Gerenciadores de Dados Biológicos: Genéricos ou Ad-hoc?

Sérgio Lifschitz

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

sergio@inf.puc-rio.br

Resumo. *A necessidade de eficiência no gerenciamento de dados biológicos traz novos rumos e desafios para as pesquisas em sistemas de bancos de dados. Neste artigo discute-se a oportunidade de construção de gerenciadores de dados específicos para a área de biologia computacional e afins. São necessárias muitas adaptações para que o uso de sistemas relacionais ou baseados em modelos de dados conhecidos seja eficaz. Nesse artigo discute-se a viabilidade de um SGBD ad-hoc para dar suporte adequado à gestão e acesso de dados biológicos. São apresentadas as abordagens correntes baseadas em extensões de SGBDs existentes e discute-se algumas das estratégias particulares, propostas ou em desenvolvimento, para lidar com esta nova área de aplicação.*

1. Introdução e Motivação

Um Sistema de Gerência de Bancos de Dados (SGBD) permite o controle de grandes volumes de dados, oferecendo persistência em memória estável, e garantindo o acesso de múltiplos usuários com eficiência e segurança. Os SGBDs normalmente implementam um determinado modelo de dados, que define tanto uma estrutura de representação para os dados como também o formalismo de manipulação. Há vários anos o modelo relacional de dados, baseado em relações matemáticas, comumente abstraídas como tabelas, vem sendo utilizado como modelo padrão de mercado [Silberschatz et al. 2005]. Para aplicações ditas convencionais, envolvendo administrações empresariais ou coorporativas, os SGBDs relacionais disponíveis no mercado atendem os requisitos de dados e funcionais sem maiores dificuldades.

Entretanto, novas aplicações trazem desafios de estruturação de dados nos diversos níveis de memória, especificação de consultas e manipulações complexas, sobre conjuntos de dados cada vez maiores. Esse é o caso das aplicações científicas (*life sciences*), compreendendo objetos geográficos e dados semi-estruturados, *petabytes* de dados e processamentos complexos [Jagadish and Olken 2004, Bell et al. 2006, Sinha et al. 2007]. Em vários casos, como por exemplo, sistemas compreendendo imagens ou dados espaciais de forma geral, é difícil utilizar SGBDs convencionais ou mesmo suas versões com funcionalidades estendidas. Por exemplo, abstrair mapas para armazenamento em tabelas de SGBDs relacionais exige muitas adaptações e simplificações que podem comprometer a semântica da aplicação. Cabe observar que, mesmo assim, muitos dos sistemas correntes

utilizam os SGBDs atualmente disponíveis ou mesmo sistemas de arquivos tradicionais para gerência dos dados [Topaloglou et al. 2004].

Assim, visando o desenvolvimento de sistemas eficazes, robustos e eficientes, pode-se pensar em duas abordagens básicas para sistemas de bancos de dados que venham dar suporte para essas novas áreas de aplicação:

1. estender e usar os gerenciadores existentes e adaptá-los para os tipos de dados e manipulações necessárias; ou
2. construir um sistema *ad-hoc* que atenda as particularidades da aplicação.

Há prós e contras nas duas abordagens. Pode-se aproveitar a base dos sistemas já existentes e, com custo relativamente baixo, obter-se uma solução em curto prazo, porém com algumas restrições operacionais. Há também o problema da utilização de sistemas de propósito geral, que envolvem vários módulos e componentes, muitos dos quais desnecessários para uma aplicação particular. Já as soluções projetadas especificamente para o problema em questão por construção atendem à demanda específica porém têm custo alto e pouca abrangência, conseqüentemente, não serem economicamente viáveis.

A alternativa de um sistema especialista, dedicado já é adotada em outras área de aplicação conhecidas, como é o caso dos SIG: Sistemas de Informação Geográficas (e.g. [Neteler and Mitásová 2002]). Por outro lado, alguns SGBDs comerciais vêm há muito tempo incorporando funcionalidades específicas. É o caso das novas bibliotecas e APIs que viabilizam soluções internas do SGBD para mineração e armazém de dados (e.g. [Poess and Othayoth 2005]).

Neste artigo estamos particularmente interessados com o contexto de aplicações científicas, em particular, a bioinformática ou biologia computacional. Sabe-se que há projetos de pesquisa que visam seqüenciar o genoma de várias espécies. As seqüências obtidas são normalmente representadas por longas cadeias de caracteres que, por sua vez, têm sido armazenadas em ”bancos de dados”, não necessariamente controlados por um SGBD. Devido ao grande interesse público e às novas tecnologias sendo disponibilizadas, o volume de dados vem aumentando consideravelmente. Experimentos com *micro-arrays* vêm gerando dados da ordem de petabytes, tornando obrigatória a busca por soluções de bancos de dados apropriadas.

As duas áreas, de computação e biológica, podem ganhar muito com a integração de tecnologias. De fato, técnicas para gerenciamento de dados possuem um papel fundamental para o desenvolvimento de aplicações biológicas pois fornecem abstrações adequadas para projetar, acessar e armazenar os dados. Por outro lado, as abstrações da áreas biológica podem trazer idéias novas de abordagem na solução de problemas computacionais (computação bio-inspirada), como é o caso da computação por DNA [Adleman 1994].

