de dados, necessitando de conhecimento mínimo de programação. Como foi desenvolvido utilizando tecnologias padronizadas e de código aberto, o InServices pode ser facilmente extendido para atender a novas necessidades. Embora disponível para reuso e/ou extensão através do contato com seu grupo de desenvolvimento, o protótipo do InServices cobre parcialmente as funcionalidades especificadas em sua arquitetura, oferecendo somente o serviço do tipo filtragem de dados.

2.5.4 TAVERNA

Concebido pelo grupo MyGrid, o Taverna é um sistema de gerência de workflows com foco em bioinformática. As tarefas podem estar implementadas via serviços Web assim como em componentes locais, também conhecidos como serviços locais. O Taverna usa a linguagem SCUFL (Simple Conceptual Unified Flow Language) para definição de seus workflows.

A figura 2.9 apresenta a arquitetura do Taverna; o componente Taverna Workflow Workbench provê a interface gráfica que permite a interação com o usuário, auxiliando nas tarefas de definição e execução de workflows. Os módulos responsáveis pelo armazenamento de dados e metadados persistem os resultados parciais e finais de cada execução em um SGBD. O componente MIR permite que os workflows sejam armazenados em um repositório central, ampliando a colaboração entre os cientistas. Apesar de mantido na arquitetura, o MIR teve seu desenvolvimento descontinuado. O componente

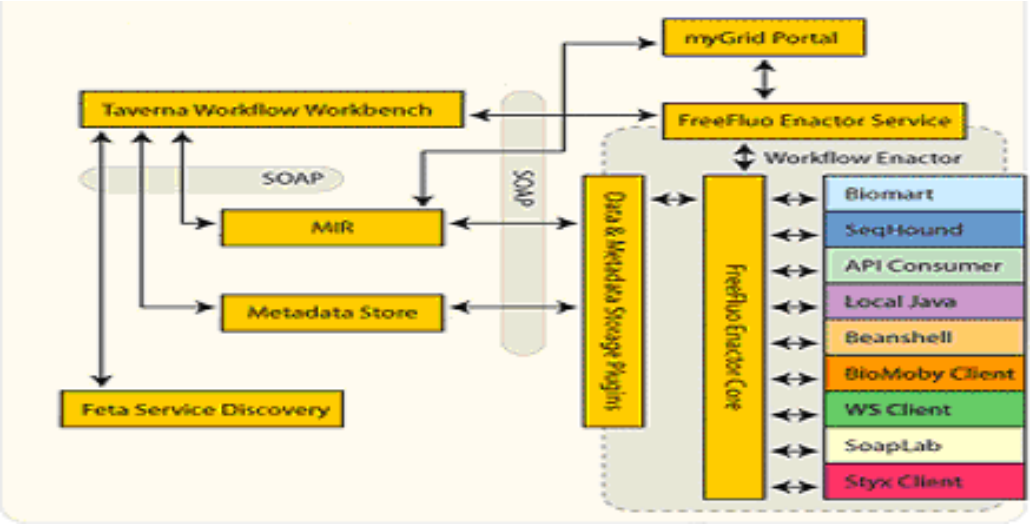

FIG. 2.9: Arquitetura do Taverna

Feta Service Discovery provê um mecanismo para busca de serviços através de suas descrições semânticas. O módulo responsável pela execução dos workflows é o FreeFluo Enactor. Inicialmente desenvolvido pela empresa IT Innovation, com o nome FreeFluo Workflow Enactor, tratava-se de um gerente de execução de workflows com capacidade de interagir com serviços Web. O FreeFluo tornou-se um projeto de código aberto e, atualmente, suporta duas linguagens de definição de workflows: uma linguagem própria baseada na linguagem IBM WSFL e a linguagem SCUFL. No Taverna, os resultados, finais e intermediários, obtidos das execuções dos workflows são identificados com uso de LSIDs. O Taverna tem embutido um gerador próprio de LSIDs, mas pode também ser configurado para interagir com provedores externos de LSIDs. O Taverna permite o armazenamento de seus resultados em arquivos locais ou em SGBD relacional. A versão disponibilizada pelo grupo MyGrid utiliza, por padrão, o SGBD MySQL. O Taverna dá suporte à proveniência de dados através do plugin LogBook (também conhecido como Taverna Provenance Plugin). Uma vez instalado, todos os registros necessários para a reprodução de uma execução de um workflow, bem como os resultados obtidos, são armazenados em dois bancos de dados MySQL: LogBook_Metadata e LogBook_Data.

2.6 CONSIDERAÇÕES FINAIS

Nesta seção os sistemas de gerência de workflows científicos serão analisados mediante os seguintes: finalidade, suporte a Serviços Web, granularidade de identificação dos

resultados, mecanismos de identificação dos resultados e meio de armazenamento dos resultados e paradigma de armazenamento dos resultados. Poderíamos acrescentar possibilidade de extensão, adesão a padrões código aberto, como características que tornam o sistema mais fácil de estender.

Dentre os sistemas analisados, o InServices e Taverna são sistemas de gerência de workflows com foco específico em bioinformática, enquanto o Vistrails e o Kepler são voltados para workflows científicos em geral.

Dada a heterogeneidade dos ambientes científicos, uma importante característica diz respeito ao suporte a Serviços Web. Todos os sistemas analisados provêem algum mecanismo para interagir com serviços Web. No caso do InServices e Taverna a interação é nativa, visto que nesses sistemas os serviços Web são diretamente acionados pelo gerente de execução. Nos casos do Kepler e Vistrails, a interação com Serviços Web se dá via componente específico que faz a interface entre a sua arquitetura e os Serviços Web.

A identificação dos resultados é de grande importância, visto que os mesmos devem ser recuperados para análise do cientista, ou reaproveitados em outras execuções, completas ou parciais, de um determinado workflow. Os sistemas InServices e Taverna apresentam mecanismos para identificação dos resultados intermediários e finais. No InServices, a identificação dos resultados segue um padrão proprietário. No Taverna, a identificação dos resultados se dá utilizando o mecanismo de LSIDs. Nos sistemas Kepler e VisTrails, não há um mecanismo formal para identificação dos resultados. Quanto ao armazenamento dos resultados, os sistemas Kepler e Vistrails não apresentam mecanismos estruturados de armazenamento dos dados em SGBDs. Dentre os sistemas analisados, apenas o InServices e o Kepler possuem um mecanismo estruturado para persistir os resultados em SGBDs. O InServices propõe a persistência dos resultados intermediários e finais no GUS (disponível atualmente para PostgreSQL e Oracle). O Taverna armazena os dados em SGBD relacional, MySQL por padrão.

Dentre os sistemas analisados, os que suportam armazenamento das informações em SGBDs - InServices e Taverna - as soluções de armazenamento dos resultados seguem o paradigma centralizado. Apesar de utilizarem a premissa da distribuição do processamento, através do uso de Serviços Web, os resultados produzidos por cada tarefa são persistidos num sítio central de armazenamento. Como a manipulação de grandes volumes de dados é característica em workflows científicos, o repetido tráfego de grandes massas de dados pode tornar o gerente central de armazenamento um ponto de

congestionamento para a execução dos workflows.

A tabela 2.2 resume os aspectos discutidos nessa seção. Num cenário de workflows com processamento distribuído, acredita-se que o paradigma de gerência distribuída de dados pode contribuir significativamente na redução do tempo total de execução dos workflows, atuando com a diretiva de reduzir os tempos de comunicação e, conseqüentemente, os tempos de transferência de dados entre os sítios de processamento. Assim, este trabalho pretende identificar oportunidades no ambiente de workflows científicos distribuídos, para aproximar os dados produzidos por cada tarefa, de seus sítios consumidores, e desta forma melhorar o desempenho destes workflows.

TAB. 2.2: Comparação entre os sistemas de gerência de workflows analisados

	Kepler	Vistrails	InServices	Taverna
Domínio de Aplicação	Científico em geral	Científico em geral	Específico na Bioinformática	Científico com foco em Bioinformática
Suporte a Serviços Web	Via Ator	Via módulo	Nativo	Nativo
Identificação dos resultados	Não possui	Não possui	Intermediários e Finais	Intermediários e Finais
Mecanismo de identificação dos resultados	Não possui	Não possui	Proprietário	LSID
Armazenamento dos dados	Arquivo local ou SGBD	Arquivo local	SGBD (esquema GUS)	Arquivo local ou SGBD
Armazenamento distribuído dos dados intermediários	Não possui	Não possui	Não possui	Não possui

3 UMA ABORDAGEM PARA DISTRIBUIÇÃO DE DADOS EM WORKFLOWS DE BIOINFORMÁTICA

O uso da gerência distribuída de dados em workflows científicos que manipulam grandes volumes de dados pode contribuir para redução do tempo total de execução dos workflows. No entanto, para projetar a distribuição dos dados ao longo da execução de um conjunto de workflows, e tirar proveito desta distribuição nos SGWf existentes, é necessário adaptar as metodologias e técnicas de distribuição de dados existentes em SGBD distribuídos para o contexto dos workflows, de acordo com uma arquitetura adequada para a gestão dos dados distribuídos em SGWf, que inclua a identificação, armazenamento e reuso.

Este trabalho endereça especificamente os workflows de Bioinformática, onde o domínio de conhecimento científico é relacionado à biologia. Os workflows de bioinfomática são empregados em experimentos *in silico* envolvendo, principalmente, pesquisas em genômica ou proteômica. Apesar de já haver sistemas de gerência de workflows científicos específicos para bioinformática, muitos cientistas ainda realizam seus experimentos através de workflows implementados como scripts em Perl. No entanto, esta abordagem normalmente não atende aos requisitos dos cientistas desta área, como por exemplo, a proveniência de dados e a possibilidade de realizar re-execuções parciais.

Neste sentido, este capítulo apresenta uma abordagem para a extensão da funcionalidade de sistemas de gerência de workflow científico, de forma a viabilizar a gerência da distribuição de dados em workflows de bioinformática.

O capítulo está organizado como detalhado a seguir. Na seção 3.1 é apresentada uma visão geral da proposta, na seção 3.2 é apresentada a arquitetura da proposta. A seção 3.3 apresenta um metamodelo proposto para apoiar a tomada de decisão de alocação dos dados intermediários. Na seção 3.4 é apresentado um modelo de custos para identificar, de acordo com o metamodelo e parâmetros de ambiente, qual é o melhor local de armazenamento dos dados intermediários entre as tarefas de um workflow executado.

3.1 VISÃO GERAL DA PROPOSTA

Conforme análise dos principais sistemas de gerência de workflows científicos, realizada no capítulo 2, observa-se em suas implementações que quando há suporte a persistência dos resultados em SGBDs, este suporte se dá seguindo o paradigma centralizado.

Neste trabalho propõe-se uma abordagem cuja arquitetura dê suporte ao paradigma de gerência distribuída de dados intermediários, interagindo com os principais sistemas de gerência de workflows existentes. A abordagem proposta pressupõe a segregação dos workflows em abstratos e executáveis. Os workflows abstratos possuem apenas a arrumação lógica das tarefas, não podem ser executados. Os workflows executáveis possuem todas as instruções necessárias para a execução distribuída dos programas que compõem cada workflow, e no contexto deste trabalho, para dar suporte à gerência distribuída de dados. A arquitetura proposta para suportar a execução com gerenciamento de dados distribuído é ilustrada na figura 3.1.

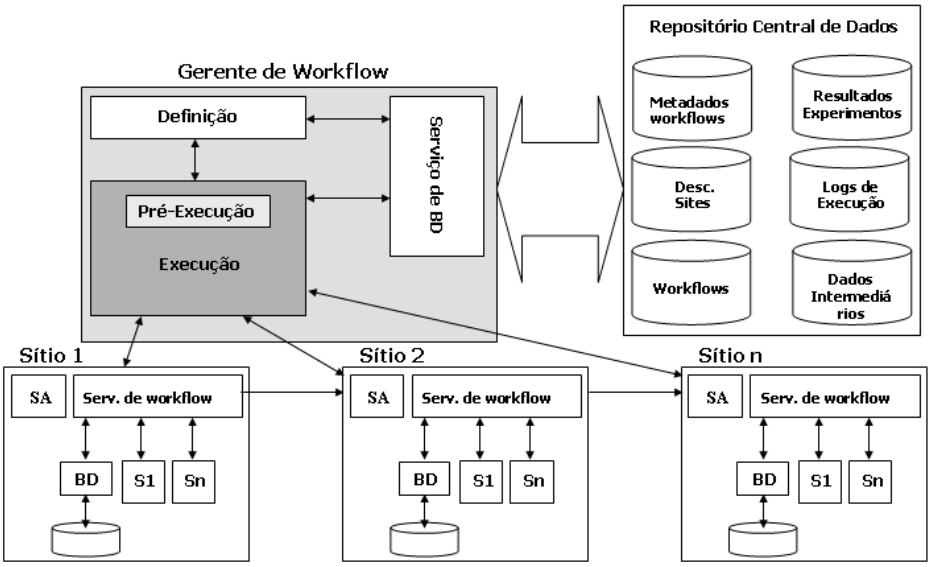

FIG. 3.1: Arquitetura proposta do D-BioFlow

A arquitetura proposta prevê a existência de um sítio central, denominado **Gerente de Workflow**, constituído por quatro módulos: **Serviço de BD**, **Definição**, **Pré-Execução** e **Execução**. Os módulos de **Definição** e **Execução** estão normalmente presentes na arquitetura de sistemas de gerência de workflows (SGWFs), e permitem que os usuários possam descrever seus workflows e mais tarde executá-los. Na abordagem proposta aqui, foi acrescentado um módulo de **Serviços de Dados**, um módulo de

Pré-Execução, e um **Repositório Central de Dados** que dá suporte aos módulos de **Serviço de Dados** e **Pré-Execução**. O módulo **Serviço de BD** é responsável por coletar e prover informações para permitir que o módulo de Pré-Execução realize a redefinição dos workflows abstratos, transformando-os em workflows executáveis capazes de tratar o armazenamento dos dados intermediários. É o módulo de Pré-Execução que irá decidir pelos locais de armazenamento mais adequados. Por fim, o Repositório Central de Dados mantém informações úteis para o Gerente de Workflows de modo geral, mas em especial este repositório inclui metadados que descrevem os workflows abstratos e executáveis, e os servidores que hospedam os serviços que implementam as tarefas e/ou que estão disponíveis para armazenamento de dados, e os dados intermediários que estão persistidos a cada momento. O esquema que especifica estes metadados está descrito na seção 3.2 mais adiante.

A arquitetura prevê também que em cada sítio estejam presentes três módulos, além dos serviços lá hospedados: **SA**, **Serviço de Workflow** e um **Gerente Local de BD**. O módulo **SA** (Serviço de Administração), provê informações administrativas - como disponibilidade, espaço disponível, permissionamento, etc.- ao **Gerente de Workflow**. O módulo **BD** (Gerente Local de BD), é responsável pela persistência e recuperação local da informação, quando o sítio em questão for de armazenamento. O **Serviço de Workflow** realiza as interações necessárias do gerente central com os serviços de processamento do sítio, serviço de **BD** e demais sítios de processamento.

