3.

Blast em Ambientes Distribuídos

Entende-se por rede computacional um conjunto de computadores autônomos interconectados através de uma simples tecnologia. Eles são ditos interconectados caso sejam capazes de trocar informações [6, 49]. Através da utilização de uma rede é possível dividir uma aplicação em partes semi-independentes de acordo com o tipo de processamento necessário. Assim, a diferentes nós de uma rede podem ser atribuídas diferentes partes de uma aplicação [7], e a coordenação do processamento dessas aplicações na rede pode ser efetuada por algum, ou alguns, dos computadores a ela pertencentes.

Clusters nada mais são do que redes compostas por *workstations*, computadores pessoais ou sistemas multiprocessados que podem estar engajados em um processamento próprio ou também colaborando com outras unidades no sentido de cumprir alguma determinada tarefa computacional. Cada nó possui uma arquitetura e um sistema operacional próprios. Para alcançar um alto desempenho, a interconexão entre os diversos nós de um *cluster* deve ser em banda larga, e possuir baixa latência. *Clusters* dedicados, normalmente, possuem todos os equipamentos que os compõem localizados fisicamente próximos, e utilizam redes ethernet de alta velocidade [7].

Grid Computing, ou simplesmente *grid*, é um tipo especial de computação, no qual computadores são interligados através da Internet [9], e que pode simular, em seu conjunto, um supercomputador virtual, da mesma maneira que os *clusters* o fazem, definindo, assim, um ambiente de alto desempenho, que extrapola as fronteiras de uma única organização.

Ambientes computacionais de *cluster* e *grid* são bastante utilizados para a execução do BLAST, uma ferramenta desenvolvida pelo *National Center for*

Biotechnology Information, ou NCBI, em 1989 [3]. O BLAST permite que se compare uma determinada seqüência de consulta, ou *query*, com uma base de dados de seqüências conhecidas, identificando aquelas que atingem um determinado grau de similaridade [11], utilizando, para isso, ferramentas estatísticas e matrizes de substituição.

No entanto, é desejável medir o ganho de desempenho obtido ao utilizar o BLAST em ambientes de *cluster* e de *grid*, tanto em relação ao BLAST executado em máquinas isoladas, quanto em relação a outras ferramentas que permitam a execução do BLAST em ambientes distribuídos. No capítulo anterior foram citados alguns fatores a considerar em um processo de avaliação de ferramentas de software, assim como critérios para a adoção de métricas adequadas para essa avaliação, e que poderiam ser aplicáveis na avaliação de ferramentas BLAST em paralelo.

Neste capítulo são inicialmente apresentados alguns tópicos atinentes à execução do BLAST em ambientes de *cluster* e de *grid*. Em seguida são citados alguns dos trabalhos relacionados ao desenvolvimento e implantação de ferramentas BLAST em paralelo, nesses ambientes. Ao final é discutido como o desempenho dessas ferramentas foi avaliado pelos seus desenvolvedores.

3.1

BLAST

BLAST é atualmente a ferramenta mais utilizada para a identificação de seqüências desconhecidas, a partir da análise de similaridades entre seqüências de nucleotídeos e proteínas, em bioinformática [3].

Existem razões para isso [3]: BLAST é rápido, e este fato é importante tendo em vista o atual crescimento das bases de dados biológicas; é um programa confiável, tanto do ponto de vista do rigor estatístico, quanto do *software* desenvolvido; e é uma ferramenta flexível que se adapta a diferentes cenários que surgem na análise de seqüências.

Entende-se por bases de dados biológicas aquelas que contêm informações sobre áreas de pesquisa que incluem dados genômicos, proteômicos, metabolômicos, *microarays* e filogenéticos [36].

A análise de similaridades entre seqüências parte de um processo de alinhamento entre essas seqüências. Encontrar alinhamentos que sejam ótimos para efetuar a análise não é uma tarefa computacional simples. Existem dois algoritmos de alinhamento que são considerados básicos em bioinformática: Global, conhecido como *Needleman-Wunsch* [3], e que considera as seqüências como um todo; e Local, conhecido como *Smith-Waterman* [3] em que o melhor alinhamento subseqüente é encontrado. Ambos os algoritmos utilizados para o alinhamento possuem um grau de complexidade O(mn), onde m e n são respectivamente os tamanhos das seqüências comparadas.