Embora sistemas de bancos de dados propriamente ditos ainda sejam pouco utilizados na prática para a manipulação de bancos de dados biológicos, alguns trabal-

hos na literatura já vêm investigando estruturas de índices [Hunt et al. 2002], integração de fontes de dados e aplicações [Seibel 2002], persistência para dados ditos científicos [Buneman et al. 2004], linguagens de acesso aos dados [Tata et al. 2006], processamento de consultas [Chen et al. 2005] e sistemas de *workflow* [Lemos 2004], apenas para citar alguns.

Neste trabalho coloca-se em discussão a viabilidade de um Sistema Gerenciador de Bancos de Dados (SGBD) específico para aplicações de biologia computacional. A maioria das ferramentas existentes ora acessa dados diretamente de arquivos textos ou binários, sem a utilização de um gerenciador apropriado, ora faz uso de SGBDs relacionais comerciais que necessitam de muita adaptação e capacidade de abstração. De certa forma cria-se obstáculos que impedem as aplicações de se beneficiar de mecanismos eficazes de armazenamento, acesso eficiente a disco e gerenciamento inteligente da memória, entre outros.

Assim, pretende-se listar aqui alguns dos principais tópicos de pesquisa no domínio de bancos de dados relacionados direta ou indiretamente às aplicações em biologia computacional. Em particular, discute-se indiretamente a idéia de construção de um "Bio-SGBD", um gerenciador de bancos de dados *ad-hoc* voltado especialmente para lidar com dados biológicos. Na próxima Seção são apresentadas algumas das características de SGBDs e como estas vem sendo consideradas na literatura e nos novos sistemas em desenvolvimento. Em seguida, na Seção 3, questões mais específicas ao contexto de aplicações da bioinformática e bancos de dados são explicitadas com algumas soluções ad-hoc existentes ou em preparação. O artigo termina na Seção 4 com conclusões e comentários finais.

2. Abordagens de Gestão de Dados Biológicos

Algo relevante no contexto desse artigo envolve lidar com sequências biológicas, representadas por cadeias de caracteres de tamanhos variados, que podem ser agrupadas também distintamente, em função da semântica associada e da manipulação relacionada com os programas de acesso.

A análise através da comparação de seqüências por similaridade tornou-se uma das operações mais importantes na biologia computacional, cujos resultados dão origem a novos tipos de dados biológicos, como as anotações, ou ainda são entradas para muitas outras operações mais elaboradas, como a busca de padrões.

É de fundamental importância mencionar a família de programas BLAST [Altschul et al. 1990] - Basic Local Alignment Search Tool - que realizam o alinhamento e a comparação entre biosseqüências. Estes programas são na verdade baseados em heurísticas, o que trouxe uma grande melhora nos tempos de respostas em relação aos algoritmos exatos propostos inicialmente. O BLAST é bastante utilizado e, por esta razão, melhorias nas estratégias de execução são muito importantes. No contexto particular deste artigo, considera-se vários aspectos relevantes, como por exemplo:

- estruturas de persistência das sequências do banco de dados;
- índices que reduzam o espaço de busca e facilitem a análise dos dados;
- a política de substituição de páginas de dados em memória;
- o escalonamento adequado dos processos; e
- a possibilidade de distribuição e alternativas de alocação dos dados.

A interpretação dos dados obtidos experimentalmente, onde se busca gerar conhecimento biológico, é outra atividade essencial. São registradas anotações manuais, e também automáticas, resultantes da execução de programas específicos de análise. Métodos de mineração de dados e descoberta de conhecimento podem ser aplicados a grandes volumes de dados biológicos. Há de se preparar também diferentes visões dos dados, diversos tipos de relatórios e interfaces de consulta flexíveis e amigáveis.

Uma das necessidades básicas em bioinformática está relacionada com a capacidade de integração de dados obtidos em bases (ou fontes) de dados públicas ou privadas, visando comparações e análises mais precisas de uma determinada pesquisa. Em muitos casos estas informações são produzidas por um único laboratório e suas informações são únicas, isto é, não estão replicadas nos repositórios públicos. Todas essas fontes de dados relevantes para os pesquisadores contêm informações biológicas (ex.: homologias, estrutura e similaridades) e afins (ex.: anotações relevantes e artigos científicos).

Em sua maioria estas fontes de dados diferem na forma de armazenamento de dados e nas informações relevantes à pesquisa. Também estão associadas a aplicativos que diferem nos serviços oferecidos: de visualização dos dados, de busca, de alinhamentos, de comparação de seqüências, entre outros. Há implementações de sistemas que armazenam informações biológicas em arquivos texto (e.g. GenBank), em bancos de dados relacionais (e.g. Swiss-Prot) e em sistemas orientados a objetos persistentes (e.g. AceDB). No entanto, como a pesquisa na área está em constante evolução, há a necessidade de alteração dos esquemas já implantados, sugerindo a adoção de modelos de dados mais flexíveis. Isto porque novas informações biológicas surgem a cada momento e é fundamental que elas possam ser representadas nas fontes de dados existentes.

Existem diferentes abordagens na literatura para tratar o problema de integração de informações de fontes de dados distribuídas e heterogêneas. A primeira trata da integração de bancos de dados através de um Sistema Gerenciador de Bancos de Dados Distribuídos e Heterogêneos (SGBDH). Uma segunda alternativa lida com múltiplas fontes de dados com uma abordagem *Multidatabases*. Outra idéia seria usar a tecnologia de *Data Warehouses*, que implementa uma visão materializada do esquema das fontes de dados componentes da integração.