3.2 METAMODELO

Para dar suporte à execução distribuída dos workflows, a arquitetura proposta necessita armazenar diversas informações do ambiente e das execuções dos workflows, além dos dados intermediários a fim de permitir seu reuso em re-execuções futuras de workflows e em consultas a dados de proveniência. Para atender a essa necessidade, foi proposto o metamodelo de dados mostrado na figura 3.2, que pode ser dividido em duas partes com objetivos distintos: armazenamento de informações no repositório central de dados e armazenamento de dados no sítio de armazenamento. A parte do metamodelo responsável pelos dados no repositório central de dados é mostrado na figura 3.3. A classe *Workflow* representa os workflows abstratos, a classe *InstânciaWorkflow* modela os workflows executáveis. Os workflows abstratos e executáveis estão relacionados a uma classe *Usuário*, que representa os usuários do sistema (podendo ser dos tipos cientista

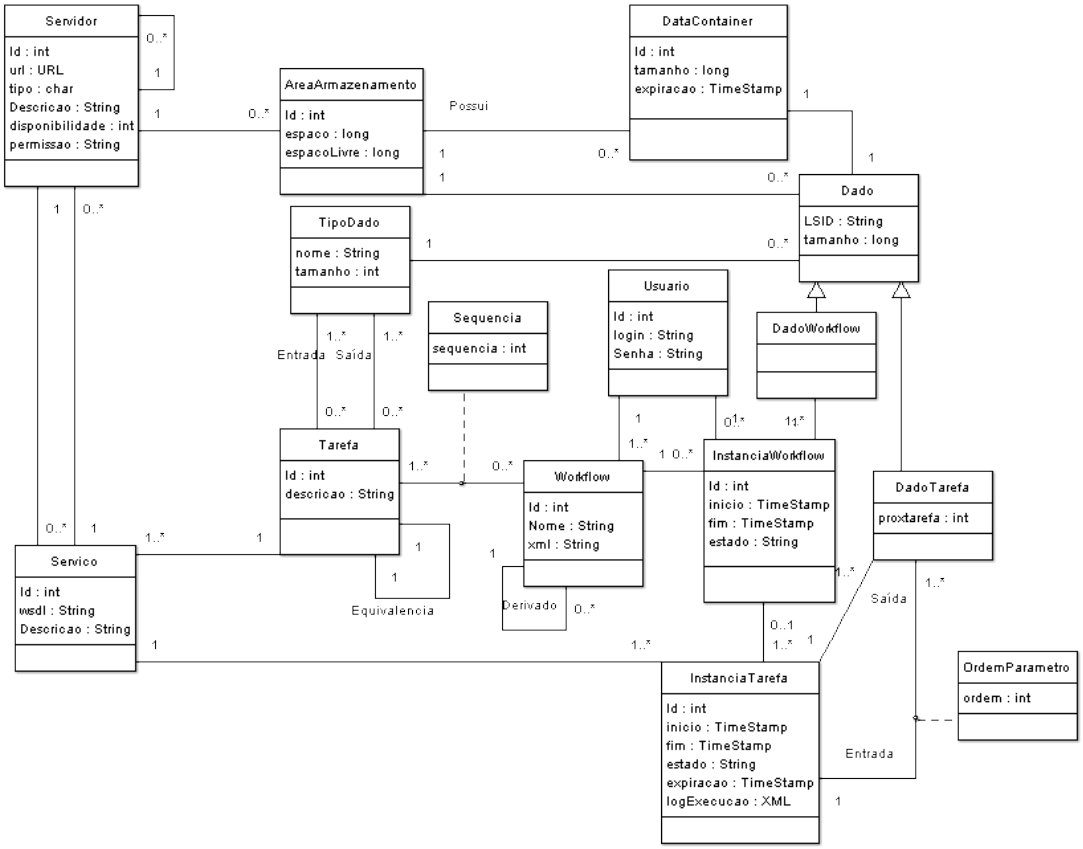

FIG. 3.2: Metamodelo de Dados

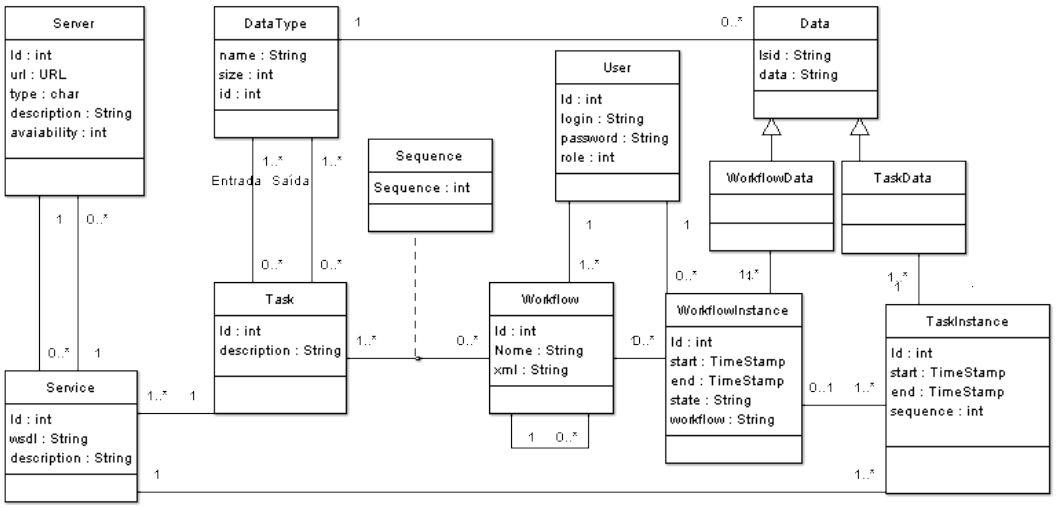

FIG. 3.3: Metamodelo, repositório central de dados

ou administrador). O usuário cientista registra workflows abstratos e realiza execuções dos workflows. A classe *Tarefa* modela os algoritmos que podem ser escolhidos para a construção dos workflows abstratos. A classe *Tarefa* relaciona-se com a classe *TipoDado*, que por sua vez relaciona-se com a classe *Dado*; estas classes mapeiam os tipos de dados utilizados nos parâmetros de entrada e saída utilizados pelos algoritmos. A classe *Tarefa*, relaciona-se também com uma classe *Serviço*, que por sua vez está relacionada a uma classe *Servidor*. Estas classes modelam os sítios remotos, onde uma *Tarefa* pode possuir vários serviços Web (classe *Serviço*) que a implementam. Os serviços Web podem estar hospedados em vários sítios de processamento (classe Servidor). A classe *Servidor* possui o atributo tipo para identificar se uma instância de servidor é do tipo processamento (que hospeda somente serviços), armazenamento (que hospeda somente dados) ou ambos.

Um workflow executável (instância da classe *InstânciaWorkflow*) representa um workflow que pode ser executado. Note que um workflow executável pode representar um workflow que foi redefinido (processo de pré-execução) mas que ainda não iniciou a sua execução, um workflow que esteja sendo executado ou, ainda, um workflow que tenha sua execução concluída. Os workflows executáveis, modelados pela classe *InstanciaWorkflow*, ao serem executados instanciarão os serviços Web que implementam as tarefas. Estas instâncias de serviços Web são modeladas pela classe *InstânciaTarefa*. Num workflow, seja abstrato ou executável, as tarefas ou instâncias de tarefa, devem seguir as regras de encadeamento especificadas. Neste encadeamento, os dados produzidos por uma tarefa são consumidos pela tarefa seguinte. Essa propriedade é modelada pelo relacionamento entre as classes *InstânciaTarefa* e *DadoTarefa*. Como uma tarefa pode receber mais de um parâmetro de entrada, a classe *OrdemParametro* contém a ordem em que os referidos dados são passados como parâmetro. Um workflow produz também um resultado final, representado pela classe *DadoWorkflow*. As classes *DadoWorkflow* e *DadoTarefa* são especializações da classe *Dado*, que modela todos os dados produzidos pelas execuções dos workflows.

A parte do metamodelo de dados que representa as informações hospedadas nos sítios de armazenamento de dados é ilustrada na figura 3.4. As classes: *Servidor*, *Serviço*, *Dado* e *DadoTarefa* apresentam no contexto do sítio remoto a mesma modelagem explicada no metamodelo de dados do repositório central. No contexto do sítio de armazenamento, o metamodelo insere as classes *AreaArmazenamento* e *DataContainer*, que serão detalhadas a seguir.

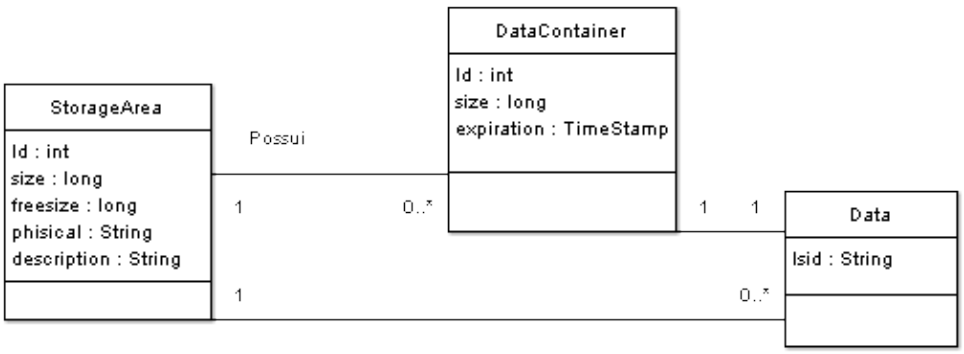

FIG. 3.4: Metamodelo de dados, sítios remotos

A classe *AreaArmazenamento* modela as possíveis áreas de armazenamento disponíveis nos sítios remotos. Quando uma instância da classe *Servidor* for do tipo armazenamento (ou do tipo processamento e armazenamento), a instância disporá de áreas de armazenamento para armazenar localmente os dados. A classe *DataContainer* modela o recurso de reserva prévia de espaço para o dado que ainda será produzido. Uma vez produzido o dado, a reserva prévia de espaço é excluída, liberando o recurso alocado para receber a informação definitiva. As instâncias de *DataContainer* possuem uma etiqueta de tempo para a expiração. Esse recurso garante que, se por qualquer motivo o dado não seja produzido no limite de tempo estabelecido, a reserva prévia de espaço liberará o espaço alocado para outros processos.

3.3 DETALHAMENTO DA ARQUITETURA D-BIOFLOW

Nesta seção serão detalhados os módulos apresentados na seção 3.1. As subseções 3.3.1, 3.3.2 e 3.3.3 abordam os módulos do Gerente de Workflows, as seções 3.3.4, 3.3.5 e 3.3.6 tratam dos módulos dos sítios remotos.

3.3.1 MÓDULO DE SERVIÇO DE BD

Este módulo é responsável pela alimentação do **Repositório Central de Dados**. Quando forem inseridos/modificados dados referentes às tarefas, servidores e serviços Web, **o serviço de BD** atualizará o **Repositório Central de Dados**, nas suas áreas de metadados de workflows e descritores dos sites. Quando são registrados workflows abstratos, bem como quando são solicitadas execuções de workflows, o **serviço de BD** interage com a base de workflows do **Repositório Central de Dados**. Quando a

execução de um workflow é finalizada, os dados finais resultantes do processamento, bem como o log de execução são armazenados nas bases de resultados dos experimentos e logs de execução, respectivamente. Em determinadas janelas de tempo, um processo batch do gerente de workflows copia os dados intermediários, produzidos pelas execuções finalizadas dos workflows, para a base de dados intermediários do repositório central de dados.

3.3.2 MÓDULO DE DEFINIÇÃO

O módulo de **Definição** apóia o cientista na definição dos workflows, isto é, na definição das tarefas que os compõem. O módulo de **Definição**, com base em informações trazidas do **Repositório Central de Dados** pelo módulo **Serviço de BD**, provê o usuário com uma descrição de todas as tarefas catalogadas no **Repositório Central de Dados**. Desta forma o cientista constrói o workflow abstrato, com auxílio de uma ferramenta de construção de workflows. O workflow definido será constituído apenas pela organização lógica das tarefas, sem a implementação efetiva das tarefas, isto é, sem a associação das mesmas com os serviços que as implementam. Este workflow é, portanto, abstrato, e não pode ser executado. Finalizada a construção do workflow abstrato, o cientista deve registrá-lo no repositório central, e para isso conta com o módulo **Serviço de BD**.

3.3.3 MÓDULO DE EXECUÇÃO

Uma vez registrado no repositório central, um workflow abstrato fica disponível para ser executado. O cientista pode então selecionar o workflow abstrato, dentre os disponíveis, e solicitar uma execução do mesmo. A execução pode ser completa, quando o cientista fornecer o conjunto de dados de entrada, ou parcial (quando um workflow é executado a partir de determinada tarefa, reaproveitando os dados produzidos em execuções anteriores). Quando a execução de um workflow abstrato é solicitada, o módulo de Pré-Execução é acionado. O módulo de Pré-Execução redefine o workflow abstrato (selecionado para execução) inserindo os serviços Web reais que implementam as tarefas definidas, no momento de construção do workflow, e insere também serviços Web necessários para o gerenciamento distribuído dos dados intermediários, bem como serviços para monitorar a execução distribuída do workflow. Assim, o módulo de Pré-Execução redefine o workflow abstrato produzindo um workflow executável. Gerado o workflow executável, o módulo de Execução gerencia a execução do workflow, interagindo com o sistema de gerência de workflow apropriado.

As decisões tomadas pelo módulo de Pré-Execução quanto à inserção de serviços para alocação dos dados intermediários são tomadas mediante a combinação dos dados do repositório central de dados (descritos segundo o metamodelo detalhado na seção 3.2), com os resultados calculados por um modelo de custos (detalhado na seção 3.4).

3.3.4 SÍTIO REMOTO - MÓDULO SA

Na arquitetura proposta, todos os sítios remotos são previamente cadastrados no repositório central de dados, onde são catalogadas informações quanto aos serviços Web que hospedam e quanto a sua finalidade, isto é, se é um sítio de processamento, onde apenas se hospedam serviços Web, ou se é um sítio exclusivo para armazenamento de dados, ou ainda, se é um sítio híbrido, que atende ambas as finalidades. Porém estas informações precisam ser atualizadas de tempos em tempos.

O módulo **SA** (Serviço de Administração), provê informações administrativas - como disponibilidade, espaço disponível, permissionamento, etc.- ao **Gerente de Workflow**. Na etapa inicial do pré-processamento, o **Gerente de Workflow** interage com o módulo SA dos servidores de armazenamento, identificando quais servidores de armazenamento do ambiente possuem espaço suficiente para o dado. Para estimar o tamanho do dado, o módulo de pré-execução utiliza o metamodelo (seção 3.2, classe *DataType*). Neste momento também são verificados os aspectos de disponibilidade e permissão de acesso. Identificado o subconjunto de servidores candidatos, o módulo de pré-execução calcula qual servidor representará a melhor alternativa de armazenamento. O gerente de workflows requisita a reserva de espaço (instancia um *DataContainer*, conforme descrito na seção 3.2) ao módulo **SA** do servidor escolhido e continua o processamento para as tarefas seguintes.

3.3.5 SÍTIO REMOTO - MÓDULO BD

O módulo **BD**, é responsável pela persistência e recuperação local da informação, quando o sítio em questão for de armazenamento. Durante a execução de um workflow, ao término do processamento de cada tarefa, os dados intermediários são encaminhados para o armazenamento. Quando o cientista solicita uma execução parcial, parte dos dados de uma execução antiga do workflow em questão são reaproveitados. Desta forma, os dados necessários são requisitados, para recuperação, aos respectivos servidores para iniciar a execução do workflow a partir da tarefa solicitada pelo cientista.

3.3.6 SÍTIO REMOTO - MÓDULO SERVIÇO DE WORKFLOW

O **Serviço de Workflow** realiza as interações necessárias entre o gerente central e os serviços de processamento do sítio, serviço de BD e demais sítios de processamento. O **Serviço de Workflow** provê a interface para as requisições de armazenamento e recuperação de dados, interagindo com o módulo de BD. Quando um dado é persistido (através da chamada ao serviço Web de armazenamento de dados do sítio remoto), o **Serviço de Workflow** comunica ao gerente de workflow o LSID correspondente. Quando solicitada a recuperação do dado, para um serviço de processamento local ou remoto, o serviço de workflow interage com o serviço local de BD, de forma a enviar o dado ao serviço requerente.

3.4 MODELO DE CUSTO DE EXECUÇÃO DE WORKFLOWS EM AMBIENTES DISTRIBUÍDOS

Conforme discutido previamente, a distribuição dos dados intermediários de um workflow envolve um Problema de Alocação de Dados. No capítulo 2 foi mostrado que, no domínio de Sistemas de Bancos de Dados Distribuídos (SBDD), diversos aspectos são considerados na definição de um projeto de alocação de dados. No domínio de Workflows Distribuídos, podem-se acrescentar novos aspectos para apoiar a construção de um projeto de alocação para os dados intermediários dos workflows. Essa seção propõe um modelo de custo que considera aspectos dos domínios de SBDD e de workflows distribuídos, para estimar o custo de execução de um workflow em um ambiente distribuído.

