1
Introdução

Um dos principais objetivos do desenvolvimento científico em Biologia Molecular é a análise dos genomas dos seres vivos, sobretudo os dos seres humanos, dos seres causadores de doenças e de seres relacionados. O crescimento acelerado do volume de dados nesta área se deve ao desenvolvimento de técnicas de laboratório sofisticadas que permitem a coleta dos mesmos (Doolittle, 1990). Estes dados consistem principalmente em seqüências de nucleotídeos e aminoácidos, denominadas biosseqüências, as quais codificam o DNA dos seres vivos e as proteínas, respectivamente.

Apesar do rápido crescimento dos dados coletados, a velocidade na qual os usuários conseguem interpretá-los ainda é lenta, exigindo mecanismos eficientes de armazenamento e análise. Atualmente, as pesquisas nesta área são apoiadas computacionalmente através da utilização de bancos de dados de biologia molecular e de diversos algoritmos de comparação e análise dos dados. A área de pesquisa que desenvolve ferramentas de Informática para a Biologia Molecular é a Bioinformática, ou Biologia Computacional.

Para o armazenamento dos bancos de dados de Biologia Molecular, são usados principalmente arquivos texto semi-estruturados, cujos formatos mais comuns são o FASTA (NCBIa, 2006), o ASN.1 (ASN, 2006) e o XML (NCBIb, 2006). Existe também a possibilidade de manter os bancos em Sistemas Gerenciadores de Bancos de Dados (SGBDs) relacionais como o Oracle (Stephens et al, 2005) e o PostgreSQL (CCB, 2006), e em sistemas orientados a objetos (Sakamoto et al., 1994).

Até o momento, não existe um SGBD específico para aplicações de Bioinformática. Os bancos dados de Biologia Molecular armazenam as biosseqüências, anotações referentes às mesmas, famílias e estruturas tridimensionais de biosseqüências, entre outros dados. Estes dados possuem

estruturas complexas e são acessados de maneira bem específica, tornando importante a utilização de um Sistema Gerenciador de Bancos de Dados adequado à área.

Para o biólogo, o primeiro passo após a obtenção de uma biosseqüência é a estimativa de sua função. O meio mais utilizado para este fim é a comparação com seqüências conhecidas, armazenadas junto com suas anotações em bancos de dados como o Genbank (Benson et al., 2000) e o SWISS-PROT (SIB, 2006). Outra maneira de descobrir a função de uma biosseqüência é através da verificação empírica, utilizando-se processos bioquímicos em laboratório. Porém, a verificação empírica é muito mais custosa em termos de tempo e material, e por isso seu uso é, em geral, restrito à confirmação de alguns resultados. A comparação de biosseqüências é uma das tarefas mais importantes na análise dos dados de Biologia Molecular, sendo a base para várias outras manipulações mais elaboradas.

Existem duas principais famílias de algoritmos que realizam comparações de biosseqüências, a FASTA (Pearson, 1991) e a BLAST (Altschul et al., 1990). O desempenho é um fator muito importante para ferramentas de comparação de biosseqüências. Por serem mais rápidas, as ferramentas da família BLAST (que serão chamadas neste trabalho de BLAST) são as mais utilizadas pelos pesquisadores atualmente e existem diversos *sites* que disponibilizam os algoritmos para os usuários.

De modo geral, as ferramentas de comparação de biosseqüências são usadas acessando bancos de dados que são arquivos textos ou binários, não sendo usados SGBDs. Logo, as ferramentas não se beneficiam de mecanismos eficientes de armazenamento, acesso a disco e gerenciamento de memória diferentes dos oferecidos pelo próprio sistema operacional.

Uma possível melhora em ferramentas de comparação de biosseqüências é a inclusão de um gerenciamento de *buffer* adequado para as mesmas, baseando-se no modo como o banco de dados é acessado. Deste modo, é possível obter uma melhora de desempenho diminuindo-se o número de leituras feitas ao disco. Uma estratégia de gerenciamento de *buffer* para o BLAST foi descrita em Lemos et al. (2003), sugerindo o uso de estruturas de armazenamento de seqüências na memória denominadas anéis. Em Mauro et al. (2004), sugere-se a utilização de

um *driver* para implementar este gerenciamento de *buffer* e uma arquitetura de ferramenta é proposta.

Este trabalho consiste na implementação de uma ferramenta provedora de dados que realiza um gerenciamento de *buffer* eficiente para o BLAST, controlando também o escalonamento dos processos do mesmo, e no estudo de técnicas de escalonamento e fatores que influenciam o desempenho dos processos do BLAST. A utilização da ferramenta é feita de maneira transparente ao BLAST, pois a comunicação com este é realizada por meio de um *driver* que substitui as chamadas a funções de leitura e escrita de arquivos do banco de dados. Com o desenvolvimento dessa ferramenta, objetiva-se criar uma solução aplicável também a outras ferramentas de Bioinformática.

Por ser transparente aos programas, a ferramenta desenvolvida é facilmente extensível, podendo ser futuramente modificada para prover dados para outros aplicativos, usar outras estratégias de gerência de *buffer* ou prover dados armazenados em formatos diferentes dos lidos por processos clientes, convertendo-os em tempo de execução. A ferramenta desenvolvida foi denominada BioProvider. Em sua implementação, foram usadas algumas das idéias propostas em Lemos et al. (2003) e Mauro et al. (2004).

No capítulo 2 será apresentado o contexto da dissertação, incluindo uma descrição de características do BLAST importantes para este trabalho e a apresentação de trabalhos relacionados. O capítulo 3 descreve a arquitetura e o funcionamento do BioProvider. No capítulo 4 serão mostrados os resultados dos testes realizados para avaliar o desempenho do BLAST usando o BioProvider, assim como fatores que o influenciam. Finalmente, serão apresentadas as conclusões obtidas com a realização do trabalho e propostas de trabalhos futuros.

Foram incluídos 3 apêndices neste trabalho, para uma melhor descrição dos conceitos envolvidos. No apêndice A é apresentado um resumo do algoritmo básico do BLAST. Alguns conceitos de gerenciamento de memória são descritos no apêndice B, assim como a maneira como este é feito pelos Sistemas Gerenciadores de Bancos de Dados. O apêndice C mostra algumas características de *drivers* do Linux e como estes são implementados.