A abordagem SGBDH pode não ser adequada para integração de dados da biologia molecular pois muitas fontes de dados são arquivos texto, com pouca estruturação. Também não há esquema global, padronizado e aceito pelos produtores de dados. Já com *Multidatabases* há o problema usual de indisponibilidade dos dados em momentos de acesso e a falta de controle sobre o desempenho em ambientes distribuídos. Por fim,

o uso de *Data Warehouses* é o mais promissor porém também pode não ser adequado quando há atualização frequente de esquemas e das instâncias de dados.

A integração de fontes de dados e aplicações para a área de bioinformática ainda é considerado um problema em aberto e com vários desafios importantes na prática [Topaloglou et al. 2004]. Alguns se referem à persistência dos dados após a efetivação da integração, e posterior acesso eficiente para análise e manipulação.

Já existem algumas boas iniciativas para lidar com dados biológicos, em particular sequências e respectivas anotações. São várias as propostas para representar esses dados em XML porém é necessário considerar a eficiência do armazenamento e acesso posteriores. Há propostas interessantes de repositórios XML (e.g [Wong et al. 2000]) que levam em consideração aspectos de armazenamento físico para as anotações e sequências em XML. Em particular, vale mencionar a idéia de catálogo ou dicionário, presente nos SGBDs tradicionais, onde se descrevem a estrutura completa dos dados em XML, os tipos de dados e também algumas estatísticas, estendendo consideravelmente a noção de DTD para o caso particular.

Outros trabalhos de pesquisa, envolvendo ou não o desenvolvimento de protótipos, envolvem a extensão de sistemas baseados no modelo relacional, tanto para persistência como para acesso. Na prática considera-se tabelas para armazenar os dados e adaptações na linguagem de acesso (como a SQL) para produzir os resultados particulares desejados. Esta abordagem é adotada por exemplo em [Tata et al. 2006], com extensões sendo propostas para o SGBD PostgreSQL. Já em [Eltabakh et al. 2007] há uma abordagem semelhante, com ênfase particular na gerência de anotações. Os usuários podem incluir os dados de anotações de forma transparente em relação à forma como as anotações são persistidas. Também é oferecida diferentes granularidades para anotação, por tuplas inteiras ou atributos únicos de determinadas tuplas, ou mesmo múltiplas colunas da tabela.

Neste caso, há a proposta de estender SQL com comandos de criação de tabelas de anotação, que têm propostas de armazenamento específicas, incluindo índices de busca sobre as anotações. Considera-se também um suporte para arquivamento de anotações que permita recuperar dados antigos rapidamente, de forma análoga à uma gestão de versões. Esta gestão de anotações viabiliza também um controle de proveniência de dados, que pode ser especificado no momento de uma consulta aos dados. Em [Eltabakh et al. 2007] há proposta de estruturas de índices para dados biológicos compactadas que podem ser usadas posteriormente sem necessidade de descompactação.

Utilização de SGBDs

Alguns dos bancos de dados biológicos utilizam sistemas baseados no modelo relacional, sistemas orientados a objetos ou ainda alguns gerenciadores específicos. No EMBL, SwissProt, TrEMBL, o SGBD Oracle foi adotado como repositório mas são utilizados arquivos texto para troca de dados. Na maioria dos casos onde gerenciadores de banco de dados relacionais são utilizados como repositórios de seqüências (DNA ou proteínas),

estas são armazenadas como cadeias de caracteres. Arquivos texto são diretamente persistidos como objetos grandes, por exemplo, do tipo *CLOB*. Isto facilita a carga dos repositórios a partir de arquivos texto, porém o acesso aos dados é limitado aos operadores tradicionais.

O SGBD Oracle versão 10g merece uma atenção em particular devido ao grande número de funcionalidades incluídas para facilitar o trabalho dos pesquisadores na área de ciências da vida, principalmente no sentido de acesso a dados provenientes de diversas fontes, algo muito comum na biologia computacional. O Oracle 10g incorpora o BLAST como função. Também consultas em linguagem SQL para pré-filtrar as seqüências ou ainda pós-processar os resultados obtidos são permitidas.

Outra estratégia semelhante foi colocada em prática utilizando o gerenciador de dados PostgreSQL. Esta implementação, chamada de BlastGres [Hsiao et al. 2005], também incorpora o programa BLAST ao gerenciador de banco de dados. Além disto, foram criados novos tipos de dados para representar segmentos de seqüência, em conjunto com um novo tipo de índice para acelerar o acesso a uma região de uma seqüência e as propriedades correspondentes. No entanto, nesta abordagem as seqüências são guardadas como cadeias de caracteres. A idéia é representar um segmento de seqüência com o tipo de dados *range*, números indicando o inicio e fim de um segmento de seqüência, e um tipo *location*, onde um valor de *range* é associado ao identificador de uma seqüência, podendo ainda utilizar um índice para realizar a busca de uma região de uma seqüência dada - o atributo do tipo *location*.

Outro exemplo de sucesso e bastante usado é o sistema ACeDB. O ACeDB está baseado no modelo orientado a objetos mas conta com um módulo de gerenciamento de banco de dados, em um modelo flexível, projetado especificamente para manipular informações biológicas. No ACeDB (A *Caenorhabditis elegans* Data Base) são armazenados os resultados de projetos de sequenciamento e mapeamento de larga escala.