O modelo de custo proposto será aplicado para orientar o projeto de alocação dos dados intermediários gerados por cada serviço durante a execução de um workflow, visando reduzir o tempo total de execução dos workflows através da redução dos custos de transferência de dados entre os serviços. O modelo de custos proposto considera diversos parâmetros, agrupados em:

- **Rede:** Constitui o custo de transferência de uma unidade de dado entre os sítios da rede. É representado por uma matriz CTR, onde $ctr_{i,j}$ é o custo de transferir um dado do sítio s_i para o sítio s_j. Para simplificar o problema, CTR é uma matriz simétrica onde $ctr_{i,i}$ é igual a 0;

- **Armazenamento:** Constitui o custo envolvido para armazenar o dado em um

determinado sítio de armazenamento. É representado por um vetor CARM, onde $CARM[i]$ representa o custo para se armazenar o dado no sítio i;

- **Leitura:** Custo envolvido na leitura do dado que esteja armazenado em um determinado sítio de armazenamento. É representado por um vetor CREC, onde $CREC[i]$ representa o custo para se recuperar o dado que está armazenado no sítio i;

Considerando estes parâmetros, define-se uma função de custo a partir da qual pode-se estimar o custo para se armazenar um dado d nos sítios de armazenamento disponíveis. Considere um workflow constituído pelo conjunto encadeado de tarefas $T = t_1, t_2, t_3, ..., t_m$, onde t_{i+1} representa a tarefa sucessora de t_i, $0 < i < m$. O ambiente de execução é constituído por um conjunto de servidores remotos $S = s_1, s_2, s_3, ..., s_n$, onde cada s_i pode ser um servidor de processamento, armazenamento ou ambos. Cada servidor de processamento s, $s \in S$, hospeda um conjunto definido de serviços que implementam um subconjunto das tarefas de T. Define-se a função *Srv(t)*, que retorna um conjunto S' ($S' \subseteq S$) de servidores remotos que hospedam os serviços que implementam a tarefa t.

Considere $K = k_1, k_2, k_3, ..., k_s$, o conjunto de todos os servidores de armazenamento do ambiente com espaço disponível para armazenar o dado d. Para cada tarefa $t_i \in T$. A função de custos deve calcular os custos envolvidos para armazenar o dado d no servidor k_y, $1 \leq y \leq s$ e retornar k_y que represente o custo mínimo. Assim, a função de custos é definida conforme na equação 3.1:

$\forall x \in S',\ onde\ \ S' = Srv(t_i),$

$\forall w \in W\ ,\ onde\ \ W = Srv(t_{i+1}),$

$(Ctr(x, k_y, d) + Ctr(k_y, w, d) + Carm(k_y, d) + Crec(k_y, d))$ (Equação 3.1)

A função $Ctr(s_i, s_{i+1}, d)$ representa o custo de transferência do dado d, em pacotes de dados, do servidor s_i para o servidor s_{i+1} é definida como:

$$Ctr(s_i, s_{i+1}, d) = \frac{d}{PCT\_DATA} \times CTR[i][i+1]$$

Onde:

PCT_DATA = Tamanho do pacote padrão de dados na rede.

As funções $Carm(k_y, d)$ e $Crec(k_y, d)$ representam os custos de armazenamento e recuperação do dado d no servidor k_y. Desta forma, as funções $Carm(k_y, d)$ e $Crec(k_y, d)$ são definidas como a seguir.

$$Carm(k_y, d) = \frac{d}{PCT\_DATA} \times CARM[k_y]$$

$$Crec(k_y, d) = \frac{d}{PCT\_DATA} \times CREC[k_y]$$

O modelo de custos apresentado nesse trabalho considera que o tempo total de execução de um workflow é composto pela soma dos tempos de execução de cada tarefa mais o tempo necessário para transferir os dados entre cada par de tarefas subseqüentes, considerando o custo de persistência dos dados intermediários. Desta forma, o tempo de execução de um workflow pode ser representado como uma função crescente.

A alocação de dados intermediários de um workflow pode ser encarada como um projeto de alocação de dados. Neste caso, utilizando a função de custos definida na equação 3.1, será encontrado um plano de alocação de dados intermediários que represente a melhor alternativa de armazenamento, conforme os critérios discutidos anteriormente. Porém, num ambiente com grande número de servidores, o número de possibilidades a serem analisadas para escolha do plano de alocação de dados pode ser substancialmente grande. Neste cenário, a função definida na equação 3.1 analisará todas as combinações possíveis de servidores de processamento e armazenamento, o que pode demandar um custo computacional inviável.

Em cenários complexos, portanto,uma alternativa para reduzir o custo computacional envolvido na construção de planos de alocação de dados seria restringir o espaço de busca. Essa alternativa levará à construção de planos de alocação de dados satisfatórios, porém não ótimos, num tempo computacional viável. O espaço de busca pode ser restringido através da inserção de heurísticas no modelo de custos utilizado.

No contexto deste trabalho, workflows de bioinformática, é característico a repetição freqüente de combinações de tarefas, bem como a existência de workflows derivados, conforme discutido no capítulo 2. Um workflow pode possuir vários níveis de derivação. Considere, por exemplo, um workflow w1 composto pelo encadeamento das tarefas $T1 -> T2 -> T3 -> T4$. Define-se um workflow w2 constituído pelo encadeamento de tarefas $T1 -> T2 -> T3 -> T5 -> T6$, derivado do workflow w1. Pode-se definir um workflow w3, constituído pelas tarefas $T1 -> T2 -> T3 -> T5 -> T6 -> T7$, que é derivado de w2. Como o workflow w2 é derivado de w1, então w3 também será derivado do workflow w1. Desta forma, uma execução do workflow w3 pode aproveitar dados de execuções anteriores dos workflows w1 e w2.

Desta forma, escolheu-se identificar a existência de workflows derivados e, dentre os workflows derivados, a associação freqüente de pares de tarefas. Pode-se ainda considerar a freqüência de execução dos workflows, induzindo o modelo a aproximar os dados dos servidores que hospedam as prováveis tarefas consumidoras. Essas alternativas podem levar a uma decisão ruim para a execução corrente, mas objetiva-se obter significativa melhora global.

Assim, no modelo de custos proposto, passa-se a inserir a função $Consumer(t_i, n)$ que retorna, quando houver, as n tarefas que foram definidas com maior freqüência como consumidoras de t_i, ordenadamente. A função de custos ficará então definida como:

$\forall x \in S',\ onde\ S' = Srv(t_i),$

$\forall h \in H,\ onde\ H = Consumer(t_i, n),$

$\forall z \in Z,\ onde\ Z = Srv(h),$

$Min(Ctr(x, k_y, d) + Ctr(k_y, z, d) + Carm(k_y, d) + Crec(k_y, d))$ (Equação 3.2)

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi proposta uma abordagem para distribuição de dados em workflows de bioinformática. A abordagem contempla de forma adequada a semântica distinta entre workflows abstratos e executáveis. A separação entre tais conceitos permitiu executar atividades de projeto de distribuição dos dados intermediários de um workflow em tempo de pré-execução, a partir da especificação do workflow abstrato já existente. Da mesma forma, a análise da seqüência de tarefas previstas para um workflow abstrato permite identificar oportunidades de reuso de dados intermediários persistentes, contribuindo para a redução do tempo de execução dos workflows através da re-execução parcial.

A arquitetura apresentada insere um módulo de pré-execução, responsável por redefinir os workflows abstratos em workflows executáveis que contenham a especificação do servidor onde os dados intermediários de saída de cada tarefa devem ser armazenados. Deve-se destacar que a abordagem proposta é genérica, podendo ser integrada aos principais sistemas de gerência de workflows científicos existentes. Para suportar a execução distribuída, foi apresentado um metamodelo de dados que descreve informações de apoio tanto para o gerente central de workflows quanto para os servidores remotos de dados. Por último, foi apresentado um modelo de custos para workflows científicos distribuídos, bem como algumas características dos workflows de bioinformática que podem ser aproveitadas na forma de heurísticas para restringir o espaço de busca para

obtenção de um plano de alocação de dados intermediários satisfatório.

4 PROTÓTIPO E-BIOFLOW

No capítulo 3 foi apresentada uma abordagem para gerência da distribuição de dados em workflows de bioinformática, com apresentação de uma arquitetura, um metamodelo de dados e uma função de custos. Baseado nessa proposta foi desenvolvido o protótipo **e-BioFlow**, que além de possibilitar a execução de workflows com suporte a gerência distribuída dos dados intermediários, simplifica os processos de definição e execução - completa ou parcial - de workflows.

O **e-BioFlow** foi concebido para poder funcionar integrado a sistemas de gerência de workflow que suportem tarefas implementadas como serviços Web e cujas máquinas de execução suportem chamadas de execução de workflows em background. No estágio atual deste trabalho, o **e-BioFlow** funciona integrado apenas ao Taverna. O **e-BioFlow** foi concebido, prevendo a segregação de usuários por perfil de trabalho: administrador e cientista. O usuário administrador é responsável pela manutenção dos cadastros de base, tais como: usuários, tarefas, servidores e serviços Web. O usuário cientista é aquele que cria, registra e executa os workflows.

Este capítulo descreve o **e-BioFlow**, seu desenvolvimento, seu modo de funcionamento e suas funcionalidades para prover o gerenciamento distribuído de dados dos workflows. Na seção 4.1 é apresentada a arquitetura do **e-BioFlow**, bem como as tecnologias utilizadas no seu desenvolvimento. Na seção 4.2 é apresentado o funcionamento do **e-BioFlow**. Na seção 4.3 são apresentadas as interfaces do **e-BioFlow**.

4.1 IMPLEMENTAÇÃO

Construído utilizando tecnologias abertas e componentes gratuitos, o **e-BioFlow** foi desenvolvido em Java versão 1.6 [4]. O servidor de aplicações utilizado foi Apache Tomcat versão 5.5.23 [5], para gerenciar o funcionamento dos serviços Web foi utilizado o Apache Axis versão 1.4 [6]. Para a camada de persistência foi utilizado o SGBD MySQL versão

[4] http://www.javasoft.com
[5] http://tomcat.apache.org
[6] http://ws.apache.org/axis/

FIG. 4.1: Arquitetura de componentes

5.0.41 [7]. A arquitetura de componentes é ilustrada na figura 4.1.

O **e-BioFlow** foi construído utilizando o padrão MVC (Model View Controller). A camada de visão foi desenvolvida em JSP (Java Server Pages), a camada de controle foi implementada através de um Servlet que recebe as requisições do navegador do cliente e realiza chamadas às classes de modelo. Para interação com a camada de persistência (SGBD MySQL) foram utilizados padrões DAO (Data Access Object) e Abstract Factory (GAMMA, 1994).

Para implementação inicial do e-BioFlow, foi escolhido o sistema de gerência de workflows Taverna, versão 1.6.2. A escolha deveu-se ao fato de o Taverna apresentar alguns recursos que facilitaram a sua integração ao **e-BioFlow** e gerenciamento de execução dos workflows. Tais recursos são: possibilidade de execução de workflows em modo linha de comando (shell) e geração de relatório de execução de workflows. A possibilidade de realizar execuções de workflows por de linha de comando é uma característica fundamental para o funcionamento do **e-BioFlow**, visto que é dessa forma que o mesmo interage com a máquina de execução; o relatório de execução de workflows é utilizado para captura de informações da execução, dentre as quais pode-se destacar o tempo de execução de cada tarefa e o estado final da execução do workflow.

O **e-BioFlow** pode ser estendido para funcionar junto a outros sistemas de gerência de workflows que possuam os mesmos recursos supracitados. Para isso devem ser

[7]http://www.mysql.com

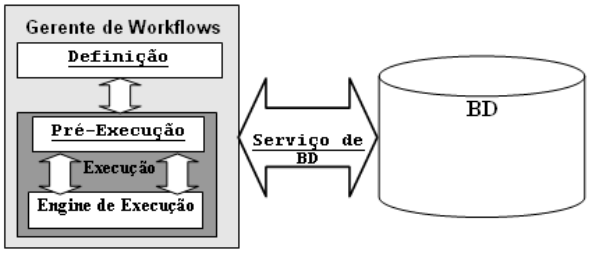

FIG. 4.2: Arquitetura funcional e-BioFlow

```
public class Services
{
     public  String concat( String input1 , String input2 )
     {
       String saida_1 = null;
       return saida_1;
     }

     public  String toupper( String input )
     {
       String saida_1 = null;
       return saida_1;
     }
}
```

FIG. 4.3: Exemplo de conteúdo do serviço Web Services

implementados os componentes de código necessários para redefinir os workflows de acordo com a nova linguagem de definição.

4.2 FUNCIONAMENTO DO E-BIOFLOW

A arquitetura funcional (implementada) do protótipo **e-BioFlow**, é apresentada na figura 4.2. A seguir ilustramos o funcionamento desta arquitetura, descrevendo em mais detalhes como os módulos interagem.

O módulo de definição apóia o cientista no processo de definição do workflow abstrato. Periodicamente, o **e-BioFlow** realiza a leitura de todas as tarefas, que foram previamente cadastradas pelo usuário administrador, no repositório central de dados (repositório de metadados de workflows, vide figura 3.1 do capítulo 3) e constrói um grande serviço Web, denominado Services, contendo uma representação vazia das tarefas. O serviço Web *Services* conterá uma função para cada tarefa registrada. Uma vez gerada, a nova versão de *Services* é automaticamente publicada, neste momento o componente Axis (vide figura 4.1) gera um novo arquivo WSDL, contendo uma operação para cada tarefa. Um exemplo do conteúdo do serviço Web *Services* é mostrado na figura 4.3.

No exemplo da figura 4.3, o repositório central de dados possui apenas duas tarefas

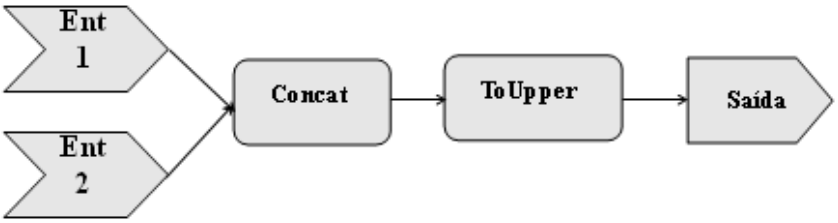

FIG. 4.4: Workflow Abstrato W_{a1}

cadastradas: concat e toupper. Ambas são representadas através de funções com diretiva pública de acesso. O conteúdo de cada função possui apenas a inicialização de uma variável e comando de retorno. A função deve retornar um dado do tipo definido como resultado da tarefa no repositório central de dados.

O cientista constrói um workflow abstrato com auxílio de uma ferramenta de construção de workflows, como o editor gráfico do Taverna por exemplo. Na construção do workflow abstrato, o cientista encadeia as tarefas apontando-as para o serviço Web Services. Ao final, todas as tarefas do workflow abstrato apontarão para o mesmo serviço Web, diferenciando-se apenas pela operação escolhida.

Considere como exemplo um workflow W_{a1} composto por duas portas de entrada de dados, uma porta de saída para o resultado final e 2 tarefas, denominadas concat e toupper, respectivamente. Um esquema do workflow abstrato W_{a1} é ilustrado na figura 4.4.

Um trecho do workflow abstrato W_{a1}, definido na linguagem SCUFL, é ilustrado na figura 4.5.

O módulo de execução é dividido em dois submódulos: Pré-Execução e Engine de Execução. O módulo de Pré-Execução recebe um workflow abstrato como parâmetro e retorna um workflow executável. A redefinição - transformação - de um workflow abstrato em um workflow executável, consiste na substituição dos serviços Web, de abstratos para físicos, bem como a inserção de novos serviços Web para gerenciar o armazenamento e execução distribuídas. O processo de redefinição será detalhado a seguir.