Ao contrário do algoritmo *Smith-Waterman*, que encontra o melhor alinhamento entre duas seqüências, o BLAST não explora todo o espaço de pesquisa entre duas seqüências, e reporta apenas os alinhamentos que possuam, alinhamentos estatisticamente significativos. Para isso ele utiliza 3 heurísticas que permitem que ele encontre similaridades entre essas seqüências sem percorrer todo o espaço de pesquisa, sendo desta forma mais eficiente [3]:

a) *Seeding*:

   BLAST assume que alinhamentos significativos possuem palavras em comum. Assim ele fixa um tamanho de palavra, digamos três letras, e identifica a vizinhança dessa palavra, que possua um *score* significativo. Para calcular o *score*, BLAST utiliza matrizes de substituição, como, por exemplo, BLOSSUM, para efetuar o cálculo. Para ser considerada uma vizinhança, é estabelecido que o *score* seja tão grande como um determinado valor T. O tamanho da vizinhança que pode ser aproveitada será determinado pelo valor de T;

b) Extensão:

   Uma vez geradas as sementes (*seeds)*, o alinhamento é estendido em ambas às direções para cada um deles. Essa extensão, no entanto, possui um ponto de parada estabelecido por um valor X, que define um limite

máximo de decréscimo do *score*, tomando como base o último valor máximo encontrado;

c) Avaliação:

Uma vez que todos os alinhamentos tenham sido estendidos, eles devem ser então avaliados para determinar quais são estatisticamente significativos. Os alinhamentos considerados mais significativos são denominados *High Score Pair*, ou simplesmente HSP.

Apesar de ser eficiente, existe atualmente uma grande preocupação quanto à performance do BLAST, considerando-se o crescimento dos dados de DNA e proteínas, resultantes de projetos de seqüenciamento [9, 10]. A utilização de ambientes paralelos para a execução do BLAST vem sendo intensamente pesquisada [10] no sentido de acelerar a execução em um ambiente de alta performance.

A partir de uma ótica voltada à tecnologia de Bancos de Dados, existem duas formas básicas de tratar o paralelismo do BLAST em ambientes distribuídos [5, 10]:

a) Replicação da Base de Dados Biológica:

A Base de Dados é replicada por todos os nós de um *cluster* ou *grid*. Durante a execução do BLAST, cada uma das seqüências de consulta a ser analisada é submetida a um, e apenas um, dos nós.

b) Fragmentação da Base de Dados Biológica:

A Base de Dados é fragmentada, e cada um dos fragmentos é colocado em algum dos nós de um *cluster* ou *grid*. Variações da estratégia permitem a alocação de mais que um fragmento a um nó.

As ferramentas que permitem a execução do BLAST em ambiente distribuído utilizam uma das estratégias de paralelismo. O mpiBLAST [18], uma das ferramentas mais conhecidas e usadas pelos biólogos e bioinformatas, utiliza como estratégia de paralelização, a fragmentação da base de dados. Já o Grid-BLAST [20], uma ferramenta voltada à execução do BLAST em ambiente de *grid*, utiliza a replicação da base de dados para obter a paralelização.

3.2

Ferramentas BLAST em Ambientes Distribuídos

São apresentadas nesta Seção, algumas das ferramentas que foram desenvolvidas para executar a ferramenta BLAST em ambientes distribuídos.

3.2.1

mpiBLAST

É uma das ferramentas mais conhecidas, pelos biólogos e bioinformatas, dentre aquelas que implementam o BLAST paralelo em ambientes distribuídos. O seu algoritmo funciona em duas etapas [18]: inicialmente a base de dados biológica é fragmentada, utilizando o programa "mpiformatdb", e colocada em um diretório compartilhado; em seguida, durante o processamento, caso um determinado nó não possua um fragmento da Base para pesquisar, ele copia um fragmento para aquele nó, a partir do diretório compartilhado.

É importante frisar que a fragmentação da base de dados efetuada para a execução do mpiBLAST permite que cada nó pesquise uma menor porção da Base de Dados, diminuindo, em conseqüência, a necessidade de se efetuar E/S, o que certamente resulta em um aumento no desempenho da ferramenta [18].

3.2.2

balaBLAST

Desenvolvido por Daniel Sousa durante a elaboração de sua dissertação de mestrado [19], balaBLAST, que utiliza uma estratégia com base de dados fragmentada para incrementar o desempenho do BLAST. Assim como o

mpiBLAST, a ferramenta utiliza o MPI, e deve seu nome à ênfase em estratégias voltadas ao balanceamento de carga, de modo a possibilitar um maior aproveitamento dos recursos computacionais existentes, e assim acelerar a execução do BLAST.

Em seu trabalho, Sousa considerou as seguintes estratégias de balanceamento [47]:

a) Sob Demanda:

   A base de dados é fragmentada e todos os fragmentos são alocados em todos os nós. As seqüências de consulta são enviadas para processamento à medida que os nós estejam ociosos.Todas as seqüências devem ser comparadas com cada um dos fragmentos;

b) Corretiva:

   A base de dados é fragmentada e todos os fragmentos alocados em todos os nós. Todas as seqüências de consulta são enviadas para todos os nós, mas ao se constatar problemas para a execução de um conjunto de seqüências em um determinado nó, elaspodem ser reenviadas a outros nós para serem processadas.

Nas estratégias acima citadas, pode-se observar que todos os fragmentos da base de dados são replicados em todos os nós. Nota-se que se trata de um procedimento distinto daquele adotado pelo mpiBLAST, em que, inicialmente, apenas um determinado fragmento é alocado a cada nó.

Adicionalmente dois conceitos bastante interessantes são adotados no balaBLAST:

a) Fragmentos primários:

   São os fragmentos contra os quais as seqüências de consulta serão comparadas, em um determinado nó, com o objetivo de encontrar similaridades. É possível um determinado nó possuir mais do que um fragmento primário.

b) Fragmentos secundários:

São os demais fragmentos alocados a um determinado nó, os quais poderão ser utilizados, na comparação de seqüências de consulta, em substituição aos fragmentos primários, caso ocorra algum problema durante o processsamento do BLAST.

A Figura 4.1 mostra a disposição dos fragmentos primários e secundários após a sua distribuição pelos nós trabalhadores. Deve-se notar que 6 fragmentos foram distribuídos por 3 nós, e desta maneira existem 2 fragmentos primários em cada nó escravo. Assim, as seqüências distribuídas para o nó 1 serão normalmente confrontadas, quanto ao alinhamento, com os fragmentos 1 e 2, que são fragmentos primários para aquela máquina.

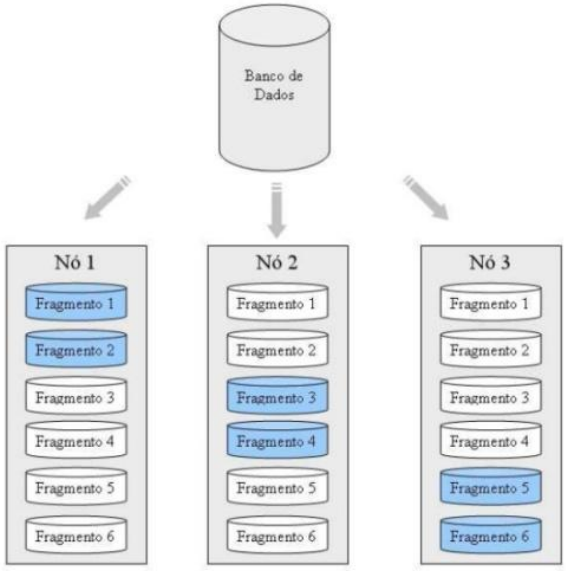

Figura 4.1: *Alocação de fragmentos utilizando a Estratégia Sob Demanda.*

Supondo que antes de concluir o processamento o nó 1 apresente algum defeito, ou esteja sobrecarregado, prejudicando o rendimento, a ferramenta balaBLAST pode reenviar parte da tarefa para o nó 2 ou 3, utilizando no processamento, os fragmentos secundários 1 e 2.

Tais conceitos vêm de encontro à robustez da ferramenta, permitindo uma maior flexibilidade no processamento e viabilizando procedimentos com o objetivo de prover tolerância a falhas.

3.2.3

Grid-BLAST

Trata-se de uma implementação desenvolvida a partir de um estudo efetuado no *Royal Institute of Technology* [20]. O NorduGrid, um ambiente de *grid* que possui como *middleware* básico uma versão do Globus Toolkit, foi utilizado para o seu desenvolvimento e implementação, utilizando, adicionalmente, como *middleware* de serviço o *Advanced Resource Connector middleware* (ARC).

O Grid-BLAST utiliza dois scripts escritos em Perl. O primeiro deles é destinado a submeter, monitorar e controlar os serviços, além de dividir o arquivo de entrada em arquivos menores. O segundo efetua a execução do BLAST.

Inicialmente, a base de dados a ser utilizada, é formatada, utilizando-se para isso o programa "formatdb", incluído no pacote BLAST. As seqüências de entrada são então divididas em um determinado número de arquivos. Um igual número de *jobs* (conjunto de seqüências a serem enviadas a cada nó para processamento) é criado e submetido aos nós do *grid*. Essas submissões consistem em pacotes compactados enviados aos nós, contendo a base de dados formatada, um dos arquivos de seqüências, o BLAST executável e a matriz BLOSSUM, que uma vez descompactados, são então executados.

3.2.4

GBTK

Construído com base em um modelo de *web services*, e mais sofisticado que o Grid-Blast, o Grid Blast Toolkit, ou simplesmente GBTK [21], foi desenvolvido em Perl e disponibiliza um framework e um portal web para a submissão de pesquisas utilizando o BLAST.

O modelo de *web services* adotado é constituído por três componentes: um *producer* de *web services*, um *broker* que possui os registros dos serviços disponíveis e manipula submissões de *jobs*, e um *consumer* que utiliza *web services* via o *Broker*. Os serviços oferecidos são: carga de CPU, que retorna o grau de carga de um nó; BLAST; *Heart Beat*, que permite ao nó central verificar se um nó escravo está ou não ativo; transferência de arquivos; e recebimento de arquivos.

As bases de dados são distribuídas pelos nós sem que existam redundâncias, de maneira que as maiores bases estejam em equipamentos com maior poder computacional, e as menores em máquinas menos potentes.

O algoritmo utilizado é baseado em um modelo "*First Come - First Serve*", FCFS. Sempre que um *job* é identificado pelo sistema, o *broker* verifica o nó em que ele deverá ser processado em função da informação contida no database *registry*, que é outro *web service* fornecido pelo *broker*, e que indica a localização de uma base de dados no *grid*. Caso o nó que contenha a base de dados esteja disponível, então o *job* é executado nele, senão uma pesquisa é efetuada para verificar se existe algum nó menos carregado, e nesse caso, a base de dados é copiada para aquele nó, utilizando os serviços de transferência e recebimento de arquivos.

3.2.5

BRIDGES Project

Biomedical Research Informatics Delivered by Grid Enabled Services [22], ou BRIDGES, implementa uma versão do BLAST que utiliza a tecnologia de *grid*, mais especificamente a camada *Fabric* do *Globus Toolkit 3*. Nesse ambiente, cada serviço representa um único *job* submetido por um único usuário.

O paralelismo, nessa implementação é obtido através da utilização do programa "mpiformatdb", do mpiBLAST, para efetuar a fragmentação da base de dados, e pelo particionamento das seqüências de consulta por *jobs*, tomando por

base o número de nós em que o BLAST será rodado. Feito isso, os *jobs* são enviados a esses nós, utilizando-se um escalonador, sendo então processados.

3.2.6

Package BLAST

Desenvolvido por pesquisadores da Universidade de Otawa e Universidade de Brasília, o Package BLAST [23] é uma ferramenta voltada ao ambiente de *grid*, que foi projetada sobre a versão 3 do Globus Toolkit, adotando um esquema *mestre-escravo.*

Para executar o Package BLAST, a base de dados deve ser fragmentada e, então, todos os fragmentos alocados em todos os nós escravos que devem participar do processamento do BLAST. Durante a execução, as seqüências de consulta são enviadas para os nós de acordo com um *framework* que incorpora políticas de alocação de serviços e de atribuição de pesos. Esse *framework* tem por objetivo adequar a política escolhida ao grau de heterogeneidade do ambiente do *grid*. Ao final de cada execução, o resultado é enviado ao nó mestre, para formatação do resultado final.

3.2.7

Squid

O Squid [24] oferece um ambiente computacional simples, independente de plataforma, e de baixo custo, que não exige a utilização de qualquer *middleware*, tal como o *Globus Toolkit*, proporcionando facilidade tanto na sua instalação, quanto na sua manutenção.

Desenvolvido em Perl, o Squid é composto por dois programas, sendo um deles responsável pela administração remota de recursos, e o outro, denominado *Tentacle*, é responsável pela administração de cada nó. Uma interface gráfica é

oferecida, e permite a visualização dos recursos computacionais existentes, além de permitir a seleção de seqüências de consulta e as bases de dados a serem utilizadas. O Squid pode também ser executado a partir de um *prompt* de comando, que permite o encapsulamento com outros programas, ou ainda a utilização de *pipelines.*

Para que o Squid possa ser utilizado, cada um dos nós do *grid* deve possuir a ferramenta BLAST instalada, assim como o programa *Tentacle*, além de uma cópia da base de dados. As seqüências de consulta, chamadas durante a execução, são divididas em fragmentos e, então, distribuídas pelos nós que irão rodar o BLAST, até que não existam mais fragmentos a serem comparados. Um fragmento só é retirado de um determinado nó após a execução ter sido satisfatoriamente completada e o arquivo com o resultado retornado ao mestre.

3.2.8

HGBS

Hardware-Oriented Grid BLAST System [25] não utiliza um *middleware* convencional para construir um ambiente de *grid*, mas sim, um chip especializado que é instalado em cada um dos equipamentos que irá compor o *grid*. Assim, em todos os PC é instalada uma ROM na placa Ethernet, que permite que ele se conecte ao *grid*, bastando para isso anotar o endereço MAC (endereço físico da placa de rede) no servidor.

No HGBS os *jobs* são mantidos em uma fila no servidor principal, enquanto aguardam uma requisição por parte dos outros nós do grid, onde eles serão processados. Cada um desses nós contém o programa BLAST instalado e uma cópia doa base de dados que será utilizada para comparação. Em tempo de execução, algumas propriedades são verificadas pelo escalonador, para testar o estado do *job*, entre elas: a sua identificação; se está aguardando, foi iniciado ou já foi processado; a lista de dados sendo processada; a hora de início e fim do processamento, dentre outras

3.2.9

W.nd BLAST

O W.nd BLAST [26] é uma ferramenta que utiliza uma rede local baseada no sistema operacional Windows, para rodar o BLAST em paralelo. Escrito em C#, ele oferece uma interface gráfica, a partir da qual o usuário executa o BLAST, e visualiza o resultado. Provê ainda mecanismos de tolerância a falhas e recuperação, caso elas ocorram.

Desenvolvido para ser executado em um ambiente Windows, o W.nd BLAST [27], procura paralelizar o BLAST, distribuindo as seqüências de consulta para equipamentos baseados em Windows, que pertençam a uma determinada rede local.

Ele é composto por módulos chamados de soluções de software integradas, assim discriminadas: *Project Control*, em que o usuário define os projetos para a execução do BLAST; *Database Engine*, que permite a criação e distribuição de novas bases de dados, utilizando uma interface completa de execução do "formatdb"; BLAST *Engine*, que pesquisa e encontra *workstations* que estejam habilitadas e configuradas para rodar o W.ind BLAST; W.nd BLAST *Distribution Algorithm*, que em uma arquitetura cliente servidor, argúi os nós sobre a execução do BLAST, utilizando para tal, o protocolo TCP/IP; e BLAST *Output Viewer*, que permite ao usuário visualizar os resultados obtidos pela execução do BLAST.

3.2.10

Green Gene

O GreenGene [28] tem como objetivo obter um ganho de performance na execução do mpiBLAST através da utilização de uma estrutura contendo três níveis: *Super Master*. *Group Master e Group*. O *Super Master* é responsável por submeter as seqüências de consultas para os *Group Masters* (*clusters*), que por sua vez executam o mpiBLAST e enviam os resultados para o *Super Master*.

Nessa arquitetura, cada um dos nós existentes nos *clusters* é um equipamento multiprocessado (SMP) ou o seu processador possui múltiplos *núcleos*. Cada *núcleo* é considerado um nó virtual, de modo a se obter um máximo poder computacional, e cada Group é composto por um conjunto de nós virtuais, que rodam um *job* mpiBLAST.

3.2.11

ThuBioGrid

Trata-se de uma implementação do BLAST em paralelo que incorpora serviços de diretório. métodos de computação em *grid* e métodos de processsamento de seqüencias genômicas [29].

Desenvolvido em Java, como um *servlet*, ele utiliza o *Java CoG Kit Plus for Bioinformatics Aplications* (CoG: *Comodities Grid*) [48], compatível com as versões 2.2 e 2.4 do Globus Toolkit, para se conectar ao ambiente de rede do *grid* composto pelo Globus Toolkit, e efetuar a comparação de seqüências utilizando para isso as ferramenta mpiBLAST e FASTA.

3.3

Trabalhos Relacionados

As Ferramentas BLAST citadas acima, foram desenvolvidas com o intuito de prover maior eficiência na execução do BLAST. No entanto, é necessário que se faça uma avaliação do seu desempenho com o objetivo de demonstrar qual o ganho real obtido com a sua utilização.

De um modo geral, houve preocupação, por parte dos desenvolvedores, em verificar somente o desempenho de uma determinada ferramenta face a diferentes números de nós, bases biológicas, tamanhos de seqüências e número de seqüências por tarefa. Ou seja, não há uma comparação entre desempenhos

obtidos em diferentes ferramentas BLAST em paralelo [19, 23, 24, 25, 26]. Dentre as ferramentas citadas foram retirados três exemplos que demonstram tal fato:

a) Em [30] foram apresentados os resultados obtidos no projeto Bridges para a execução do BLAST em paralelo. As avaliações consideraram os tempos de execução do BLAST tanto em uma máquina isolada, como em um ambiente de *grid*, para arquivos de consulta contendo diferentes números de seqüências a serem processadas.

b) Na avaliação do Package Blast [23], os testes foram efetuados considerando os diferentes tipos de estratégias de escalonamento de tarefas suportadas pela ferramenta, contemplando tanto um único nó como também mais nós de um *grid*, rcrcutando o programa "blastx" do pacote NCBI BLAST.

c) No caso do ThuBioGrid [29] a avaliação considerou o desempenho das ferramentas mpiBLAST e do GE-FASTA (uma ferramenta criada para rodar o FASTA em paralelo em ambiente de *grid*) quando executados com, e sem, o ThuBioGrid. Na realidade o que se procurou foi medir os ganhos que se poderia obter pelo uso do paralelismo para ferramentas NCBI BLAST e FASTA, e não propriamente uma comparação entre estas ferramentas.

Nos três casos acima, assim como nos demais, a métrica considerada para a medição do desempenho foi o "Tempo de Execução" das ferramentas em cada uma das condições submetidas para avaliação.

Embora os desenvolvedores das ferramentas aqui citadas não tenham efetuado comparações entre diferentes ferramentas BLAST em paralelo, mas sim, em diferentes situações de uso de uma mesma ferramenta, algumas idéias consideradas nas avaliações por eles utilizadas foram julgadas perfeitamente aplicáveis ao presente trabalho, dentre as quais:

a) A utilização da métrica "Tempo de Execução" para avaliação do desempenho das ferramentas;

b) A verificação do comportamento das ferramentas considerando diferentes situações em um ambiente distribuído, como por exemplo, a utilização de diferentes números de nós, e o uso de equipamentos com diferentes capacidades de processamento para a execução de testes;

c) A preocupação em verificar o comportamento das ferramentas considerando grandes bases de dados;

d) A utilização de seqüências de consulta retiradas da própria base de dados a ser considerada na execução;

3.5

Conclusão

Neste capítulo, inicialmente, foi argumentado como a utilização do paralelismo para executar o BLAST pode, de fato, acelerar a sua execução, através da divisão de tarefas entre diversos nós permitindo utilizar a potencialidade de múltiplos processadores na análise de similaridades.

A paralelização da execução do BLAST é obtida a partir da replicação ou fragmentação de uma base de dados biológica por diversos nós, de modo a tomar partido da menor amplitude de dados a analisar por processador, além de uma menor utilização de memória RAM, no caso de fragmentação da base, nos diversos equipamentos, diminuindo, portanto, a necessidade de E/S e, em conseqüência, acelerando o processo.

Em seguida foram apresentados alguns trabalhos que resultaram na criação de ferramentas BLAST voltadas ao ambiente distribuído, inclusive uma desenvolvida na PUC-Rio, e finalmente como foram efetuadas as avaliações dessas ferramentas pelos seus desenvolvedores.

No que tange às avaliações efetuadas pelos desenvolvedores das ferramentas BLAST aqui apresentadas, pode-se constatar, a partir dos artigos apresentados, que não foram efetuadas comparações entre diferentes ferramentas BLAST voltadas ao ambiente distribuído. No entanto, tal comparação toma importância à

medida do atual crescimento das bases de dados biológicas, como citado na Seção 3.1 e, por este motivo, constitui um dos objetivos da presente dissertação.