O ACeDB representa internamente os dados em forma de árvore, em formato binário. A entrada e saída dos dados é feita via arquivos texto denominados *ACE files*, onde as informações são representadas de acordo com uma sintaxe específica, semelhante à XML. A base do AceDB é utilizada para outras pesquisas, como é o caso do TcruziDB, coordenado pelo DBBM da FIOCRUZ, Rio de Janeiro.

3. Algumas Soluções Ad-hoc

Como já visto anteriormente, a família de programas BLAST é a mais utilizada pelos pesquisadores e existem diversos sítios WWW que disponibilizam os programas para os usuários, podendo ocorrer diversos acessos simultâneos. Os programas de comparação de biosseqüências objetivam ser ao mesmo tempo rápidos e confiáveis. Como são largamente usados, mesmo pequenas melhorias nestes podem trazer grandes benefícios.

Uma idéia seria a inclusão de um gerenciamento de *buffer* adequado, tornando mais rápidos seus tempos de execução. Uma estratégia de gerenciamento de *buffer*

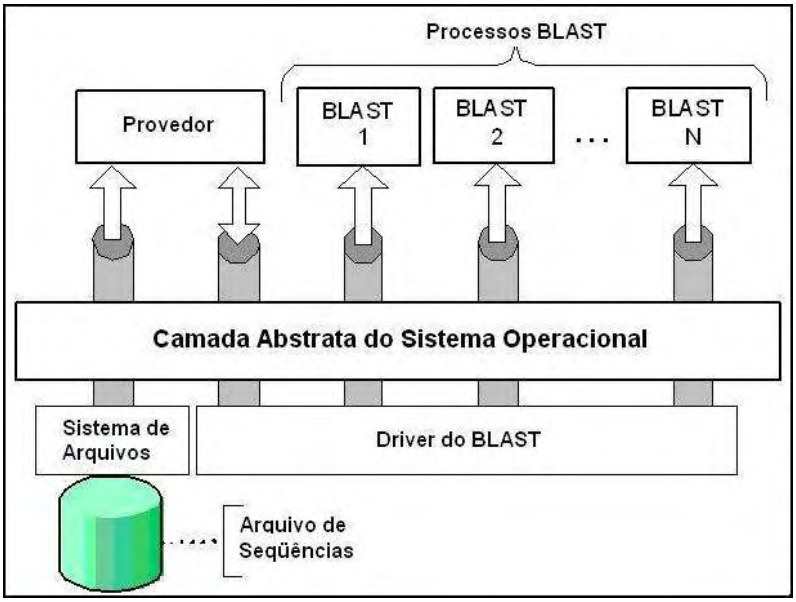

Figura 1. Driver para Gerência de Buffer

para o BLAST foi publicada em [Lemos and Lifschitz 2003], sugerindo o uso de estruturas de armazenamento de seqüências na memória denominadas anéis. Já em [Lifschitz and Mauro 2005], sugere-se a implementação de uma arquitetura de *driver* para realizar este gerenciamento de *buffer*. Outros programas podem ser beneficiados por essas técnicas.

Há dois modos de implementar o gerenciamento de *buffer* para o BLAST, de maneira intrusiva no código e de maneira não-intrusiva. A maneira intrusiva de implementar é através da substituição, no código, de cada chamada às funções de leitura de seqüências por outras funções que se comunicam com um processo que irá realizar o gerenciamento de buffer. A maneira não-intrusiva de implementar é não modificando o código do BLAST, o que pode ser feito através da criação de um *driver* que simule o funcionamento dos arquivos do banco de dados e realize ao mesmo tempo o gerenciamento de *buffer*.

Aqui será descrito brevemente como se dá a implementação de maneira não-intrusiva através de um *driver* [de Noronha 2006]. Este é um programa que possibilita a comunicação de aplicativos com dispositivos de *hardware* e *software*, escondendo a maneira como é realizada a comunicação direta com os mesmos. O uso de um *driver* para implementar o gerenciamento de *buffer* para o BLAST traz grandes vantagens, pois ele não exige modificações no código fonte e pode ser utilizado para diferentes versões do BLAST com poucas modificações (ou nenhuma) no código do *driver*.

No caso de sistemas operacionais Linux, os drivers devem ser implementados como módulos do kernel. A Figura 1 mostra a arquitetura do driver para o BLAST.

Devido às limitações existentes para implementações de módulos do kernel, optou-se por criar um processo provedor, com o qual o processo do driver irá se comunicar, que acessará os arquivos do banco de dados e fará o gerenciamento de buffer. Como pode ser visto na Figura 1, os processos BLAST realizarão leituras dos falsos arquivos do banco de dados, executando a função de leitura do driver, e informando deste modo quais páginas desejam ler do banco de dados.

O driver irá comunicar-se com o processo provedor, enviando as novas requisições. O processo provedor possuirá os anéis de buffer, além das posições do arquivo de seqüências a partir das quais cada processo iniciou a leitura. Estas posições serão usadas para o cálculo da posição real do arquivo que será lida por cada processo quando este requisitar novas páginas, somando-se a posição da página requisitada com a posição a partir da qual o processo iniciou a leitura. O processo provedor poderá acessar o banco de dados diretamente, e fornecerá ao driver as seqüências desejadas, lidas do banco ou dos anéis em memória, ao executar a função de escrita do driver. Finalmente, ao receber os dados do provedor, o driver os envia aos processos.