Considere que as tarefas concat e toupper são implementadas pelos serviços Web WS1 e WS2, respectivamente. Considerando que um mesmo serviço Web pode estar replicado em muitos servidores e visando encontrar as melhores alternativas de armazenamento, o **e-BioFlow** seleciona, dentre os servidores de armazenamento do ambiente, os servidores que possuem espaço disponível necessário para armazenar o dado produzido por cada

```
...
<s:processor name="concat">
 <s:arbitrarywsdl>
   <s:wsdl>
     http://localhost:8080/ime/services/Services.jws?wsdl
   </s:wsdl>
   <s:operation>concat</s:operation>
 </s:arbitrarywsdl>
</s:processor>
<s:processor name="toupper">
  <s:arbitrarywsdl>
    <s:wsdl>
      http://localhost:8080/ime/services/Services.jws?wsdl
    </s:wsdl>
    <s:operation>toupper</s:operation>
  </s:arbitrarywsdl>
</s:processor>
<s:link source="Sequencia1" sink="concat:input1" />
<s:link source="Sequencia2" sink="concat:input2" />
<s:link source="concat:concatReturn" sink="toupper:input" />
<s:link source="toupper:toupperReturn" sink="Resultado" />
<s:source name="Sequencia1" />
<s:source name="Sequencia2" />
<s:sink name="Resultado" />
...
```

FIG. 4.5: Trecho de workflow abstrato definido em SCUFL

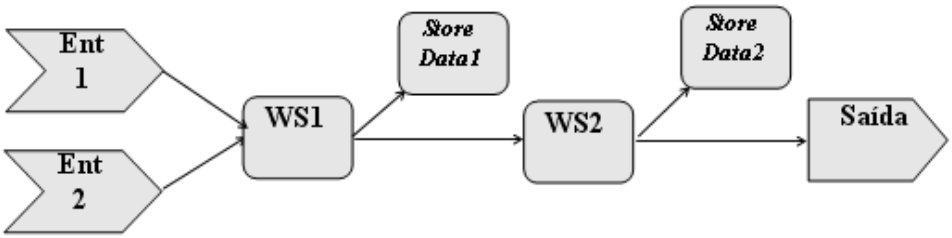

FIG. 4.6: Workflow executável W_{e1}, workflow com serviços de dados

tarefa. Para isso é estimado o tamanho produzido por cada tarefa, utilizando os dados da classe *DataType* do modelo de dados (seção 4.3). O **e-BioFlow** busca, utilizando a função de custo definida no capítulo 3 (Função 3.2, seção 3.4), para cada par de tarefas origem-destino, a melhor combinação de servidores que hospedem tanto as tarefas de origem e destino quanto o servidor de armazenamento. Identificados os servidores, o **e-BioFlow** repete o processo, sucessivamente, para as tarefas seguintes até a última tarefa do workflow. Finalizada essa etapa do processo de redefinição, um esquema do workflow W_{e1} se apresenta conforme mostrado na figura 4.6.

O armazenamento distribuído ocorre através da associação dos serviços Web de dados às saídas dos serviços Web que implementam as tarefas. Cada sítio de armazenamento disponibiliza os serviços *StoreData* e *GetData*, responsáveis pelo armazenamento e recuperação de dados, respectivamente. O serviço Web *StoreData* recebe o dado a ser

```
...
<s:processor name="concat">
 <s:arbitrarywsdl>
   <s:wsdl>
     http://localhost:8080/ime/services/Services.jws?wsdl
   </s:wsdl>
   <s:operation>concat</s:operation>
 </s:arbitrarywsdl>
</s:processor>
<s:processor name="toupper">
  <s:arbitrarywsdl>
    <s:wsdl>
      http://localhost:8080/ime/services/Services.jws?wsdl
    </s:wsdl>
    <s:operation>toupper</s:operation>
  </s:arbitrarywsdl>
</s:processor>
<s:link source="Sequencia1" sink="concat:input1" />
<s:link source="Sequencia2" sink="concat:input2" />
<s:link source="concat:concatReturn" sink="toupper:input" />
<s:link source="toupper:toupperReturn" sink="Resultado" />
<s:source name="Sequencia1" />
<s:source name="Sequencia2" />
<s:sink name="Resultado" />
...
```

FIG. 4.7: Trecho de workflow executável W_{e1}, arquivo SCUFL

persistido como parâmetro e retorna o LSID referente ao dado que foi persistido. O serviço Web *GetData* recebe o LSID referente ao dado requerido como parâmetro e retorna o dado propriamente dito. A persistência dos dados nos servidores remotos é feita via replicação do dado, isso ocorre para que o armazenamento dos dados não impacte a execução do workflow. O trecho de um arquivo em SCUFL é mostrado na figura 4.7.

Neste momento o workflow W_{e1} está apto para a execução com armazenamento distribuído porém, faltam informações ao gerente de workflows. Para dar suporte a proveniência de dados e recursos, como a re-execução parcial por exemplo, o gerente de workflows precisa saber quais dados foram produzidos por quais serviços Web, em cada execução do workflow, dada uma entrada de dados. Desta forma, uma estrutura de captura de dados da execução deve ser acrescentada ao workflow. O **e-BioFlow** acrescenta uma estrutura de coleta de dados da execução distribuída, composta por uma porta de entrada de dados WI e chamadas ao serviço Web WorkflowService, publicado no gerente de workflows. A porta WI é utilizada para passar o *id* da instância do workflow como parâmetro, o serviço Web WorkflowService registra os dados produzidos por cada serviço Web. Para isso, WorkflowService recebe como parâmetro o id da instância corrente, a identificação do serviço Web e o LSID do dado gravado pelo serviço *StoreData*.

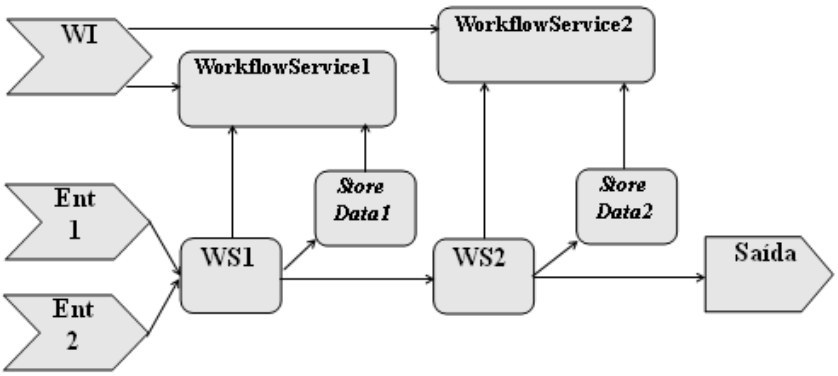

FIG. 4.8: Workflow executável W_{e1}, com estrutura de gerenciamento de execução

A figura 4.4 ilustra o estado final do workflow We1.

Finalizada a etapa de **Pré-Execução**, o workflow executável construído é enviado ao módulo **Engine de Execução** para realizar a sua execução. O módulo de Engine de Execução realiza a chamadas em background para o sistema de gerência de workflows selecionado. Conforme dito anteriormente, na atual implementação do **e-BioFlow**, o sistema de gerência execução de workflows é o Taverna, podendo ser estendido para novos sistemas. O **Engine de Execução** recupera os dados de entrada do workflow, armazenados no **Repositório Central de Dados**, realiza os ajustes de formato necessários para o sistema de gerência de execução selecionado e inicia a execução do workflow via chamada em background. Finalizada a execução do workflow, são coletados os dados do log de execução e resultado final, que são armazenados no **Repositório Central de Dados**.

4.3 MODELAGEM DE DADOS

O e-BioFlow possui um repositório central de dados, baseado no metamodelo de dados apresentado no capítulo 3 (seção 3.2). Para implementação, alguns ajustes foram realizados visando simplificar a construção inicial do e-BioFlow. As figuras 4.9 e 4.10 apresentam os esquemas utilizados para o gerente central e servidores remotos de armazenamento, respectivamente.

Como o e-BioFlow foi projetado para suportar múltiplos engines de execução de workflows, conseqüentemente com linguagens de definição diferentes, a classe *Workflow* recebeu o atributo type para que os workflows abstratos possam ser classificados de acordo com a linguagem de definição. No estágio atual, o **e-BioFlow** reconhece e classifica

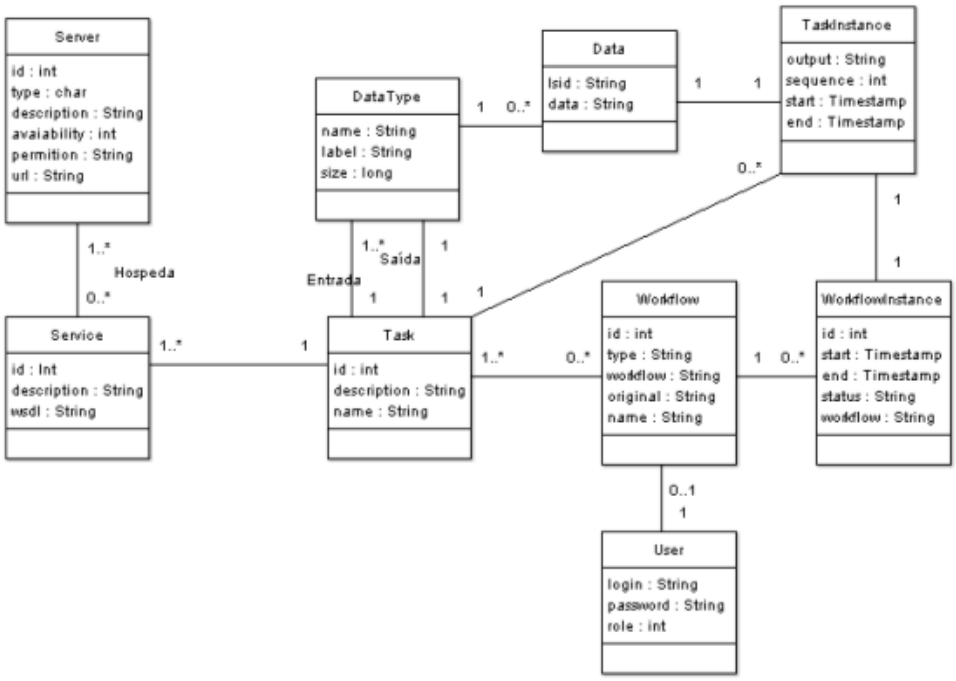

FIG. 4.9: Modelo de dados gerente de workflows

workflows definidos em SCUFL. O atributo *original* foi acrescentado para identificar uma possível relação entre workflows derivados e originais. Quando um workflow é derivado de outro, o atributo original contém o id do workflow a que se refere. Quando o workflow não derivar de nenhum (workflow semente) o atributo original possui valor 0. A classe *TaskInstance*,que representa as instâncias de execução de cada serviço, recebeu o atributo *sequence* para identificar o ponto do encadeamento o qual a referida execução representa. A classe *WorkflowInstance*, que representa as instâncias do workflow, recebeu o atributo status. O campo status registra o estado de execução do workflow. Quando um workflow encerra o processo de pré-execução fica em estado Ready(RD), quando sua execução é efetivamente iniciada seu estado é modificado para Execute(EX). Finalizada a execução, o estado do workflow pode ser atualizado para dois estados Successfull(SC) ou Error(ER).

O relacionamento entre as classes *Task* e *DataType*, foram modelados de forma que uma tarefa possa receber um ou mais tipos de dados, mas as tarefas só podem produzir dados de um tipo. A classe *DataType* recebeu um atributo label que contém um tipo primitivo de dados que é utilizado na construção do serviço Web Services (descrito na seção anterior).

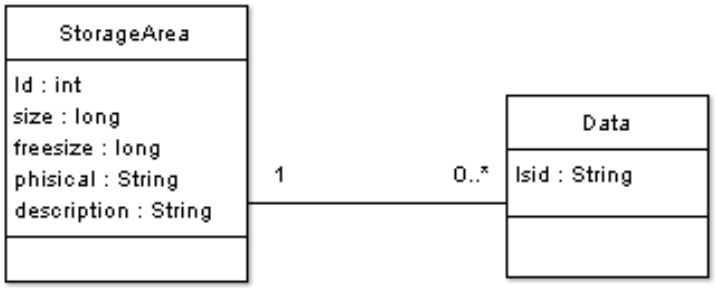

FIG. 4.10: Modelo de dados servidores remotos

4.4 INTERFACES DO E-BIOFLOW

Um usuário de perfil cientista realiza três atividades no ambiente proposto: definição do workflow abstrato, registro de um workflow abstrato e solicitação de execução dos workflows. A definição do workflow abstrato é realizada com auxílio de uma ferramenta de construção de workflows, como o editor gráfico de workflows do Taverna, por exemplo. O registro do workflow abstrato e as solicitações de execução são realizadas no **e-Bioflow**. Podem ser feitas execuções totais ou re-execuções parciais. Uma execução é total quando ocorre a partir da primeira tarefa com carga de dados de entrada via upload. Numa re-execução parcial, o cientista escolhe uma instância do workflow e solicita a sua execução a partir de uma determinada tarefa. Um usuário de perfil administrador é responsável pela manutenção dos cadastrados de infra-estrutura do **e-BioFlow**, tais como usuários, tarefas, servidores e serviços Web. Esta seção apresenta as principais interfaces implementadas do **e-BioFlow**. As subseções 4.4.1, 4.4.2 e 4.4.3 apresentam as principais interfaces do usuário cientista.

4.4.1 DEFINIÇÃO DE WORKFLOW ABSTRATO

Nessa interface o usuário cientista cria um workflow abstrato, definindo apenas a organização das tarefas. Para isso utiliza-se uma ferramenta de criação de workflows, um editor gráfico por exemplo. As tarefas são representadas como serviços Web que devem ser configurados para o serviço Web *Services*, publicado pelo **e-BioFlow**, diferenciando-se pela operação a ser realizada. Ao final dessa etapa têm-se um workflow abstrato composto apenas pelo encadeamento das tarefas, que apontam para um mesmo serviço Web. A construção de um workflow abstrato com o editor gráfico do Taverna é mostrado na figura 4.11.

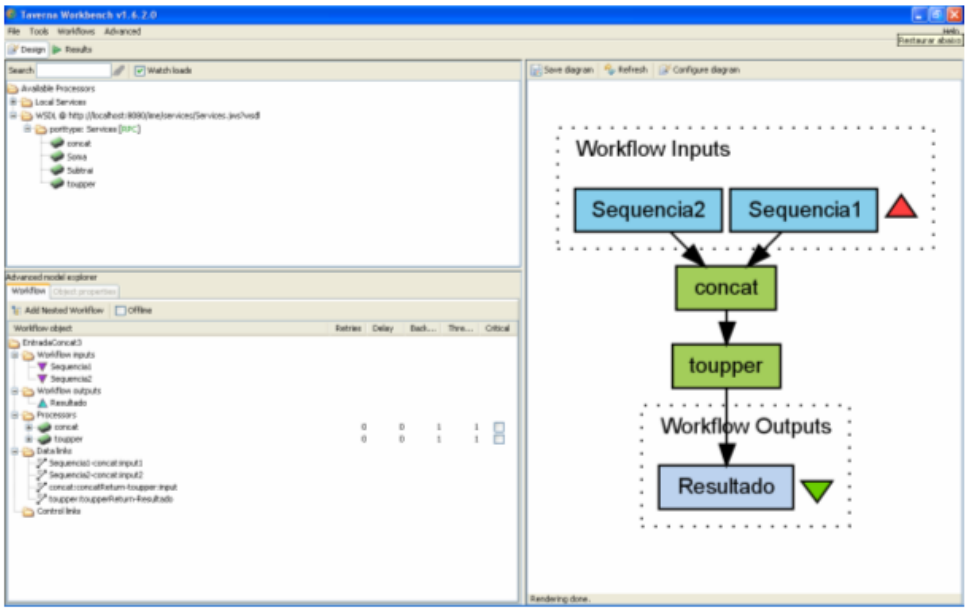

FIG. 4.11: Editor gráfico do Taverna

No editor gráfico do Taverna, os serviços disponíveis aparecem na parte superior esquerda da tela. Quando inserido no arquivo de configuração, *mygrid.properties*, o serviço Web *Services* torna-se disponível na paleta de serviços. O cientista procede então a montagem do workflow abstrato. O frame á direita mostra a representação gráfica do encadeamento das tarefas. No exemplo da figura 4.11, é ilustrada a construção de um workflow abstrato composto por duas tarefas: concat e toupper.

4.4.2 REGISTRO DE WORKFLOW ABSTRATO

Para o cientista registrar o workflow abstrato no repositório central do **e-BioFlow**, na tela de registro do workflow abstrato (figura 4.12), o cientista deve informar: o arquivo do workflow, um nome único para o workflow e o workflow origem, caso se trate de uma derivação. O **e-BioFlow** suporta apenas um nível de derivação de workflows, isto é, o **e-BioFlow** não reconhece múltiplos níveis de derivação para efeitos de reaproveitamento de dados.

4.4.3 SOLICITAÇÃO DE EXECUÇÃO

Quando é solicitada uma execução de um workflow abstrato, o mesmo é recuperado do repositório central e passa então por uma etapa de pré-execução, onde o workflow abstrato será redefinido, gerando um workflow executável, que será então encaminhado

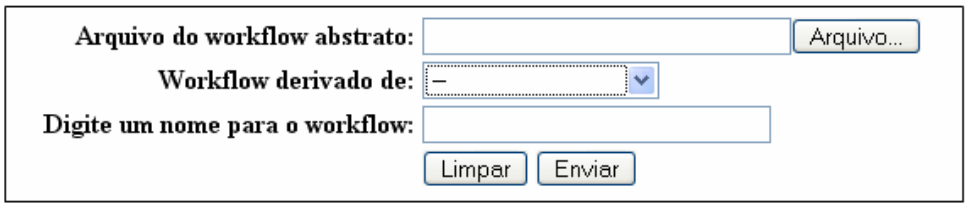

FIG. 4.12: Tela de registro de workflows abstratos no e-BioFlow

para execução. O usuário cientista deve primeiro selecionar qual workflow deseja executar. Na tela mostrada na figura 4.13, seleciona-se o workflow SubGarsa.

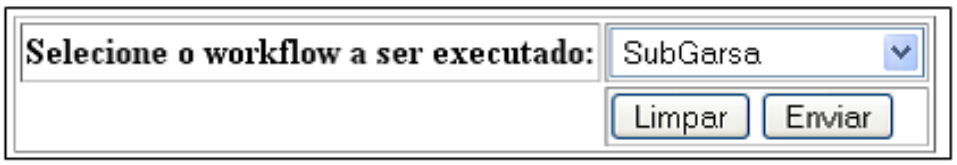

FIG. 4.13: Seleção de workflows

Selecionado o workflow, o usuário deve selecionar o tipo de execução que deseja fazer, se uma execução total ou re-execução parcial, como ilustrado na figura 4.14.

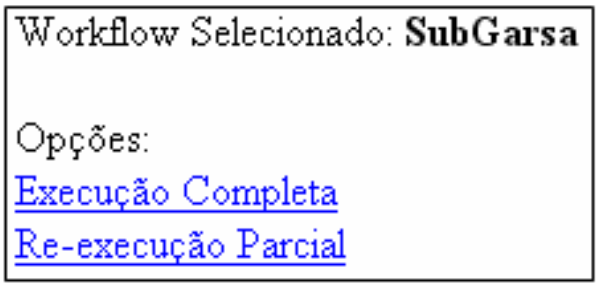

FIG. 4.14: Seleção de workflows

Caso deseje realizar uma execução total, o usuário deverá realizar o upload do arquivo de entrada de dados e iniciar a execução do workflow, conforme ilustrado na figura 4.15.

O arquivo de entrada de dados consiste de um arquivo XML construído pelo cientista seguindo o esquema mostrado na figura 4.16.

Um exemplo de arquivo de entrada de dados para o esquema acima é mostrado na figura 4.17.

Quando o cientista fornece o arquivo de entrada de dados, o **e-BioFlow** realiza o processamento do arquivo. Em primeiro lugar separa os dados de entrada do workflow e cria instância da classe Data para armazenar no repositório de dados (de acordo com

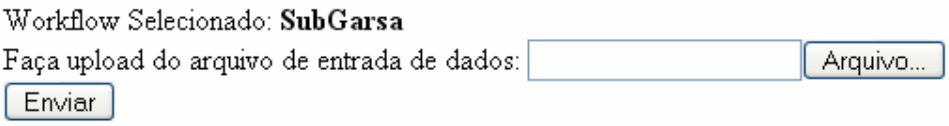

FIG. 4.15: Carga do arquivo de entrada de dados e início da execução

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.e-bioflow.org"
xmlns="http://www.e-bioflow.org"
elementFormDefault="qualified">
  <xs:element name="workflow">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="input">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="portName" type="xs:string"/>
              <xs:element name="data" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