Cabe observar que a política de escalonamento do processo provedor pode ser adaptada da maneira que a aplicação que solicita sequências desejar. No caso de atendimento aos processos BLAST, há a preocupação com a ordem sequencial de leitura e cálculo de posições reais e virtuais nos arquivos de dados. Para outros programas, este inteligência de provimento de sequências pode ser modificada, mantendo-se a estratégia não intrusiva de lidar com as aplicações e os dados. No que diz respeito aos trabalhos em andamento e futuros, podemos mencionar estudos de desempenho específicos de técnicas de compactação de dados [Rosa 2006] aplicadas na execução do BLAST em conjunto com a ferramenta BioProvider [de Noronha 2006].

Distribuição e Paralelismo de E/S

Como alternativa para obtenção de melhor desempenho, a utilização de ambientes paralelos para a resolução de problemas complexos têm sido amplamente estudada. Os avanços nas tecnologias de microcomputadores e de redes fez com que o emprego de máquinas de arquitetura de memória distribuída se tornasse viável. Esse é o caso dos *clusters de PCs*, que têm sido cada vez mais utilizados, pois apresentam uma relação *custo vs. desempenho* bastante atrativa [de Carvalho Costa and Lifschitz 2003].

Para a execução do BLAST em máquinas de memória distribuída, uma primeira abordagem consiste na replicação da base de dados em todos os nós. Para este caso, apresentamos um esquema mestre-escravo, conforme apresentado na Figura 2(a). As requisições (seqüências para comparação com a base de dados) são submetidas ao nó que desempenha o papel de mestre. Cada requisição recebida pelo nó mestre pode conter uma ou mais seqüências. Em cada nó escravo existirá uma réplica de toda a base de dados. Além disso, os nós escravos terão o algoritmo BLAST devidamente implementado e configurado, pronto para execução. Assim, cada nó escravo terá a possibilidade de executar a "operação de BLAST" sobre a base de dados local, independentemente dos

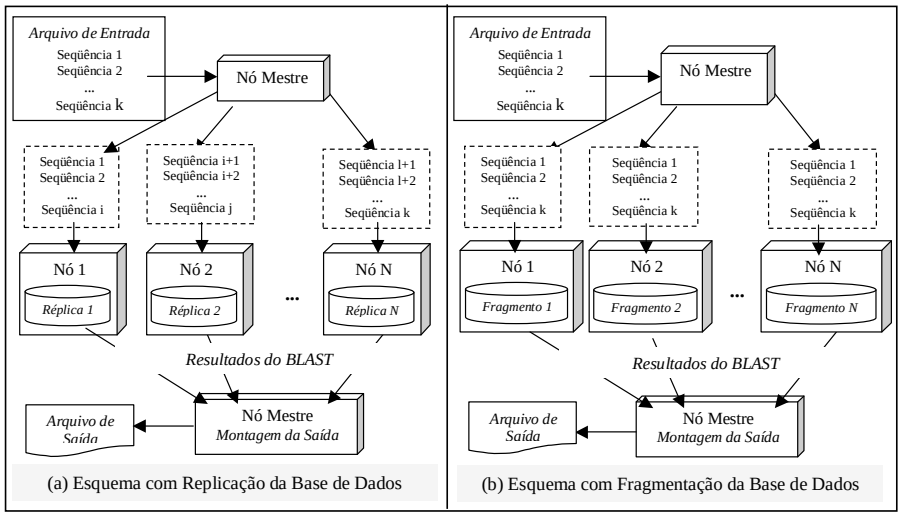

Figura 2. Esquemas de Distribuição e Alocação de Dados

outros.

O nó mestre deve distribuir as seqüências recebidas pelos diversos nós, alocando, assim, tarefas aos nós escravos. Esta alocação deve fazer com que a carga de trabalho de cada um dos nós escravos seja semelhante. Para a distribuição de tarefas, adotaremos, inicialmente uma estratégia *circular* semelhante a tradicional. Uma tarefa a ser atribuída a um nó escravo é a de realizar a comparação de uma dada seqüência com a base de dados. Após a distribuição das seqüências pelos nós escravos estes devem executar o BLAST para cada uma das seqüências recebidas. Vários resultados serão gerados. Cada um deles deverá ser remetido ao nó mestre. Este receberá todos os resultados e montará um resultado único para a solicitação recebida.

Para a execução do BLAST, considerando um particionamento da base de dados com a devida alocação dos fragmentos pelos diversos nós, é proposto, também, um esquema mestre-escravo, apresentado na Figura 2(b). O nó mestre é o responsável por receber as requisições e encaminhá-las aos nós escravos, os quais realizam o BLAST. Porém, nesta etapa, cada nó escravo conterá somente uma parte da base de dados e não toda ela, como no caso anterior. A principal dificuldade desta abordagem reside, justamente, na geração dos fragmentos.

Na distribuição das tarefas surge outra diferença entre a estratégia replicada e a fragmentada. Na estratégia fragmentada, para completitude do resultado final, cada seqüência submetida ao nó mestre deve ser repassada a todos os nós escravos, para que o BLAST seja executado sobre toda a base de dados original. Após a realização do BLAST em cada nó, os resultados são enviados para o nó mestre. Este é responsável pela montagem do resultado final e apresentação ao usuário. A montagem do resultado final é mais uma etapa mais complexa para o caso com fragmentação da base de dados do

que para o caso com replicação pois, no esquema com fragmentação, existirão para cada seqüência submetida, n resultados de BLAST, onde n vale o número de fragmentos da base de dados.

Vários testes foram realizados em um cluster de até 32 nós, com três bases de dados de tamanhos variados: *Ecoli.aa*, *SwissProt* e *nr*. Maiores detalhes podem ser obtidos em [de Carvalho Costa 2002].

São vários os fatores que influenciam no desempenho da implantação do algoritmo BLAST em máquinas de memória distribuída. O desvio de carga foi apresentado como um dos problemas a ser resolvido. Características como similaridade das seqüências de entrada com as da base de dados, comprimento das seqüências de entrada e das formadoras de fragmentos da bases de dados, entre outras, mostraram-se causadoras de desvios.

Diferentes propostas de correção para os desvios podem ser consideradas em função da forma de distribuição dos dados considerada. A realização de alterações na política de distribuição de dados aos nós é uma possível solução para este problema. Estão sendo avaliadas também técnicas de particionamento físico de dados porém com alocação replicada e controle de fragmentos primários por nó de processamento [Sousa 2007]. Neste caso, testes de políticas de distribuição de dados sob demanda e adaptativa tem obtido resultados promissores. Neste trabalho também se contempla o problema de tolerância à falta e confiabilidade, de forma análoga ao protocolo WAL (*write-ahead logging*) utilizado pelos SGBD como mecanismo de recuperação transacional. Utilizando estratégia semelhante o trabalho [Carvalho et al. 2005] permite que execuções da ferramenta BLAST em ambientes distribuídos ou mesmo em grade não tenham de ser reiniciados em caso de falha em alguma máquina participante da operação.

Integração de Fontes de Dados e Aplicações

As limitações das abordagens de integração mencionadas motivaram a proposta da ferramenta Bio-AXS [Seibel and Lifschitz 2001], que utiliza uma arquitetura baseada em um *framework* orientado a objetos, visando prover flexibilidade e extensibilidade. Este *framework* inova ao tratar da integração de esquemas baseada em um meta-modelo. A integração é feita através de um mediador que captura os esquemas e dados das fontes, faz as conversões necessárias e materializa as informações no repositório.

Bio-AXS (Figura 3) se propõe a integrar dados de qualquer fonte de dados de biologia molecular. Tanto os esquemas como os dados são armazenados em um repositório que utiliza o modelo de dados semi-estruturado. Assim, os esquemas são armazenados em XMLSchema e os dados em XML.

A arquitetura do *framework* está subdividida em quatro módulos, cuja interdependência está esquematizada na Figura 3. O módulo *Administrador* realiza a interface com os usuários, de forma a prover as seguintes funções: permitir gerenciar o modelo da biologia e esquemas específicos, solicitar a captura de esquemas e/ou de dados, permitir a formulação de consultas, permitir a execução dos programas instanciados no próprio

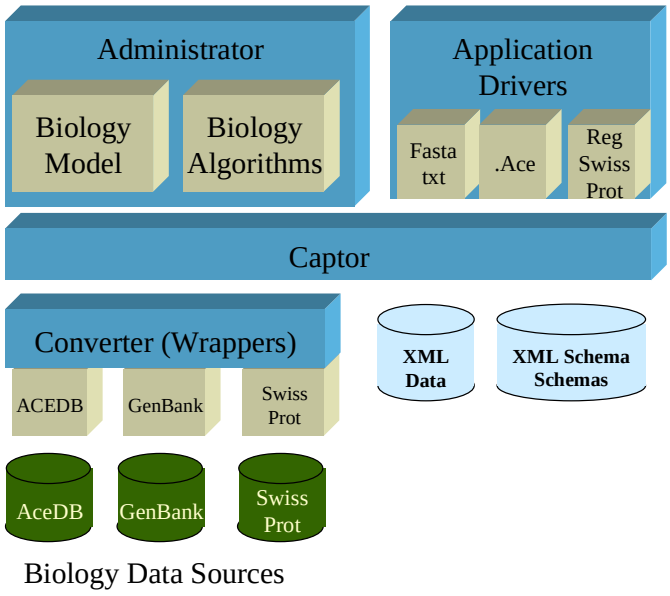

Figura 3. Arquitetura Bio-AXS de integração

framework e a execução de aplicações externas. Este módulo contém um repositório, que armazena o modelo global da biologia. O módulo *Capturador* administra o repositório de dados e de esquemas da arquitetura. Os *Wrappers* implementam o acesso às fontes de dados de biologia, efetuando a tradução dos esquemas das fontes de dados para XML Schema e dos dados para XML. Os *Drivers* implementam a conversão dos dados do formato interno do framework (XML) para o formato esperado pelas aplicações. Os detalhes da arquitetura de cada módulo, suas classes e métodos, e seus relacionamentos, são descritos em [Seibel 2002].

Fruto de resultados já disponíveis, cabe aqui relatar que a arquitetura da ferramenta Bio-AXS está na base de uma outra ferramenta de anotação específica chamada Bio-Notes [Lemos et al. 2003]. Esta vem sendo utilizada anotações do genoma do Trypanosoma Cruzi (FIOCRUZ) e para o sistema de anotação do genoma da Gluconacetobacter diazotrophicus (UFRJ Bioquímica).