FIG. 4.16: Exemplo de um esquema xml para o arquivo de entrada de um workflow

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <workflow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="inputschema.xsd">
  - <input>
      <portName>Sequencia1</portName>
      <data>AACCTGTAATTCAA</data>
    </input>
  - <input>
      <portName>Sequencia2</portName>
      <data>TTGCATGACCCCGGGGGATTY</data>
    </input>
</workflow>
```

FIG. 4.17: Exemplo de arquivo de entrada de dados

o modelo de dados da seção 4.3). Depois, associa os dados recém armazenados às suas respectivas portas (indicadas pela tag portName) de entrada do workflow.

Caso o cientista deseje realizar uma re-execução parcial, ele deverá selecionar uma instância do workflow e indicar a partir de qual tarefa a execução será reiniciada, conforme mostrado na figura 4.18.

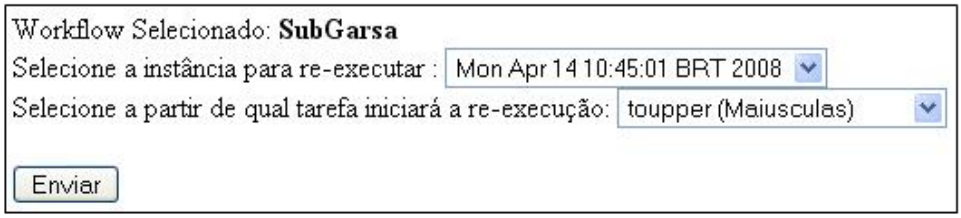

FIG. 4.18: Tela de re-execução parcial

4.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o **e-BioFlow**, um software que apóia o cientista nas tarefas de definição e execução de workflows de bioinformática distribuídos, com recursos de gerenciamento de dados distribuídos. A idéia inicial deste protótipo é validar a abordagem de distribuição de dados proposta no capítulo 3. Esse protótipo servirá de base para um sistema mais completo, onde será possível a execução de testes com workflows reais de bioinformática, validando a proposta de forma mais completa.

Como o protótipo ainda não pode ser utilizado em casos reais, neste trabalho foi construído um simulador para validar a proposta. O detalhamento do modelo de simulação, bem como os resultados obtidos serão apresentados no próximo capítulo.

5 MODELO DE SIMULAÇÃO

No capítulo 3 foi proposta uma abordagem para distribuição de dados em workflows de bioinformática, com apresentação de uma arquitetura, um metamodelo de dados e uma função de custos. Para validar a abordagem proposta, sugere-se um modelo de simulação, composto por um gerador de massa de dados e um simulador que serão detalhados neste capítulo.

Na seção 5.1 são especificados um ambiente de simulação e os parâmetros definidos para realizar a simulação. A seção 5.2 apresenta um gerador de massa de dados necessário para o uso do simulador. A seção 5.3 apresenta o simulador proposto para validar a proposta. Na seção 5.4 é realizada a análise dos resultados obtidos pela simulação. A seção 5.5 apresenta algumas considerações finais.

5.1 AMBIENTE DE SIMULAÇÃO E PARÂMETROS UTILIZADOS

Para tornar possível a simulação, foram definidos um ambiente e um conjunto de parâmetros. O ambiente projetado para simulação é constituído de um conjunto de servidores, localizados em sítios remotos, que hospedam serviços de processamento e áreas de armazenamento. Os serviços implementam tarefas e podem estar hospedados em mais de um servidor. Para facilitar análises comparativas entre os diferentes paradigmas de armazenamento, que serão detalhados na seção 5.4, admitiu-se que não houvesse problemas de falta de espaço nas áreas de armazenamento do ambiente. Os parâmetros definidos para geração e execução simulada dos workflows foram:

- Parâmetros de Tarefas:
  - Número de tarefas: foram utilizadas um total de 15 tarefas;
  - Número de grupos de tarefas: as tarefas foram agrupadas em 5 grupos de similaridades de função;
  - Total de tarefas em cada grupo: cada grupo conterá 3 tarefas;
  - Interdependência entre tarefas: definir um conjunto de regras de associação entre os grupos de tarefas, mostrando quais tarefas podem se associar a quais

outras tarefas, formando um par ordenado dentro da seqüência de execução de um workflow. Para a simulação realizada foram utilizados 5 grupos contendo 3 tarefas cada;

- Parâmetros de Workflows:
  - Número mínimo e máximo de tarefas por workflow: total de tarefas que podem compor o workflow, caracterizando com isso o tipo de workflow típico da aplicação em questão. Para esta simulação adotou-se um mínimo de 3 e um máximo de 5 tarefas por workflow, caracterizando workflows pequenos e simples;
  - Número de workflows abstratos: total de workflows em uso no ambiente científico em estudo, incluindo suas derivações, caracterizando o porte do ambiente. Para esta simulação, foram utilizados 50 workflows abstratos distintos, incluindo derivações;
  - Número/Percentual de workflows derivados: total de workflows derivados a partir dos workflows abstratos definidos, caracterizando um ambiente onde a derivação é freqüente. Nesta simulação os workflows abstratos possuem de 0 a 5 workflows derivados:
- Parâmetros de Dados:
  - Tamanho da entrada de dados: a massa de dados de entrada dos workflows deve variar de 100KB a 10MB;
- Parâmetros de Servidores: número de servidores simulados: 16 (15 + 1 servidor central);

5.1.1 BASE DE DADOS

Para apoiar tanto o gerador de massa de dados quanto o simulador, e com base no metamodelo de dados apresentado no capítulo 3, foi definida uma base de dados de simulação, cuja modelagem é mostrada na figura 5.1.

A classe Tarefa modela as tarefas que podem compor um workflow abstrato, o atributo saída representa um fator de multiplicação para o dado de entrada. Desta forma, caso uma tarefa que possua o valor 1,35 associado ao atributo saída e receba como entrada

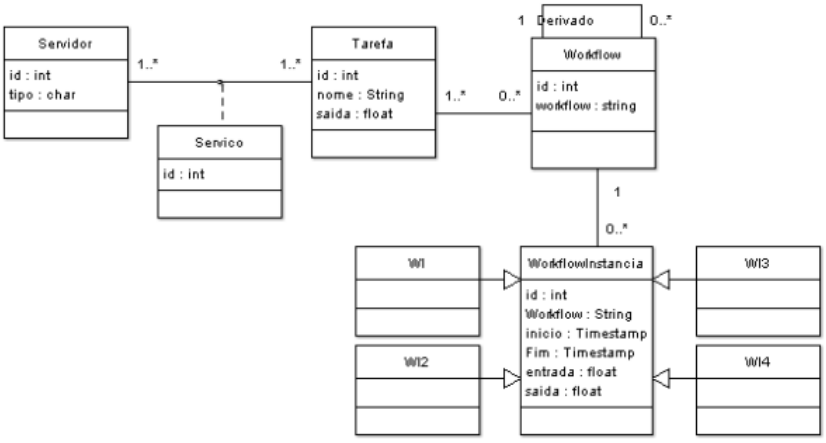

FIG. 5.1: Esquema da base de simulação

um dado de tamanho 10KB, sua saída terá o tamanho de 13,5KB. A classe Servidor modela os servidores, hospedados em sítios remotos. A classe Serviço modela os serviços disponibilizados pelos servidores. A classe Workflow modela os workflows abstratos, os workflows abstratos podem possuir workflows derivados. Os workflows executáveis são modelados pela classe WorkflowInstancia, os atributos inicio e fim armazenam as etiquetas de tempo que marcam o início e fim da execução do workflow. O atributo entrada armazena o tamanho do dado de entrada utilizado para a execução, o atributo saída armazena o resultado final produzido pela execução do workflow. As classes WI, WI2, WI3, WI4 modelam especializações da classe WorkflowInstancia que armazenam os dados das execuções de acordo com os paradigmas de armazenamento utilizados para simulação, cujo detalhamento encontra-se na seção 5.4.

5.1.2 MATRIZES DE CUSTOS

A matriz de custos de transferência de dados é armazenada em um arquivo de parâmetros, cujos índices de linha e coluna representam servidores de origem e destino, respectivamente. Um exemplo de matriz de custos de transferência de dados utilizada no simulador, para o caso de ambiente de 16 servidores, é mostrada na figura 5.2.

Para simulação, admitiu-se que o custo de comunicação interna de um servidor é desprezível. Portanto, os elementos pertencentes à diagonal principal da matriz têm custo zero. A matriz de custos mostrada na figura 5.2 apresenta simetria porque, para simplificação do problema, admitiu-se que o custo de comunicação fosse recíproco entre

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
S1	0	2	1	3	3	5	7	8	9	8	6	6	5	9	7	3
S2	2	0	2	2	2	5	7	8	9	8	6	6	5	9	7	2
S3	1	2	0	3	3	5	7	8	9	8	6	6	5	9	7	2
S4	3	2	3	0	1	5	6	7	8	7	6	6	5	8	7	1
S5	3	2	3	1	0	5	6	7	8	7	6	6	5	8	7	1
S6	5	5	5	5	5	0	4	8	9	8	2	2	1	9	7	5
S7	7	7	7	6	6	4	0	8	8	8	5	5	4	9	7	6
S8	8	8	8	7	7	8	8	0	4	1	9	9	8	4	8	7
S9	9	9	9	8	8	9	8	4	0	4	9	9	9	1	9	8
S10	8	8	8	7	7	8	8	1	4	0	9	9	8	4	8	7
S11	6	6	6	6	6	2	5	9	9	9	0	1	2	9	6	6
S12	6	6	6	6	6	2	5	9	9	9	1	0	2	9	6	6
S13	5	5	5	5	5	1	4	8	9	8	2	2	0	9	7	5
S14	9	9	9	8	8	9	9	4	1	4	9	9	9	0	9	8
S15	7	7	7	7	7	7	7	8	9	8	6	6	7	9	0	7
S16	3	2	3	1	1	5	6	7	8	7	6	6	5	8	7	0

FIG. 5.2: Matriz de custos de transferência de dados entre servidores

os servidores.

Os vetores de custos de armazenamento e recuperação de dados dos servidores remotos são armazenados em arquivos de parâmetros. Um exemplo de vetores de armazenamento e recuperação, para o caso de 16 servidores, utilizados para simulação são mostrados nas figuras 5.3 e 5.4, respectivamente. O custo de armazenamento e recuperação de cada servidor, é representado pelo valor correspondente no vetor. Desta forma, por exemplo, o custo de armazenamento do sítio 2 é 3.

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
3	3	4	4	4	3	7	4	4	4	5	5	7	7	6	3

FIG. 5.3: Vetor de custos de armazenamento de dados

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
3	3	4	4	4	3	7	4	4	4	5	5	7	7	6	3

FIG. 5.4: Vetor de custos de leitura de dados

5.1.3 ARQUITETURA E AMBIENTE DE SIMULAÇÃO

Um diagrama ilustrando o funcionamento e as interações do gerador de massa de dados e simulador é mostrado na figura 5.5. O módulo **Gerador de Massa de Dados** é responsável pela geração dos workflows abstratos, o módulo **Simulador** realiza a simulação propriamente dita. Maiores detalhes serão mostrados nas seções seguintes. O computador utilizado para realizar a simulação possui a seguinte configuração:

- Processador: AMD Athlon X2 3.6Ghz;
- Memória RAM: 2GB;
- Sistema Operacional: Windows XP Professional SP2;
- Java Virtual Machine: 1.6.0_03-b05

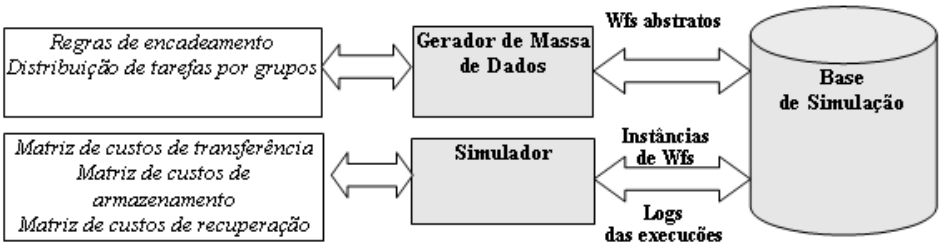

FIG. 5.5: Diagrama de interação do gerador de massa de dados e simulador

5.2 GERADOR DE MASSA DE DADOS

O gerador de massa de dados atua na fase de definição de workflows, produzindo workflows abstratos, de acordo com os parâmetros definidos na seção 5.1. Foi definida uma estrutura simplificada de representação de workflows, onde os workflows abstratos são representados por strings contendo as tarefas separadas pelo símbolo "$->$". Desta forma, um exemplo de workflow abstrato composto pelo encadeamento das tarefas *T1, T3, T5* seria representado como $T1->T3->T5$.

Para geração dos workflows abstratos, o gerador de massa de dados realiza a leitura de dois arquivos de parâmetros: o primeiro contém a distribuição das tarefas por grupo e o segundo contém as regras de encadeamento dos grupos de tarefas. As regras de encadeamento dos grupos de tarefas apresentam a estrutura $< grupo >:< sucessor1 >, < sucessor2 >$. Assim, por exemplo,uma regra 2:1,3,4 indica que uma tarefa pertencente ao grupo 2 pode ser sucedida por tarefas dos grupos 1, 3, 4.

Para construção de um workflow, o gerador de massa de dados primeiro define o número de tarefas que o mesmo possuirá. Essa decisão se dá através da geração de números aleatórios utilizando uma função randômica. Definido o número de tarefas, o gerador define, mais uma vez de forma aleatória, qual grupo de tarefas será utilizado para selecionar a primeira tarefa do workflow.

Definido o grupo de tarefas, o gerador seleciona, aleatoriamente, a tarefa. O gerador então identifica no conjunto de regras, quais grupos de tarefas podem ser utilizados para selecionar a tarefa seguinte. O processo repete-se até que o workflow em questão atinja o número de tarefas encadeadas definido no início do processo de construção. Durante a construção dos workflows, o gerador de massa de dados inibe a repetição de tarefas num mesmo workflow.

Terminado processo de construção do workflow, o gerador de massa de dados verifica se o workflow construído já está catalogado na base de dados de simulação. Caso não seja, o workflow é catalogado na base de simulação. O gerador de massa de dados pode determinar ainda se o workflow gerado terá ou não workflows derivados. Essa decisão é tomada de forma a garantir que no máximo 35% dos workflows terão derivações. O gerador de massa de dados decide então quantos workflows derivados existirão e em quais pontos do workflow ocorrerá a derivação, ambas as decisões são tomadas de forma aleatória.

5.3 SIMULADOR

O simulador implementa os módulos de pré-execução e execução dos workflows da arquitetura **D-Bioflow**, apresentada no capítulo 3. Os workflows abstratos, registrados na base de simulação pelo gerador de massa de dados, são redefinidos, produzindo workflows executáveis, e têm suas execuções simuladas. Para realizar as tarefas de redefinição e execução, o simulador realiza a leitura de três arquivos de parâmetros, mais informações obtidas da base de simulação. Os arquivos de parâmetros possuem as seguintes informações: a matriz de custos de comunicação entre os sítios, o vetor de custos de armazenamento de cada sítio e o vetor de custos de recuperação de dados de cada sítio. As demais informações como serviços disponíveis, número de instâncias dos workflows, etc. são obtidas da base de simulação. Os processos de redefinição e execução de workflows serão detalhados a seguir.

Iniciada a sua execução, o simulador realiza a leitura dos arquivos de parâmetros e em seguida carrega todos os workflows abstratos registrados na base de simulação. Para cada workflow abstrato, o simulador decide, aleatoriamente, quantas instâncias o workflow abstrato terá (máximo de 30 instâncias). Para cada instância de workflow, o simulador define se a mesma se trata de uma instância nova, se é uma re-execução total de uma instância já existente ou se é uma re-execução parcial. Caso a instância seja uma nova instância, o simulador realizará a redefinição do workflow abstrato, gerando uma instância

de workflow (workflow executável) para cada um dos paradigmas de armazenamento, detalhados mais adiante. Caso seja uma re-execução total, o simulador selecionará, aleatoriamente, uma instância existente na base de simulação e a mesma será re-executada com nova massa de dados. Caso a instância do workflow seja re-execução parcial, o simulador selecionará aleatoriamente qual instância será utilizada para re-execução e a partir de que tarefa a re-execução ocorrerá.

A instância do workflow estende a estrutura de representação dos workflows abstratos, apresentado na seção anterior. Um serviço de processamento é representado através da combinação da tarefa que ele implementa com o sítio que o hospeda. Assim um exemplo de um serviço que implementa a tarefa $T1$ e que esteja hospedado no sítio *S3*, será representado na instância de workflow na forma *S3:T1*. Os repositórios de armazenamento dos dados intermediários são representados apenas com o nome dos sítios que os hospedam. Desta forma, o trecho de instância de workflow $S3 : T1 -> S5$ indica que o resultado do serviço hospedado no sítio *S3*, que implementa a tarefa *T1* será armazenado no sítio *S5*.

O simulador foi projetado para criar instâncias de workflows, de acordo com quatro diferentes paradigmas de armazenamento, os paradigmas utilizados foram: centralizado, posterior, distribuído e **D-Bioflow**. O paradigma de armazenamento centralizado prevê a existência de um único servidor de armazenamento, desta forma os dados produzidos pela execução de um workflow são gravados no servidor central de armazenamento para depois serem enviados para processamento da tarefa seguinte. O paradigma posterior prevê a distribuição dos dados intermediários, de forma que os dados produzidos por uma tarefa sejam sempre armazenados no servidor que hospeda a tarefa seguinte do encadeamento (tarefa consumidora do dado). A abordagem distribuída sem heurística faz uso de uma função de custos para encontrar, dentre os servidores do ambiente, aqueles que representam as melhores alternativas de armazenamento (equação 3.1 do capítulo 3). Por fim, a abordagem **D-BioFlow**, utiliza uma função de custos combinada a uma heurística que busca reaproveitar dados de execuções de workflows derivados (equação 3.2 do capítulo 3).

O simulador gera, aleatoriamente, respeitando os parâmetros estabelecidos na seção 5.1, o tamanho do dado de entrada que será utilizado para simular a execução. Definidos o número de instâncias, os workflows executáveis e o tamanho do dado de entrada, é iniciada a simulação da execução dos workflows.

5.4 RESULTADOS OBTIDOS

A simulação foi realizada utilizando-se 50 workflows abstratos. A partir deles, o simulador gerou um total de 320 workflows executáveis para cada abordagem de armazenamento. Para comparação dos resultados obtidos, foi considerado o tempo médio de execução dos workflows executáveis. O simulador realiza 3 tipos de simulação: execução completa, re-execução completa e re-execução parcial. A execução completa representa uma execução de ciclo completo, quando um workflow abstrato é selecionado para execução. Numa re-execução total, um workflow executável é escolhido e re-executado com nova entrada de dados. Uma re-execução parcial ocorre quando apenas parte de um workflow é executada. O gráfico apresentado na figura 4 mostra uma comparação do tempo médio de execução das 4 abordagens consideradas.

O propósito da simulação era comparar quatro diferentes abordagens de armazenamento: centralizado, distribuído posterior, distribuído sem heurística e distribuído com heurística, sendo esta última a proposta da arquitetura **D-BioFlow**. A abordagem centralizada simula um ambiente em que um único servidor do ambiente guarda todos os dados intermediários dos workflows, tradicionalmente utilizado nas abordagens existentes InServices, Kepler, Taverna e Vistrails. No contexto do armazenamento distribuído dos dados intermediários de um workflow, na abordagem posterior, os dados intermediários são sempre gravados no servidor que hospeda a tarefa subseqüente.

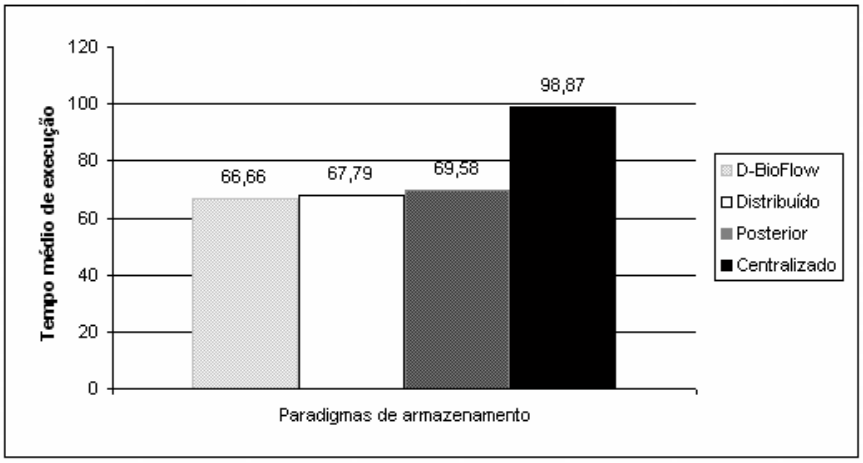

FIG. 5.6: Tempo médio de execução para cada abordagem de armazenamento

Observa-se que as 3 abordagens distribuídas apresentaram ganho de tempo na

execução dos workflows, quando comparadas com a abordagem centralizada. A abordagem servidor posterior, apresenta um ganho de tempo médio de execução de aproximadamente 29,6%. A abordagem **D-BioFlow** apresentou um ganho de tempo médio de execução de 32,58%. Dentre as abordagens distribuídas, a abordagem **D-BioFlow** apresentou um ganho aproximado de 4,2%, frente à abordagem servidor posterior.

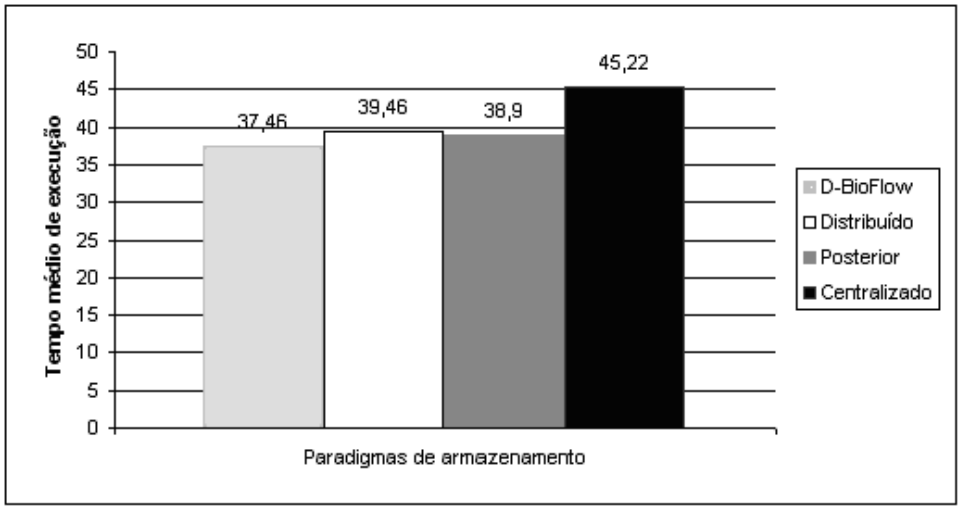

FIG. 5.7: Tempo médio de re-execuções parciais para cada abordagem de armazenamento

O gráfico da Figura 5.7 apresenta os dados obtidos, considerando apenas os casos de re-execuções parciais e um percentual de re-execuções parciais de aproximadamente 23%. Observa-se que a abordagem **D-BioFlow** apresentou um ganho de aproximadamente 17,16% em relação à abordagem centralizada. Comparado à abordagem distribuída sem heurística, o **D-BioFlow** apresentou um ganho de aproximadamente 5%. Esse resultado evidencia a atuação da heurística discutida no capítulo 3 deste trabalho. Ampliando o número de workflows executáveis para 411, foi observado um aumento no ganho, do **D-BioFlow**, para aproximadamente 6,5%. Considerando ambientes reais, onde diversos workflows têm inúmeras re-execuções parciais (devido grande tempo requerido no processamento das tarefas) o uso da heurística será mais vantajoso, visto que haverá maiores oportunidades de reaproveitamento de dados.

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um modelo de simulação utilizado para validação da abordagem **D-BioFlow**. Outras possíveis simulações seriam com diferentes números de servidores (a simulação foi realizada com 16 servidores), diferentes combinações de matrizes de custos, simulações envolvendo servidores reais e diferentes faixas de entrada de dados. Não está no escopo deste trabalho esgotar todas as possibilidades. Os resultados obtidos poderiam ser separados por faixa de dados, mas o intervalo de tamanho de dados mostrou pouca variação nessa apresentação. Novos testes utilizando cargas maiores de dados devem ser realizados como trabalhos futuros. Apesar disso, os resultados apurados são suficientes para apontar que a abordagem proposta traz ganhos de desempenho no tempo de execução dos workflows, com especial ênfase nos casos em que as re-execuções parciais são mais freqüentes. Além disso, os resultados da simulação mostraram que a abordagem **D-BioFlow** é promissora quanto ao crescente ganho de desempenho na medida em que cresce a taxa de re-execução parcial dos workflows, situação bastante freqüente em cenários reais de experimentos *in-silico*.

6 EXEMPLO DE USO DO E-BIOFLOW

Para ilustrar a idéia da arquitetura proposta e o funcionamento do protótipo **e-BioFlow**, este capítulo apresenta um exemplo de uso de um workflow real. O workflow utilizado é baseado no STINGRAY (WAGNER, 2007). Na seção 6.1 é realizada uma breve apresentação do workflow STINGRAY. A seção 6.2 apresenta o workflow exemplo. O exemplo de uso é apresentado na seção 6.3.

6.1 O WORKFLOW STINGRAY

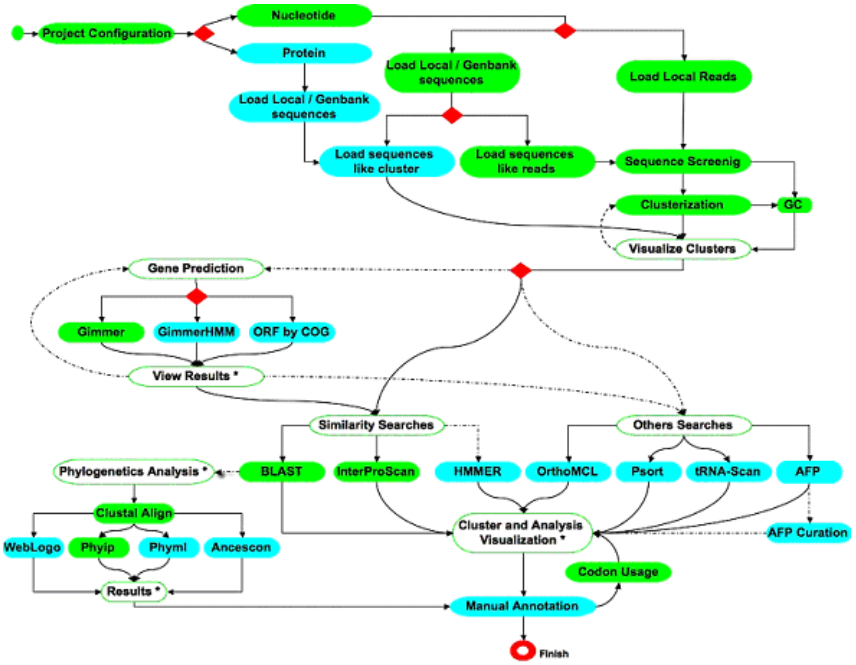

FIG. 6.1: Esquema de encadeamento do workflow STINGRAY

O workflow STINGRAY é utilizado para anotação genômica de alguns organismos em estudo pelo projeto BioWebDB [8], como por exemplo o *Trypanosoma vivax*, *Trypanosoma rageli* entre outros. Baseado no workflow GARSA (DÁVILA, 2005), o STINGRAY incorpora 26 programas de bioinformática, dentre os quais podem ser citados: BLAST (ALTSCHUL, 1997), Phred/Phrap (EWING, 1998b) (EWING, 1998a), CAP3 (HUANG,

[8] http://www.biowebdb.org

1999) e HMMER (EDDY, 2003). O workflow STINGRAY é implementado na linguagem Perl que contém as invocações embutidas nos programas mencionados. A figura 6.1 ilustra em detalhes como o workflow STINGRAY está definido. Está em andamento atualmente no projeto BioWebDB, onde esse workflow é utilizado, a migração gradual dos passos do workflow para a plataforma de serviços Web.

6.2 WORKFLOW EXEMPLO

No exemplo de uso apresentado neste capítulo, o workflow utilizado constitui-se dos passos iniciais do workflow STINGRAY. Esse workflow foi aqui denominado **we** e utiliza apenas os três primeiros passos definidos no workflow STINGRAY. Ou seja, o workflow é constituído pela seqüência de programas: Phred, Cap3 e Blast. A tarefa Phred extrai as seqüências genômicas a partir dos dados fornecidos de um seqüenciador genômico. A tarefa Cap3 monta as seqüências geradas pelo passo anterior em um ou mais conjuntos de segmentos de seqüências genômicas sobrepostas - também conhecida como clusterização de seqüências. Em seguida, esse conjunto de seqüências agrupadas é submetido a uma busca de similaridade com outras seqüências genômicas já conhecidas e armazenadas, através do uso da tarefa Blast. Para enriquecer o exemplo de uso foi utilizado também um segundo workflow, denominado **we'**, que é derivado de **we** diferenciando-se apenas pela execução da tarefa InterProScan (MULDER, 2005) no lugar da tarefa Blast.

Para criação do exemplo de uso todas as tarefas foram publicadas como serviços Web e o workflow abstrato foi definido através da linguagem SCUFL, utilizando o editor gráfico do Taverna. Apesar de já existirem versões de serviços Web para os quatro programas, os mesmos tiveram que ser implementados para atender o requisito da compatibilidade de tipos de dados entre os serviços Web do workflow. Ou seja, para os dados de saída do Phred poderem ser utilizados como dados de entrada no serviço do Cap3, e os dados de saída do serviço do Cap3 serem utilizados como dados de entrada do serviço do Blast houve a necessidade de torná-los compatíveis entre si. A Figura 6.2 mostra como o workflow exemplo **we** foi estruturado com os três serviços Web que o compõem. De maneira geral, no **e-BioFlow** a decisão pelo local de armazenamento se dá através do módulo de **Pré-Execução** que é responsável também pela inserção serviços Web de armazenamento no workflow executável, ligando-os à saída de cada tarefa original correspondente. Outra tarefa é acrescentada ao workflow executável, responsável por coletar informações sobre a execução de cada tarefa e armazená-las no **repositório central de dados**. Entre as

informações coletadas pode-se destacar: identificador da instância em execução e LSID do dado produzido pela tarefa. Nas seções seguintes será ilustrado um exemplo de uso, contendo as principais atividades dos usuários do **e-BioFlow** para definição e execução de **we**.

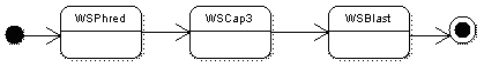

FIG. 6.2: Esquema de encadeamento das tarefas do workflow exemplo we

6.3 EXEMPLO DE USO

6.3.1 REGISTRANDO AS TAREFAS DO WORKFLOW EXEMPLO

Para que seja possível construir o workflow, deve-se registrar no **e-BioFlow** os programas, bem como os serviços que os implementam. No registro das tarefas dos workflows são inseridos os metadados das tarefas, tais como o nome e descrição da tarefa, informações pertinentes aos seus parâmetros de entrada e saída (nome dos parâmetros e tipos de dados). Essas informações serão utilizadas durante a redefinição do workflow para estimar o espaço requerido para armazenar os dados. A Figura 6.3 ilustra o registro da tarefa WSBlast.

FIG. 6.3: Registro de Tarefa WSBlast no e-BioFlow

Uma vez registrada a tarefa, devem-se registrar os serviços Web que a implementam através da função de registro de serviços Web do **e-BioFlow**, conforme explicado no capítulo 4, seção 4.4. Além da tarefa WSBlast, foram registradas também as tarefas WSPhred e WSCap3.

6.3.2 CONSTRUINDO O WORKFLOW EXEMPLO ABSTRATO

Conforme apresentado neste trabalho, uma vez que as tarefas tenham sido registradas no **e-BioFlow**, uma representação vazia das mesmas é publicada como operações do serviço Web *Services*, conforme ilustrado na figura 6.4.

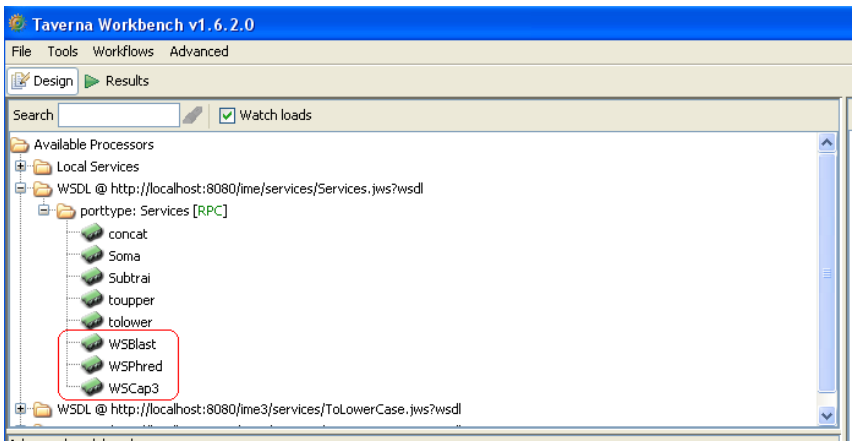

FIG. 6.4: Operações do serviço Web *Services*

Desta forma, o cientista pode construir o workflow **we abstrato** utilizando o editor gráfico do *Taverna*, ilustrado na figura 6.5. Note que workflow construído possuirá apenas a organização lógica das tarefas. Um techo do arquivo de definição do workflow **we** abstrato é apresentado na figura 6.6

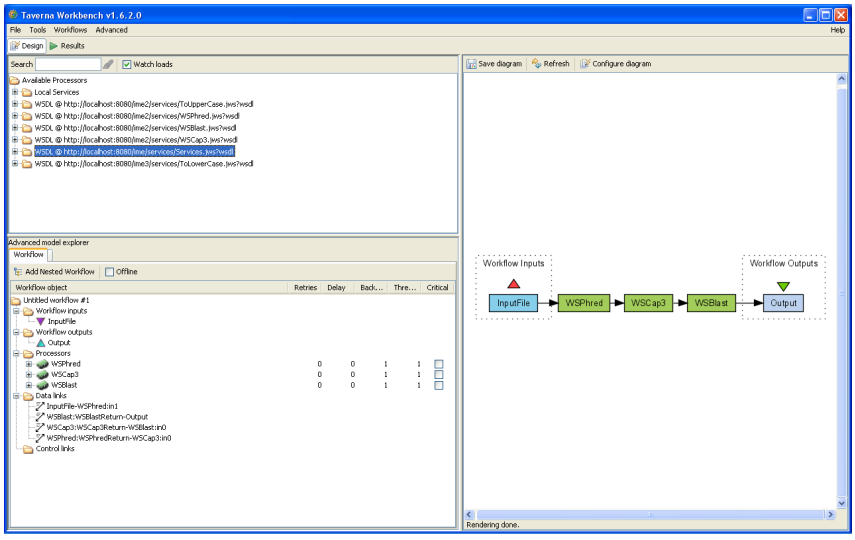

FIG. 6.5: Construção do workflow exemplo abstrato

Uma vez construído o workflow **we abstrato**, o cientista deverá então registrá-lo no