Linguagem de Modelagem de Dados

Para que a solução de integração baseada em frameworks possa ser aplicada ao domínio da biologia molecular é necessário dispor de um modelo conceitual de dados [Seibel et al. 2003]. Este é obtido por meio de um processo de projeto, como descrito anteriormente, que enriquece a comunicação entre o projetista e o especialista do domínio, facilitando futuras mudanças na aplicação ou na implementação do banco de dados.

O resultado do processo do projeto conceitual de banco de dados é um esquema conceitual de dados. São pelo menos dois os objetivos dos modelos conceituais. Primeiro,

os modelos conceituais são usados para descrever a informação a ser manipulada por um sistema de informação. Segundo, os esquemas conceituais são traduzidos em esquemas de dados lógicos que serão usados para implementar um banco de dados.

Em [de Macêdo et al. 2007] são listados alguns requisitos para modelagem conceitual de dados, em particular para a biologia molecular. Esses requisitos são baseados nos problemas encontrados utilizando-se linguagens de modelagem conceitual de dados tradicionais, como ER e UML.

Assim, é sugerida em [de Macêdo 2005] a construção de uma nova linguagem de modelagem, chamada BioConceptual, projetada para atender os requisitos da Tabela **??**. O objetivo é oferecer ao projetista do banco de dados uma linguagem mais expressiva para projetar esquemas de dados para aplicações biológicas facilitando a sua construção. A BioConceptual tenta trazer o modelo conceitual de dados mais perto do domínio da biologia, sem detalhar aspectos de implementação.

Após analisar diversos paradigmas de modelagem, optou-se por usar um modelo orientado a objetos como base para a BioConceptual. São dois os maiores benefícios no caso:

- Um modelo orientado a objetos tem bastante expressividade para especificar representações complexas e permite a unificação do desenvolvimento da aplicação e do banco de dados em um ambiente sem divisões entre o banco de dados e a linguagem de programação;
- As aplicações requerem menos código, usam mais naturalmente a modelagem de dados, e os códigos são mais fáceis de manter. Desenvolvedores de sistemas orientados a objeto podem criar aplicações completas de banco de dados sem muito esforço.

Como conseqüência desta decisão, foi decidido usar e estender a especificação do padrão ODMG 3.0para incorporar as necessidades da nova linguagem. A adoção do padrão ODMG facilita a compreensão do modelo e a compatibilidade com outras linguagens de modelagem orientadas a objetos como UML, linguagens de programação usando objetos (ex. Java), bancos de dados objeto-relacional e bancos de dados puramente orientados a objetos. Mais detalhes da proposta BioConceptual devem ser consultados em [de Macêdo 2005].

4. Comentários Finais

A eficácia e eficiência no gerenciamento dos dados vem trazendo novos rumos e aplicações para as pesquisas em sistemas de bancos de dados. Estes envolvem informações simples (e.g anotações textuais) e complexas (imagens tridimensionais), que precisam ser bem organizados e estruturados para serem depois acessados pelos usuários potenciais.

Apesar do rápido crescimento dos dados genômicos, o desempenho associado ao acesso e interpretação destes dados pelos usuários ainda é insatisfatória, exigindo mecanismos eficientes de armazenamento e análise.

Nesse artigo, procuramos evidenciar a necessidade de pesquisas na área de bancos de dados que dêem suporte às pesquisas na área de biologia computacional. Particularmente, buscamos colocar em discussão se seria apropriado dispor de sistemas de bancos de dados ad-hoc que atendessem com mais eficácia, e provavelmente também eficiência, os usuários potenciais dessas áreas biológicas e ciências da vida em geral.

Neste trabalho são relatadas várias iniciativas, a partir de idéias discutidas nos últimos anos, em eventos nacionais [Seibel and Lifschitz 2002] e internacionais [Paton and Goble 2001]. As soluções recentes que estendem SGBDs existentes, como no caso do PostgreSQL [Tata et al. 2006], são interessantes porém não são fáceis de usar ou instalar. Um SGBD tradicional por si só inclui uma série de módulos componentes que devem ser configurados mas que por vezes nunca são utilizados. Se não há necessidade de controlar acesso concorrente aos dados, este gerente de transações torna-se algo inútil do ponto de vista de um usuário de sistemas de bancos de dados biológicos,

Algumas idéias bem interessantes surgidas nos últimos anos [Howe et al. 2007, Jagadish and Olken 2004] trazem novos pontos favoráveis a uma solução de SGBD ad-hoc. Estratégias baseadas em *data services* ou ainda em linguagens declarativas e operadores especialmente adaptados, como o *similar join* e outros operadores *bio-específicos* [Chen and Carlis 2003], vem ganhando força e podem se tornar soluções e abordagens práticas no futuro próximo.

Referências

(2003). *14th International Workshop on Database and Expert Systems Applications (DEXA'03), September 1-5, 2003, Prague, Czech Republic*. IEEE Computer Society.

(2007). *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*. www.crdrdb.org.

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J Molecular Biology*, 215(3):403–410.

Bell, G., Gray, J., and Szalay, A. S. (2006). Petascale computational systems. *IEEE Computer*, 39(1):110–112.

Buneman, P., Khanna, S., Tajima, K., and Tan, W. C. (2004). Archiving scientific data. *ACM Trans. Database Syst.*, 29:2–42.

Carvalho, P. C., Glória, R. V., de Miranda, A. B., and Degrave, W. M. (2005). Squid - a simple bioinformatics grid. *BMC Bioinformatics*, 6:197.

Chen, J. Y. and Carlis, J. V. (2003). Similar join: Extending dbms with a bio-specific operator. In *SAC*, pages 109–114. ACM.

Chen, J. Y., Carlis, J. V., and Gao, N. (2005). A complex biological database querying method. In *SAC*, pages 110–114. ACM.

de Carvalho Costa, R. L. (2002). Alocação de dados e distribuição de carga para execução paralela da estratégia blast de comparação de sequência. Master's thesis, Departamento de Informática da PUC-Rio.

de Carvalho Costa, R. L. and Lifschitz, S. (2003). Database allocation strategies for parallel blast evaluation on clusters. *Distributed and Parallel Databases*, 13(1):99–127.

de Macêdo, J. A. F. (2005). *Um Modelo Conceitual para Biologia Molecular*. PhD thesis, Departamento de Informática da PUC-Rio.

de Macêdo, J. A. F., Porto, F., Lifschitz, S., and Picouet, P. (2007). Dealing with some conceptual data model requirements for biological domains. In *AINA Workshops*. IEEE Computer Society.

de Noronha, M. F. (2006). Implementação e avaliação de desempenho de um driver para gerência de e/s em aplicações de bioinformática. Master's thesis, Departamento de Informática da PUC-Rio.

Eltabakh, M. Y., Ouzzani, M., and Aref, W. G. (2007). Bdbms - a database management system for biological data. In [DBL 2007], pages 196–206.

Howe, B., Maier, D., and Bright, L. (2007). Smoothing the roi curve for scientific data management applications. In [DBL 2007], pages 185–195.

Hsiao, R.-L., Jr., D. S. P., and chih Yang, H. (2005). Support for bioindexing in blastgres. In *DILS*, volume 3615 of *Lecture Notes in Computer Science*, pages 284–287. Springer.

Hunt, E., Atkinson, M. P., and Irving, R. W. (2002). Database indexing for large dna and protein sequence collections. *VLDB Journal*, 11(3):256–271.

Jagadish, H. V. and Olken, F. (2004). Database management for life sciences research. *SIGMOD Record*, 33(2):15–20.

Lemos, M. (2004). *Workflow para Bioinformática*. PhD thesis, Departamento de Informática da PUC-Rio.

Lemos, M. and Lifschitz, S. (2003). A study of a multi-ring buffer management for blast. In [DBL 2003], pages 5–9.

Lemos, M., Seibel, L. F. B., and Casanova, M. A. (2003). Bionotes: A system for biosequence annotation. In [DBL 2003], pages 16–20.

Lifschitz, S. and Mauro, R. C. (2005). An i/o device driver for bioinformatics tools: the case for blast. *Genetics and Molecular Research (GMR)*, 4(1):563–570.

Neteler, M. and Mitásová, H. (2002). *Open Source GIS: A GRASS GIS Approach*. Kluwer.

Paton, N. W. and Goble, C. A. (2001). Information management for genome level bioinformatics. In *VLDB 2001*. Morgan Kaufmann.

Poess, M. and Othayoth, R. (2005). Large scale data warehouses on grid: Oracle database 10g and hp proliant systems. In *VLDB 2005*, pages 1055–1066. ACM.

Rosa, J. O. M. (2006). Estruturas de armazenamento e persistência de seqüências e dados biológicos. Master's thesis, Departamento de Informática da PUC-Rio.

Seibel, L. F. B. (2002). *BioAXS: Uma Arquitetura para Integração de Fontes de Dados e Aplicações da Biologia Molecular*. PhD thesis, Departamento de Informática da PUC-Rio.

Seibel, L. F. B., de Macêdo, J. A. F., Lemos, M., Lifschitz, S., de Miranda, A. B., Alves, M., and Degrave, W. M. (2003). A conceptual model for molecular biology information. In *WOB*, pages 47–56.

Seibel, L. F. B. and Lifschitz, S. (2001). A genome databases framework. In *DEXA*, volume 2113 of *Lecture Notes in Computer Science*, pages 319–329. Springer.

Seibel, L. F. B. and Lifschitz, S. (2002). An overview of genomic databases research issues. In *SBBD*, page 10. UFRGS.

Silberschatz, A., Korth, H. F., and Sudarshan, S. (2005). *Database System Concepts, 5th Edition.* McGraw-Hill Book Company.

Sinha, R. R., Termehchy, A., Mitra, S., and Winslett, M. (2007). Maitri demonstration: Managing large scale scientific data (demo). In [DBL 2007], pages 219–224.

Sousa, D. X. (2007). Balanceamento de carga com bancos de dados de sequências genômicas com partições replicadas. Master's thesis, Departamento de Informática da PUC-Rio. (previsão de defesa).

Tata, S., Patel, J. M., Friedman, J. S., and Swaroop, A. (2006). Declarative querying for biological sequences. In *ICDE*, page 87. IEEE Computer Society.

Topaloglou, T., Davidson, S. B., Jagadish, H. V., Markowitz, V. M., Steeg, E. W., and Tyers, M. (2004). Biological data management: Research, practice and opportunities. In *VLDB*, pages 1233–1236. Morgan Kaufmann.

Wong, R. K., Lam, F., Graham, S., and Shui, W. M. (2000). An xml repository for molecular sequence data. In *BIBE*, pages 35–42.