```
- <s:processor name="WSPhred">
  - <s:arbitrarywsdl>
    - <s:wsdl>
        http://localhost:8080/ime/services/Services.jws?wsdl
      </s:wsdl>
      <s:operation>WSPhred</s:operation>
    </s:arbitrarywsdl>
  </s:processor>
- <s:processor name="WSCap3">
  - <s:arbitrarywsdl>
    - <s:wsdl>
        http://localhost:8080/ime/services/Services.jws?wsdl
      </s:wsdl>
      <s:operation>WSCap3</s:operation>
    </s:arbitrarywsdl>
  </s:processor>
- <s:processor name="WSBlast">
  - <s:arbitrarywsdl>
    - <s:wsdl>
        http://localhost:8080/ime/services/Services.jws?wsdl
      </s:wsdl>
      <s:operation>WSBlast</s:operation>
    </s:arbitrarywsdl>
  </s:processor>
```

FIG. 6.6: Trecho do código SCUFL do workflow exemplo abstrato

repositório central de dados para que possa realizar as execuções desejadas, conforme explicado no capítulo 4.

6.3.3 DERIVANDO O WORKFLOW ABSTRATO WE

Suponha que o cientista deseje, para avaliar os resultados produzidos, construir um workflow semelhante ao workflow **we** porém substituindo a tarefa WSBlast pela tarefa WSInterProScan. Desta forma, ele irá construir um workflow abstrato **we'**, derivado do workflow **we**, com a composição de tarefas ilustrado na figura 6.7.

FIG. 6.7: Esquema de encadeamento das tarefas do workflow exemplo we'

Para construir **we'** o cientista executa as mesmas etapas para a construção do workflow **we**. A diferença ocorre no registro de **we'** no **repositório central de dados**. No registro de **we'**, o cientista deve informar o nome do workflow original (**we** neste caso). Um trecho do workflow **we'** abstrato é mostrado na figura 6.8.

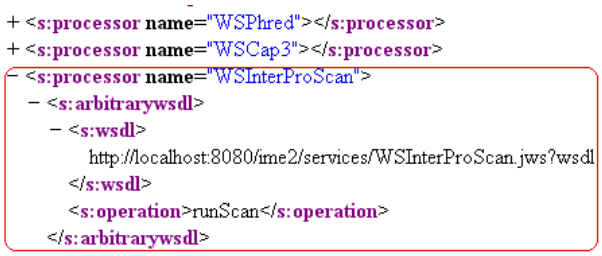

FIG. 6.8: Trecho do código SCUFL do workflow we' abstrato

6.3.4 EXECUÇÃO COMPLETA

Para este exemplo de execução, considere que todos os servidores do ambiente tenham espaço suficiente para armazenamento dos dados intermediários. Considere ainda que o **Repositório Central de Dados** já possua dados e metadados referentes a 5 execuções do **we'** e 1 execução do **we**.

Considere um ambiente composto pela seguinte distribuição de serviços Web e servidores:

- Servidor 0: Gerente de Workflows, IP: 192.168.0.15;
- Servidor 1: Serviço Web WSPhred, IP: 192.168.0.20;
- Servidor 2: Serviço Web WSCap3, IP: 192.168.0.21;
- Servidor 3: Serviço Web WSBlast, IP: 192.168.0.22;
- Servidor 4: Serviço Web WSInterProScan, IP: 192.168.0.25;

Neste cenário, o cientista solicita uma execução completa do workflow **we**. Para realizar a execução, o cientista deve fornecer o arquivo de entrada de dados. No caso do workflow exemplo utilizado, o serviço Web que implementa a tarefa WSPhred recebe o conjunto de arquivos do seqüenciador na forma de um arquivo compactado com a extensão *.zip*. Desta forma, o arquivo de entrada de dados é informado conforme ilustrado na figura 6.9.

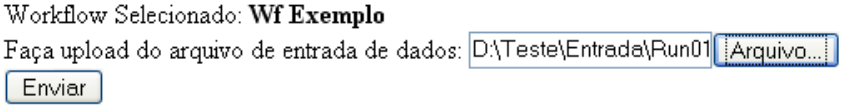

FIG. 6.9: Arquivo de entrada de dados

Solicitada a execução, o workflow exemplo **abstrato** é redefinido em um workflow we **executável**. Neste momento, o módulo de **pré-execução** definirá os locais de armazenamento dos dados intermediários. Trechos do workflow **we** executável (após a execução do módulo de **pré-execução**), na liguagem SCUFL, são mostrados na Figura 6.10.

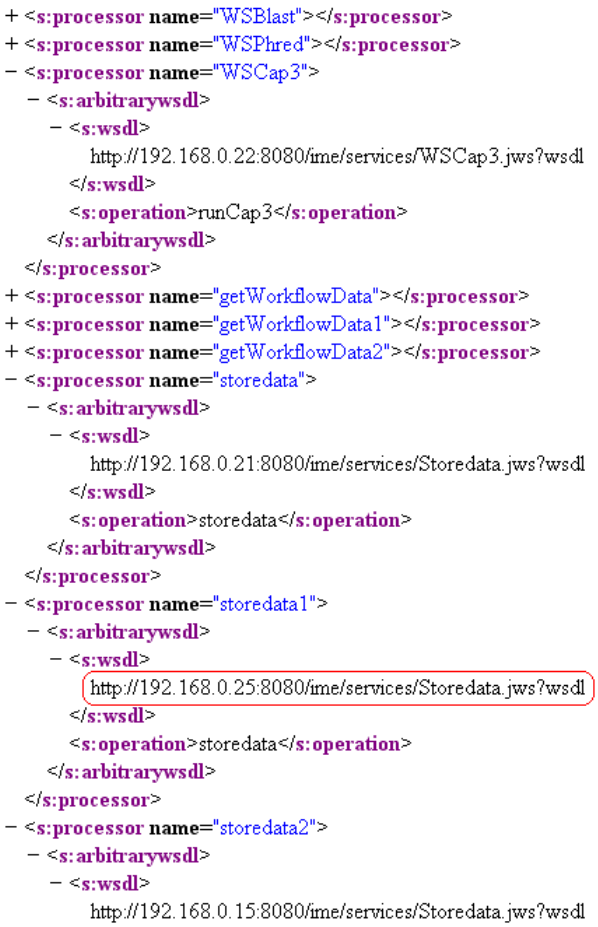

FIG. 6.10: Trecho do código SCUFL do Workflow Exemplo Executável

O serviço Web de armazenamento, destacado na Figura 6.10, mostra a decisão do módulo de **Pré-Execução** em armazenar o dado produzido pela tarefa WSCap3 no servidor 4. Os servidores 3 e 4 hospedam tarefas consumidoras dos dados produzidos pelo serviço Web WSCap3, mas a decisão pelo servidor 4 ocorreu porque **we'** apresenta um maior número de execuções tornando a tarefa a WSInterProScan candidata mais provável para utilizar o dado. Tal decisão evidencia a atuação da heurística, descrita no capítulo 3, seção 3.4. Neste cenário, o módulo de **Pré-Execução** do **e-BioFlow** decidiu persistir o dado de forma a reduzir o tempo gasto na transferência do dado para o processamento de

WSInterProScan, de forma a melhorar uma possível execução futura de **we'**. A figura 6.11 ilustra os fluxos de execução dos workflows **we** e **we'**, bem como o fluxo de transferência e armazenamento de dados.

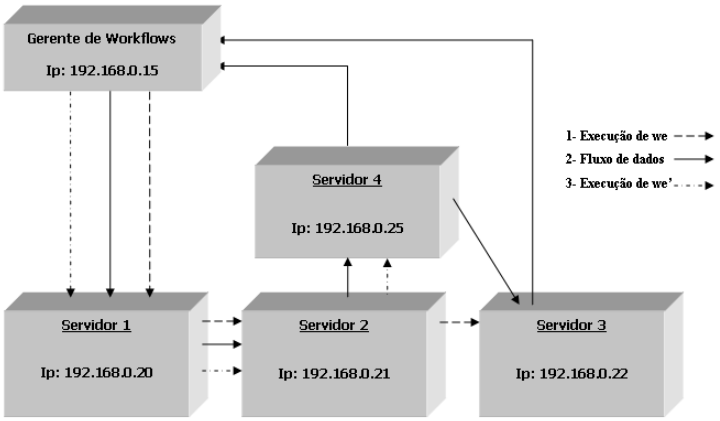

FIG. 6.11: Fluxo de execução e de dados dos workflows we e we'

6.3.5 RE-EXECUÇÃO PARCIAL DO WORKFLOW WE'

Terminada a execução completa de **we**, suponha que o cientista deseje, para comparar os resultados, realizar uma execução do workflow **we'** utilizando a mesma entrada de dados. Como **we'** foi registrado como derivado de **we**, o cientista pode reutilizar os dados produzidos por uma execução anterior de **we**, iniciando uma execução efetiva de **we'** a partir da tarefa em que os workflows se diferenciam. Neste caso, o **e-BioFlow** trata a execução de **we'** como uma re-execução parcial visto que o mesmo será executado a partir de um ponto intermediário do encadeamento com reaproveitamento de dados. O cientista precisa apenas selecionar a instância que deseja reaproveitar os dados e selecionar a tarefa a qual a execução iniciará. Neste caso a tarefa escolhida pelo cientista é a tarefa WSInterProScan (conforme mostrado na figura 6.12), que utilizará o dado produzido pela tarefa WSCap3 que já se encontra armazenado no mesmo servidor 4. O esquema workflow **we'** executável produzido pela re-execução parcial é ilustrado na figura 6.13. Trechos do código SCUFL é mostrado na figura 6.14.

6.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou um exemplo de uso real do **e-BioFlow**. Foram mostradas as principais atividades a serem realizadas para fazer uma execução e re-execução parcial

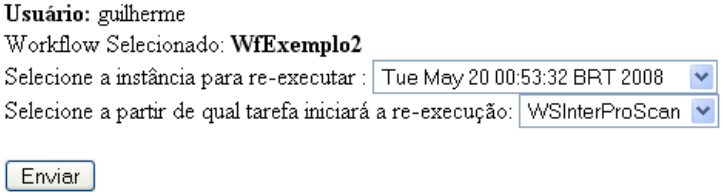

FIG. 6.12: Definindo dados para realizar uma re-execução parcial do workflow we'

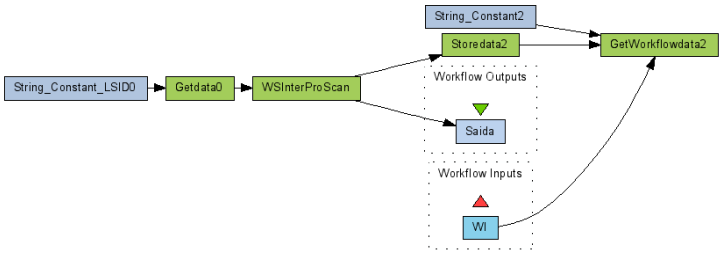

FIG. 6.13: Esquema do workflow we' gerado para re-execução

```
− <s:arbitrarywsdl>
  − <s:wsdl>
      http://192.168.0.15:8080/ime/services/Storedata.jws?wsdl
    </s:wsdl>
    <s:operation>storedata</s:operation>
  </s:arbitrarywsdl>
</s:processor>
− <s:processor name="GetWorkflowdata2">
  − <s:arbitrarywsdl>
    − <s:wsdl>
        http://192.168.0.15:8080/ime/services/Workflowservice.jws?wsdl
      </s:wsdl>
      <s:operation>getWorkflowData</s:operation>
    </s:arbitrarywsdl>
</s:processor>
− <s:processor name="String_Constant2" boring="true">
  − <s:stringconstant>
      http://192.168.0.25:8080/ime/services/WSInterProScan.jws?wsdl
    </s:stringconstant>
</s:processor>
− <s:processor name="String_Constant_LSID0" boring="true">
    <s:stringconstant>urn:lsid:ime.eb.br:1199461433359SA1</s:stringconstant>
</s:processor>
− <s:processor name="Getdata0">
  − <s:arbitrarywsdl>
    − <s:wsdl>
        http://192.168.0.25:8080/ime2/services/Getdata.jws?wsdl
      </s:wsdl>
      <s:operation>getdata</s:operation>
    </s:arbitrarywsdl>
</s:processor>
```

FIG. 6.14: Trecho do código SCUFL gerado para a execução de we'

de workflows. O exemplo foi ilustrado com o uso de dois workflows que diferenciavam-se apenas na última tarefa do encadeamento do workflow, sendo demonstrado um exemplo de execução completa do workflow original (we) e uma re-execução parcial do workflow derivado (we') aproveitando dados da execução do workflow origem.

O exemplo utilizado ilustra a necessidade de registro das diversas execuções dos workflows no ambiente de laboratório, bem como de manter o vínculo entre os worfklows e suas derivações. Além disso, o exemplo evidencia também o benefício que uma re-execução parcial de workflow obteve ao resgatar os dados intermediários do sítio mais próximo, ou seja o sítio onde se inicia a re-execução. Com este exemplo foi possível também demonstrar a habilitação do **e-Bioflow** para dar suporte às necessidades citadas, isto é, execuções, re-execuções, derivações, etc.

No momento da escrita deste trabalho, não foi possível realizar a execução de fato do workflow com uma massa de dados real. Tal impossibilidade ocorreu devido a um problema de integração encontrado, onde a máquina de execução de workflows do Taverna apresentou problemas para a leitura de arquivos binários.

7 CONCLUSÃO

A realização de experimentos *in silico* com o auxílio de workflows científicos, compostos por serviços Web, tem sido gradativamente adotada pela comunidade científica de bioinformática. As tecnologias de serviços Web e internet permitiram a definição e execução de experimentos científicos, em ambientes computacionais heterogêneos e geograficamente diversos. Os serviços Web normalmente disponibilizam a execução de programas científicos e podem ser acessados por uma interface padrão via Web. Assim, workflows compostos por serviços Web estão de fato executando uma série de programas científicos através de invocações desses serviços Web.

Na execução de workflows científicos, a análise dos dados intermediários produzidos é de grande importância para que os cientistas validem os resultados gerados e possam reutilizar os dados em execuções futuras de workflows semelhantes (workflows derivados, por exemplo). O perfil distribuído dos serviços Web que compõem as tarefas dos workflows científicos levantou a questão de como armazenar, adequadamente, os dados intermediários produzidos durante a execução dos workflows. Atualmente, vários sistemas de gerenciamento de workflows científicos têm dado suporte à definição e execução de workflows compostos por serviços Web. Porém, observa-se uma constante na distribuição do processamento e centralização do armazenamento de dados. O armazenamento centralizado pode impactar negativamente a possibilidade de reuso dos dados. Muitas vezes, um serviço Web pode estar localizado num sítio que possua acesso dificultado ao sítio central de armazenamento. Outras vezes, múltiplas re-execuções parciais de um workflow derivado - que são muito freqüentes em ambientes científicos - requerem um tráfego desnecessário e custoso entre os sítios envolvidos, o que impacta o tempo de execução. Adotando-se um paradigma distribuído de armazenamento de dados, torna-se possível aproximar os dados dos sítios que os acessam, reduzindo o tempo gasto para transferir os dados. Nenhum dos trabalhos pesquisados trata, adequadamente, o problema do armazenamento distribuído dos dados intermediários.

Neste trabalho foi proposto uma abordagem cuja arquitetura, denominada **D-BioFlow**, estende as arquiteturas tradicionais de gerenciamento de execução de workflows, possibilitando o gerenciamento distribuído dos dados intermediários. A

abordagem baseia-se no conceito de **workflows abstratos** que são redefinidos em **workflows executáveis**. Os **workflows executáveis** são constituídos, além dos serviços Web físicos de processamento, de serviços de armazenamento de dados que visam persistir o dado em sítios que facilitem um possível reuso futuro da informação, seja em execuções de workflows semelhantes ou numa re-execução do mesmo workflow. As decisões sobre os locais de armazenamento são tomadas através da combinação de metadados, função de custos e heurísticas, que foram propostos neste trabalho.

O protótipo desenvolvido, **e-BioFlow**, baseou-se no uso de tecnologias de distribuição livre. Essas tecnologias envolvem desde o banco de dados utilizado no sistema, passando pela linguagem na qual o mesmo foi implementado, o uso (geração e disponibilização) dos serviços de dados além da máquina de execução de workflows. A implementação do **e-BioFlow** contribuiu para mostrar a viabilidade da arquitetura proposta.

A proposta foi validada através de um ambiente simulado, onde reproduziu-se um ambiente real de produção com múltiplas execuções de diversos workflows. O exemplo de uso realizado permitiu demonstrar a viabilidade da proposta em um workflow real.

7.1 CONTRIBUIÇÕES

A proposta da abordagem para o gerenciamento de dados distribuídos de workflows de bioinformática é a principal contribuição deste trabalho. Deriva da proposta as seguintes contribuições:

- **Elaboração de um metamodelo para gerência de workflows científicos distribuídos:** o metamodelo de dados mapeia o ambiente e auxilia a função de custos, através de seus metadados, a decidir pelos locais de armazenamento;
- **Concepção de uma arquitetura para suportar o gerenciamento distribuído de dados:** a arquitetura apresentada estende a arquitetura tradicional de gerente de workflows apresentando sítios remotos que podem realizar apenas o processamento das tarefas, o armazenamento dos dados, ou ambos;
- **Definição de um modelo de custos de armazenamento:** a função de custos de armazenamento identifica, dentre os servidores do ambiente, o servidor que apresenta a melhor alternativa de armazenamento;

- **Identificação de heurísticas para armazenamento distribuído de dados intermediários:** as heurísticas auxiliam o modelo de custos na decisão pelos locais de armazenamento, identificando possibilidades de reuso futuro dos mesmos;
- **Criação do protótipo e-BioFlow:** o **e-BioFlow** facilita o cientista a executar as tarefas de definição e execução de workflows de bioinformática;
- **Criação de um simulador:** realiza simulações de execuções distribuídas de workflows, contabilizando o tempo gasto na transferência de dados entre as tarefas;

7.2 MELHORIAS E TRABALHOS FUTUROS

Como melhorias a serem desenvolvidas no **e-BioFlow** pode-se destacar a identificação de múltiplos níveis de derivação de workflows. Como apresentado neste trabalho, o **e-BioFlow** identifica apenas o workflow original e seus derivados diretos, mas os workflows derivados podem originar novos workflows derivados, aumentando ainda mais a possibilidade de reuso dos dados. Uma melhoria a ser implementada seria um mecanismo que ajuste os custos de comunicação entre os servidores do ambiente, bem como os custos de armazenamento e recuperação de dados. Uma melhoria relevante seria o desenvolvimento de uma ferramenta que apóie o cientista na construção do arquivo de entrada de dados. Outra melhoria a ser adotada no sistema seria a possibilidade de se trabalhar workflows com tarefas paralelas ou alternativas, além dos workflows seqüenciais.

Outra melhoria importante seria o desenvolvimento de um módulo com recursos para o gerenciamento da execução dos workflows. Com esse recurso os usuários do **e-BioFlow** poderiam dispor de um acompanhamento detalhado dos workflows em execução. Verificar quais workflows ainda não iniciaram sua execução e, ainda, analisar possíveis erros que estejam ocorrendo.

Como trabalho futuro pode-se pensar na implementação do mecanismo de *DataContainer*, aprimorando a reserva de espaço para dados nos sítios remotos de armazenamento. Outra possibilidade seria estender o **e-BioFlow** para funcionar em ambientes de computação em grade. Por fim, outro trabalho poderia ser a evolução do **e-BioFlow** para suportar outras linguagens de definição de workflows, além da SCUFL, e interagir com outras máquinas de execução de workflows.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AALST, W. V. D. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41:639–650, 1999.

ALTINTAS, I., BARNEY, O. e JAEGER-FRANK, E. Provenance collection support in the kepler scientific workflow system. *IPAW*, 2006.

ALTSCHUL, S., MADDEN, T., SCHAFFER, A., ZHANG, J., ZHANG, Z., MILLER, W. e LIPMAN, D. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

BAIÃO, F. A., MATTOSO, M. e ZAVERUCHA, G. A distribution design methodology for object dbms. *Distributed and Parallel Databases*, 16(1):45–90, 2004.

BUNEMAN, P., KHANNA, S. e TAN, W. C. Why and where: A characterization of data provenance. Em *ICDT*, págs. 316–330, 2001.

CALLAHAN, S. P., FREIRE, J., FREIRE, J., SANTOS, E., SCHEIDEGGER, C. E., SILVA, C. T. e VO, H. T. Managing the evolution of dataflows with vistrails. Em *ICDE Workshops*, pág. 71, 2006.

CAVALCANTI, M. C., MATTOSO, M. L. Q. e CAMPOS, M. L. M. Gerência de recursos cientificos: Apoiando a realização de experimentos in silico. Technical report, COPPE/UFRJ, 2003.

CAVALCANTI, M. C., TARGINO, R., BAIÃO, F. A., RÖSSLE, S. C., BISCH, P. M., PIRES, P. F., CAMPOS, M. L. M. e MATTOSO, M. Managing structural genomic workflows using web services. *Data Knowl. Eng.*, 53(1):45–74, 2005.

CLARK, T., MARTIN, S. e LIEFELD, T. Globally distributed object identification for biological knowledgebases. *Briefings in Bioinformatics*, 5(1):59–70, 2004.

DÁVILA, A. M. R., LORENZINI, D. M., MENDES, P. N., SATAKE, T. S., SOUSA, G. R., CAMPOS, L. M., MAZZONI, C. J., WAGNER, G., PIRES, P. F., GRISARD, E. C., CAVALCANTI, M. C. R. e CAMPOS, M. L. M. Garsa: genomic analysis resources for sequence annotation. *Bioinformatics*, 21(23):4302–4303, 2005.

EDDY, S. Hmmer - profile hidden markov models for biological sequence analysis version 2.3.2. Technical report, Howard Hughes Medical Institute and Dept. of Genetics Washington University School of Medicine, 2003.

EWING, B. e GREEN, P. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome Res*, 8:186194, 1998a.

EWING, B., HILLIER, L., WENDL, M. e GREEN, P. Base-calling of automated sequencer traces using phred. i. accuracy assessment. *Genome Res*, 8:175185, 1998b.

FAHRINGER, T., PRODAN, R., DUAN, R., NERIERI, F., PODLIPNIG, S., QIN, J., SIDDIQUI, M., TRUONG, H. L., VILLAZÓN, A. e WIECZOREK, M. Askalon: a grid application development and computing environment. Em *GRID*, págs. 122–131, 2005.

FOSTER, I. T., VÖCKLER, J.-S., WILDE, M. e ZHAO, Y. The virtual data grid: A new model and architecture for data-intensive collaboration. Em *CIDR*, 2003.

GAMMA, E., R. HELM, R. J. e VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1994.

HUANG, X. e MADAN, A. Cap3: A dna sequence assembly program. . *Genome Res*, 9: 868877, 1999.

LEMOS, M. e CASANOVA, M. A. On the complexity of process pipeline scheduling. *Simpósio Brasileiro de Banco de Dados*, págs. 57–71, 2006.

LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., HIGGINS, D., JAEGER, E., JONES, M., LEE, E. A., TAO, J. e ZHAO, Y. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

MEDEIROS, C., VOSSEN, G., e WESKE, M. Wasa: A workflow-based architecture to support scientific database applications extended abstract. *6th DEXA Conference*, págs. 574–583, 1995.

MULDER, N., APWEILER, R., ATTWOOD, T., BAIROCH, A., BATEMAN, A. e BINNS, D. Interpro, progress and status in 2005. *Nucleic Acids Res*, 33:D201D205., 2005.

OINN, T. M., ADDIS, M., FERRIS, J., MARVIN, D., SENGER, M., GREENWOOD, R. M., CARVER, T., GLOVER, K., POCOCK, M. R., WIPAT, A. e LI, P. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

OSZU, M. T. e VALDURIEZ, P. *Principles of Distributed Database Systems*. Prentice-Hall, 1999.

RUBERG, G., BAIÃO, F. A. e MATTOSO, M. Estimating costs of path expression evaluation in distributed object databases. Em *DEXA*, págs. 351–360, 2002.

SILVA, F., CAVALCANTI, M. e DÁVILA, A. Managing structural genomic workflows using web services. *17th International Workshop on Database and Expert Systems Applications DEXA 2006*, págs. 206–210, 2006.

TAYLOR, I., SHIELDS, M., WANG, I. e HARRISON, A. Visual Grid Workflow in Triana. *Journal of Grid Computing*, 3(3-4):153–169, September 2005.

WAGNER, G., SORIANO, K., JUCÁ, H., BELLOZE, K. T., TSCHOEKE, D. A., GERONIMO, G. A., ALBRECHT, F., MATTOSO, M. L. Q., CAVALCANTI, M. C., CAMPOS, M. L. M., GRISARD, E. e DÁVILA, A. M. R. Stingray: System for integrated genomic resources and analysis. 2007.

WILDEMBERG, M., DE PAULA, M. M. V., BAIÃO, F. A. e MATTOSO, M. Alocação de dados em bancos de dados distribuídos. Em *Simpósio Brasileiro de Banco de Dados*, págs. 215–228, 2003.

9 ANEXOS

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo