Avaliação Prática da Ferramenta BioProvider

Aluno: Waldecir Faria
Orientador: Sérgio Lifschitz

Introdução

A dissertação de mestrado de Maíra Noronha [1], defendida em 2006 no Departamento de Informática da PUC-Rio realizou um estudo voltado para a área de gerenciamento de banco de dados para bioinformática. Este estudo trata de uma ferramenta capaz de controlar e acelerar o acesso a um banco de sequências pelo programa BLAST [2] chamada BioProvider.

Este BLAST pode, dado um banco de sequências biológicas e um conjunto de sequências de consulta, aproximar quais sequências de aminoácidos ou nucleotídeos tem maior semelhança entre si. O BLAST faz isto com uma velocidade considerável se comparado com algoritmos como o de Smith-Waterman [2], por usar heurísticas eficazes e eficientes, assim sendo vastamente usado por pesquisadores.

Um dos trabalhos futuros na época mencionado e que não havia sido explorado até o momento dizia respeito ao uso da ferramenta na presença de bases de dados mais volumosas. Além disto, como o *kernel* dos sistemas operacionais pode se alterar com o tempo, seria necessário gerar uma versão mais robusta da ferramenta. Por fim, recentemente o NCBI passou a adotar uma versão nova de BLAST, BLAST+ [6] que necessita de avaliação para verificar a adequação do uso com o Bioprovider

Objetivos

Checar se esta ferramenta BioProvider continua funcionando para sistemas operacionais atuais, corrigir *bugs,* atualizar o programa, acelerar a execução e tornar o acesso à memória de programas BLAST mais eficiente através de um gerenciamento de *buffer e de* processos de forma não intrusiva.

Metodologia

A. Estudo e Teste de funcionamento

Primeiramente foi feita uma leitura da tese de dissertação da Maíra [1] sobre o BioProvider para se entender exatamente o que ele é. Também foi feito um pequeno estudo para entender o básico da ferramenta BLAST [2].

A ideia por trás do BioProvider é prover dados realizando um gerenciamento de *buffer* eficiente para o BLAST, controlando também o escalonamento dos processos do mesmo. A comunicação entre o BioProvider e os processos do BLAST, assim como o controle de concorrência e bloqueios, é feita por meio de um *driver*, que substitui as chamadas a funções de leitura e escrita de arquivos do banco de dados. Deste modo, o código do BLAST não precisa ser modificado para se realizar a comunicação com o BioProvider e este pode ser usado para diferentes versões do BLAST.

Depois foi preciso conferir se o *software* continuava funcional devido ao tempo que se passou desde quando o primeiro estudo foi realizado. Baseado nos *scripts* e nas documentações do estudo anterior, foi criado um novo *script* de preparação para a execução do BLAST na presença do BioProvider.

Este primeiro *script* basicamente prepara o BioProvider para rodar com o banco de proteínas pataa [3], usando parâmetros fixos, sem nenhum tipo de interação com o usuário. Com algumas modificações foi possível rodar o BLAST com o auxílio do BioProvider. Assim

mostrou-se que o BioProvider continua funcionando corretamente numa versão de sistema operacional *Ubuntu* com *kernel* mais atual que a do primeiro estudo (2.6.32-21-generic) e que inicialmente não seria necessária nenhuma mudança no código do BioProvider para seu funcionamento.

B. Melhoramento do script

Após este teste, baseado em um *script* criado anteriormente pela aluna Natália, foi criado outro cujo objetivo é configurar os parâmetros necessários para a execução do BioProvider via linha de comando de uma maneira amigável para o usuário. Com ele a execução dos testes se torna mais fácil, pois ele pode prevenir erros de escolha dos parâmetros, o esquecimento da preparação do banco antes do seu processamento ou do uso de arquivos não existentes e também facilita o uso do BioProvider para outros usuários.

Resumidamente, os parâmetros que podem ser modificados via linha de comando são:
- O banco de sequências;
- Quantidade de memória disponível para o BioProvider gerar o anel (tamanho em Mbytes ou em porcentagem de memória livre disponível no momento);
- Quantidade de blocos pelo qual o anel será dividido.

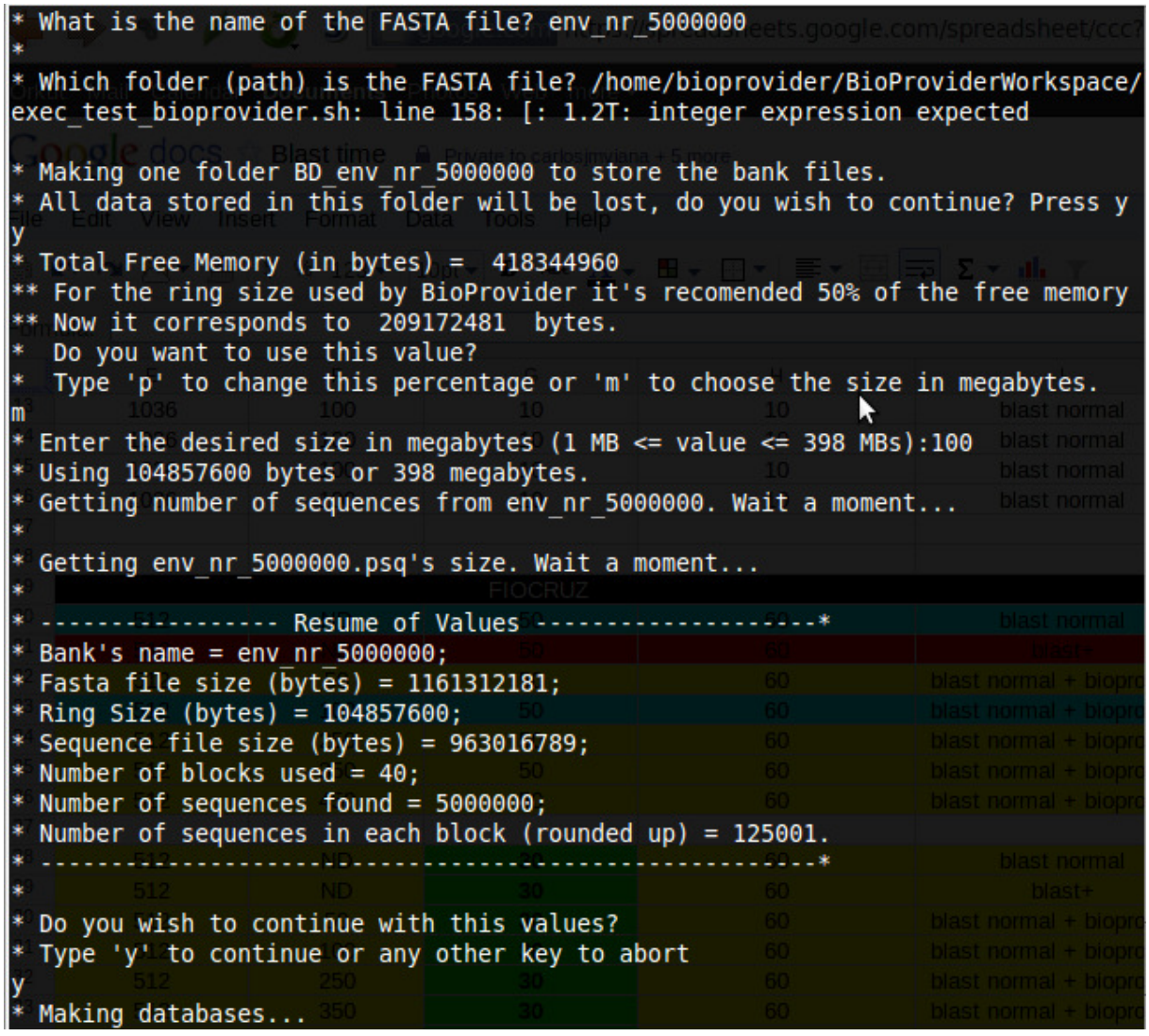

Figura 1: Imagem do script em execução no terminal.

C. Preparação para o uso do BioProvider com o BLAST

O BioProvider roda com informações retiradas de um arquivo de configurações. Desde que os arquivos usados por ele estejam preparados, é possível criar este arquivo de configurações manualmente e executar o BioProvider. Porém como os passos de preparação para a execução são sempre os mesmos, é mais prático automatizar esta preparação através de um *script.* Logo a explicação dos passos abaixo voltados para o uso do BioProvider são os mesmos usados por este *script.*

Para se usar o BLAST para o alinhamento de sequências é preciso executar o formatdb [4], programa associado ao BLAST que visa preparar o banco para ser processado. Com parâmetros padrões, o formatdb gera três arquivos: um arquivo de sequências (.psq), um de índices (.pin) e um de anotações (.phr).

Como o BioProvider faz o controle dos processos que acessam estes arquivos e dos dados lidos a partir destes, principalmente o de sequências, o *script* precisa executar este formatdb uma vez e armazenar o tamanho deste arquivo de sequências para saber quantos blocos serão precisos a partir da quantidade de memória disponível para o seu anel de dados.

O BioProvider usa uma estrutura para fazer o *buffer* dos dados requisitados chamada de anel. Este modelo recebe este nome por ser, resumidamente, uma lista encadeada circular de blocos que faz o papel de um *buffer* do banco de sequências a serem consultadas.

Os blocos são trechos do banco carregados na memória para acesso futuro. Quando todos os processos ativos tiverem processado um determinado bloco, este pode ser substituído por um novo. O importante para que o BLAST funcione corretamente assim é que todo o bloco se inicie com um começo de sequência. Desta forma, mesmo se lendo o banco fora de ordem, o resultado da execução será correto ao se terminar de processar todos os blocos. Caso o bloco não começasse com um início de sequência, o BLAST não saberia disto e consideraria o início do bloco como um início de sequência, gerando um resultado errôneo.

Ao se usar o BioProvider, os processos BLAST leem o arquivo gerenciado por ele de maneira diferente da habitual. Normalmente a leitura começa sempre da primeira sequência do banco. Usando o BioProvider ele pode começar de uma sequência em qualquer ponto do banco. Caso se deixe o sistema operacional controlar os outros dois arquivos sem se fazer nada enquanto o BioProvider controla apenas o acesso ao arquivo de sequências, ocorreria um problema, pois o arquivo de índices poderia estar apontando para uma posição errada do arquivo de sequências gerenciado pelo BioProvider. Para se solucionar este empecilho os arquivos sofrem uma "permutação" para cada tipo de visão que este pode ter.

O número de permutações é determinado pelo número de blocos necessários para se armazenar o banco no anel de memória, porque cada processo BLAST pode iniciar a leitura de um bloco qualquer, desde que este bloco comece com um início de sequência e no final todo o arquivo seja processado.

Estas permutações também são formatadas com o formatdb, pois não é possível saber de qual bloco a leitura começará, e suas informações são carregadas para o BioProvider através do arquivo de configurações. O *driver* usado para passar informações de maneira não intrusiva é carregado no *kernel* e são gerados três arquivos de dispositivos de caracteres, gerenciados por este *driver*, que serão usados no lugar dos três arquivos originais gerados pelo formatdb. Desta maneira é possível controlar as informações lidas pelo BLAST de maneira transparente para o mesmo.

Após disto tudo os arquivos gerenciados pelo BioProvider estão prontos para serem usados pelo BLAST. Eles são exibidos pelo sistema operacional como arquivos de zero *byte*, porém ao se requisitar leitura deles, os dados são transmitidos corretamente graças ao *driver* que implementa as operações de processamento de arquivos [5].

Estas operações incluem ações pré-definidas pelo sistema operacional para manipulação de dispositivos de caracteres como leitura e escrita de arquivo e deslocamento da posição atual de leitura dentro do arquivo, todas baseadas nas informações contidas no anel. Desta forma é possível usar ler os arquivos de maneira transparente, seja via o BLAST ou por outros meios como os comandos de Linux "head" ou o "tail" sobre estes arquivos.

Para se executar o BLAST com o BioProvider basta rodar o BLAST normalmente pela linha de comandos passando como o caminho do banco de consultas o local onde os três arquivos gerados pelo BioProvider estão.

Como é preciso carregar este *driver* para executar o BioProvider, quando se desejar encerrar o programa ou então desligar/reiniciar a máquina deve-se ter um cuidado especial. É preciso usar um programa especial para avisar ao BioProvider que ele deve ser encerrado e que o *driver* pode ser desativado. Se isto não for feito o sistema pode ficar inconsistente.

D. Execução de testes

Com o BioProvider funcional e com o *script* de configuração pronto, pode-se começar a fazer os testes. Para isto foram criados mais dois *scripts* para automatizar os passos de testes. Um destes para executar uma carga de processos BLASTs sem o BioProvider e outro com ele. O *script* que usa o BioProvider não precisa chamar o formatdb antes da execução do BLAST pois ele já é executado em um dos passos para a preparação do BioProvider.

Além de facilitar a escolha dos parâmetros de teste, os *scripts* facilitam na hora de armazenar e organizar os resultados obtidos a cada execução do teste. Também há um *script* auxiliar usado para calcular o tempo usado pelo teste.

Para mostrar o momento no qual o uso do BioProvider pode realmente auxiliar na execução do programa ao invés de o tornar mais lento, foram feitos diferentes testes buscando simular diversos ambientes para cada banco de sequências. Em cada instância de um teste um ou mais dos seguintes elementos foram alterados:

- Quantidade de memória disponível;
- Banco de sequências;
- Conjunto de sequências para consulta;
- Quantidade de BLASTs executados por teste;
- Intervalo de tempo em segundos entre o início de um BLAST e outro, simulando uma carga de trabalho em laboratório;
- Ao se usar o BioProvider, quantidade de memória disponível para a criação do anel;

Com estes novos *scripts* o BioProvider foi testado inicialmente com o mesmo banco pataa [3] do *script* inicial. Os resultados da execução com o pataa mostraram que o BioProvider realmente pode acelerar a execução do programa BLAST em alguns casos, particularmente ao se processar bases maiores do que a quantidade de memória RAM disponível e ao se rodar muitos processos concorrentemente. Além disto, cabe ressaltar que o resultado dos testes obtidos com e sem o uso do programa provedor de dados são os mesmos.

A ideia usada pelo BioProvider para acelerar programas que fazem leituras sequenciais como o BLAST é a de se reduzir a necessidade de se fazer *swap* do conteúdo para o disco rígido no lugar de se usar a memória. Com o BioProvider é possível reaproveitar a informação já carregada no seu anel na memória para outros processos que se iniciem posteriormente.

Por conta disto, um dos principais parâmetros dos testes é a quantidade de memória disponível em relação ao tamanho do banco de sequências. Quando o banco cabe totalmente na memória, o uso do BioProvider, em tese, torna o desempenho do BLAST mais lento do

que sem ele. Já quando o banco não couber totalmente na memória, o BioProvider provavelmente acelerará seu desempenho através do uso do anel.

Outro parâmetro importante de um teste é o tempo entre disparos de execuções do BLAST. Se ele for muito curto, os processos que forem disparados depois de outro podem reutilizar boa parte da informação armazenada no anel por este. Caso contrário talvez eles precisem recarregar boa parte do banco no anel, já que com o tempo trechos do anel vão sendo substituídos por outros novos.

Considere, por exemplo, dois processos BLAST concorrentes acessando um banco. Caso o intervalo de disparo entre eles fosse bem curto, quando um deles estivesse terminando de processar um bloco inteiro, provavelmente o outro também estaria terminando. Assim este bloco já poderia ser renovado e reusado por outros processos caso necessário.

Agora se o intervalo entre os disparos fosse longo, quando um processo estivesse terminando um bloco, outro poderia estar começando a processá-lo. Então quando um processo iniciado anteriormente precisar de alguma informação, este precisaria aguardar os que começaram posteriormente a terminarem de processar um bloco para que este possa ser atualizado.

E. Resultado dos testes

Posteriormente planejou-se partir para testes com bancos de sequências maiores. Porém quando um banco passa de 2GBytes, o BLAST o quebra em bancos menores de até 2GBytes e usa um arquivo *alias* [2] para os ler como se fossem um único banco. Como o BioProvider inicialmente foi criado se baseando que o banco tinha sempre apenas um volume, ele ainda não consegue processar bancos de tal tamanho.

Então foram feitos outros testes com um banco de tamanho menor que 2GBytes. Este banco (env_nr_5000000) foi criado com cinco milhões de sequências aleatórias extraídas do env_nr[3]. Os testes foram feitos com diferentes configurações, executados em diferentes máquinas.

Os gráficos abaixo indicam os resultados de alguns testes. Tanto no caso da base env_nr_5000000 (1.1Gbytes) quanto no caso do pataa (150Mbytes), um arquivo de 50 sequências extraídas aleatoriamente do banco que foi consultado é usado como arquivo de sequencias de consulta. Este arquivo é o mesmo para todos os testes. Onde se menciona "BLASTs concorrentes" se faz referência ao número de BLASTs que serão chamados, com um intervalo de 60 segundos entre cada disparo. O tempo entre disparos é tal que os novos processos não tenham a oportunidade de reutilizar muito as informações carregadas para processos mais antigos.

O teste com o pataa mostrado nesta seção é diferente do teste executado anteriormente, pois neste são usados outros parâmetros e o *script* de preparação é diferente do usado no teste inicial.

O eixo horizontal nos gráficos diz se o teste foi feito com o NCBI BLAST, versão 2.2.18, com a nova versão do BLAST, o NCBI BLAST+, versão 2.2.25, ou então com o NCBI BLAST 2.2.18 usando o BioProvider. Neste último caso, está indicado no eixo como o tamanho do anel de *buffer* usado.

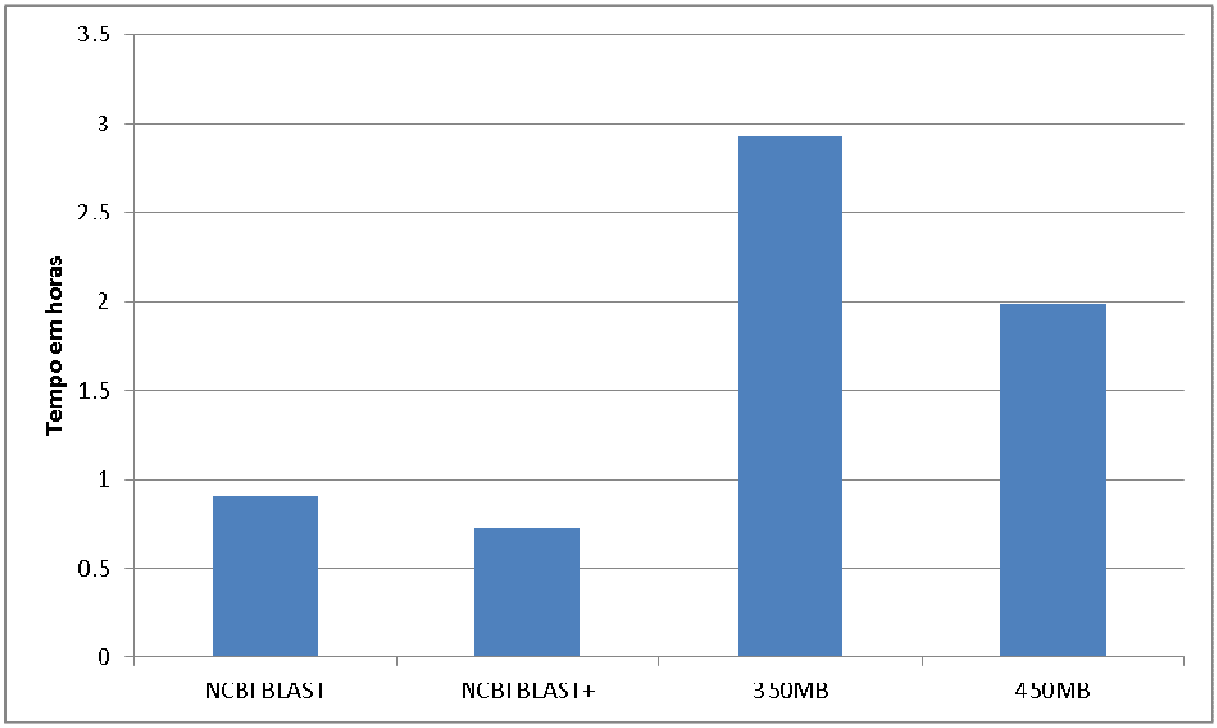

Figura 2: Máquina com 2GBytes de RAM e 10 BLASTs rodando concorrentemente contra o env_nr_500000.

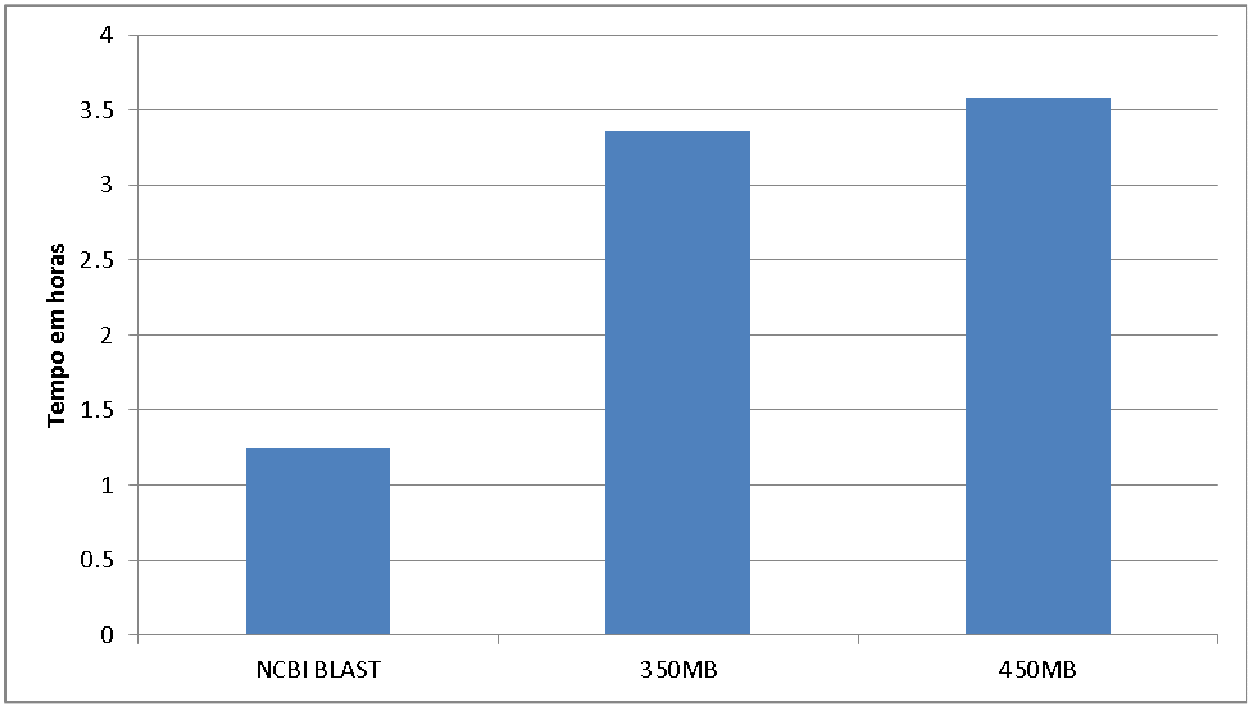

Figura 3: Máquina com 2GBytes de RAM e 50 BLASTs rodando concorrentemente contra o env_nr_500000.

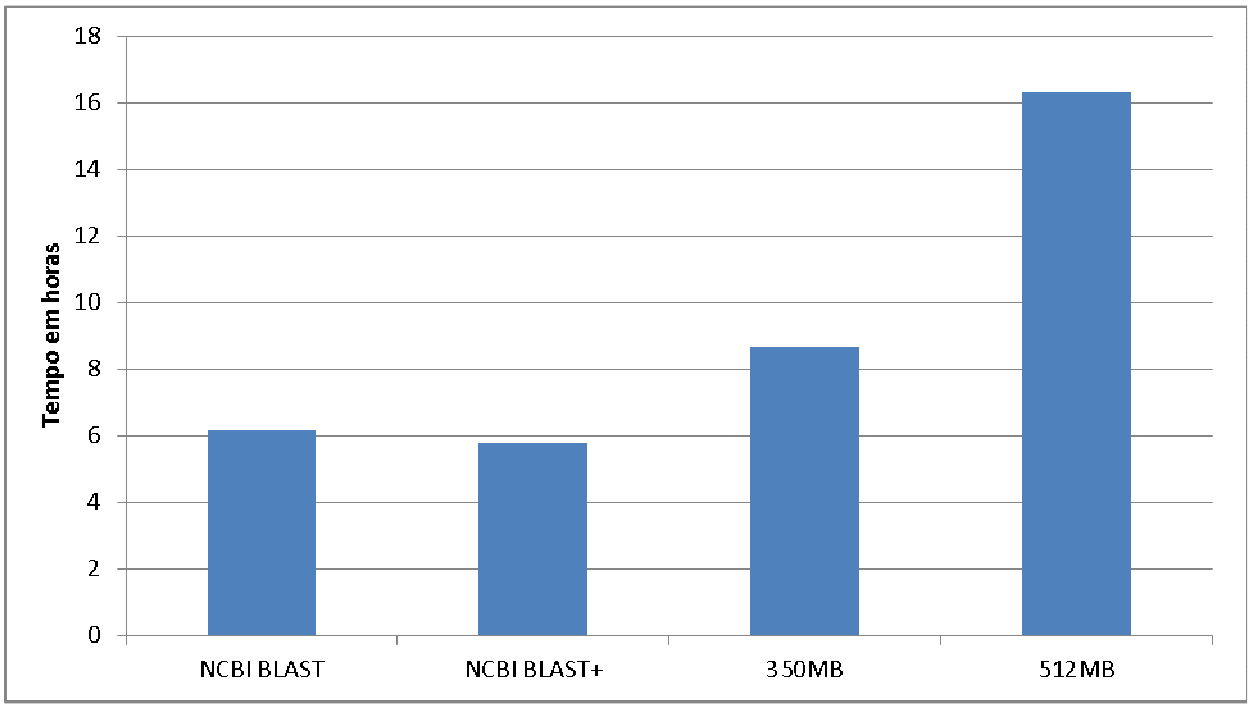

Figura 4: Máquina com 1GByte de RAM e 50 BLASTs rodando concorrentemente contra o env_nr_500000.

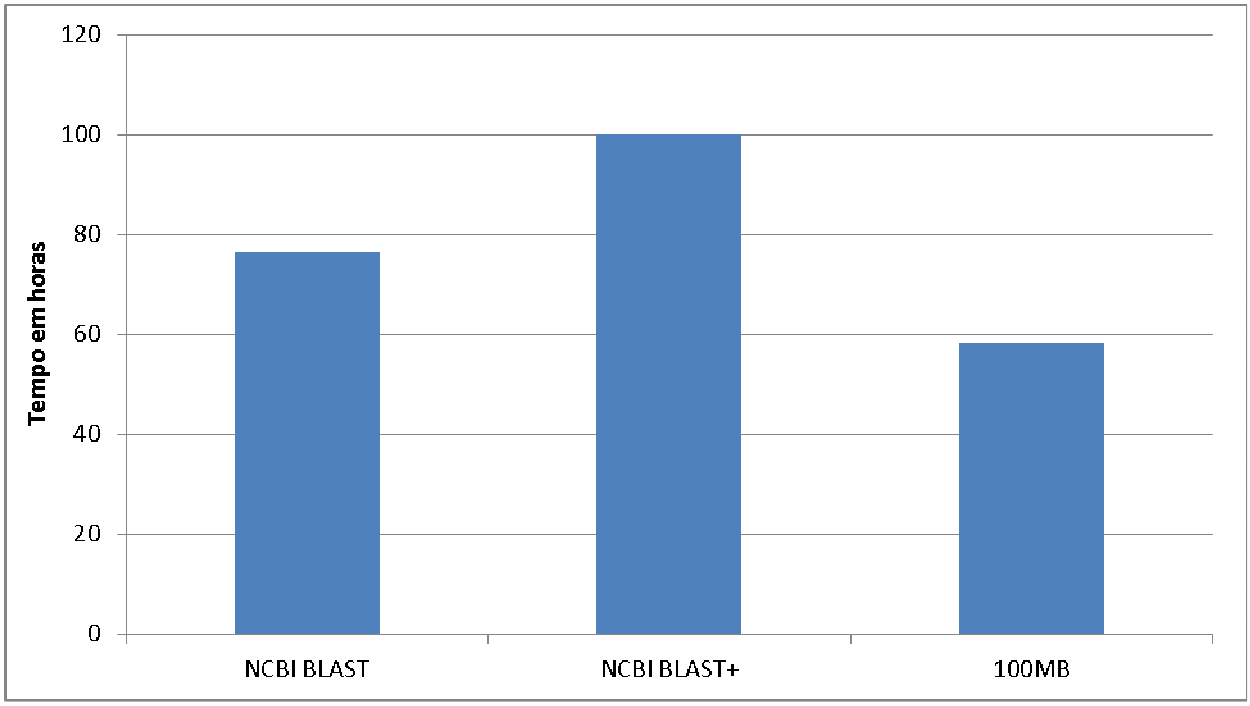

Figura 5: Máquina com 512MBytes de RAM e 50 BLASTs rodando concorrentemente contra o env_nr_500000, o teste com o BLAST+ foi interrompido após 100 horas passadas desde o seu início.

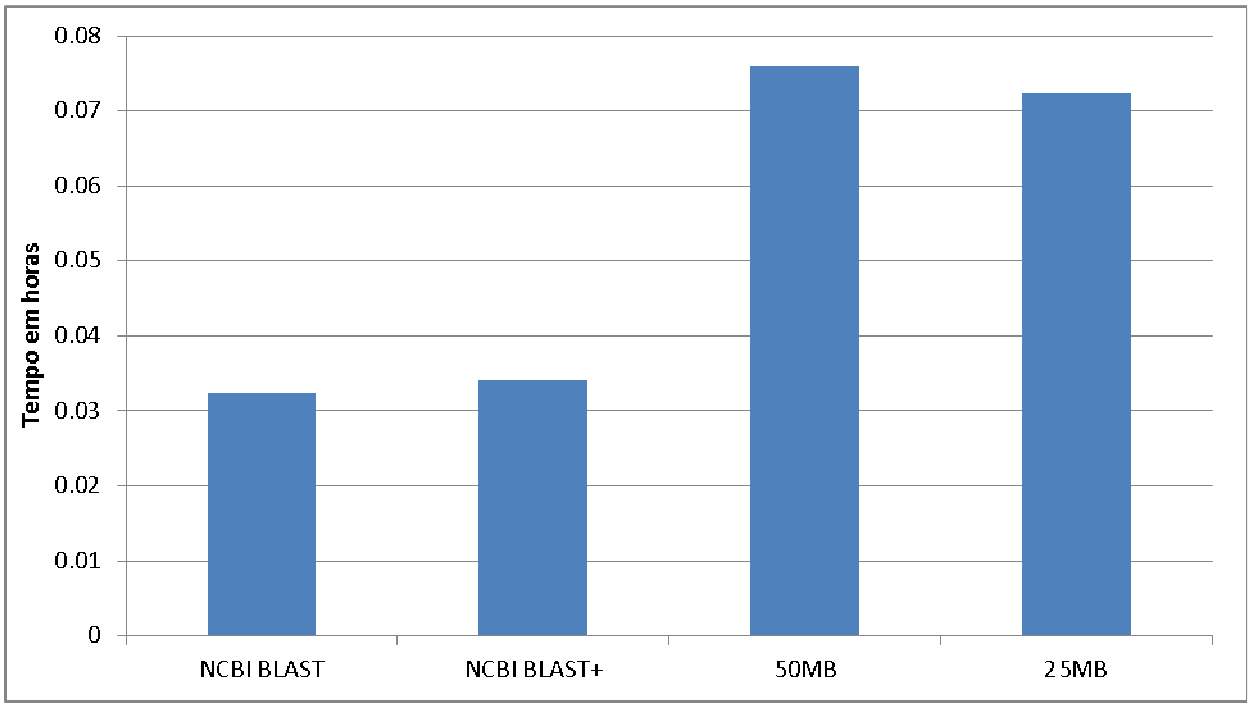

Figura 6: Máquina com 128MBytes de RAM e 10 BLASTs rodando concorrentemente contra o pataa.

Como se pode ver acima, o BioProvider teve um desempenho melhor que o NCBI BLAST em apenas um dos gráficos exibidos. Este resultado é relativo a um teste com a quantidade de memória RAM menor que o tamanho do arquivo de sequências.

Um dos motivos disto, além da quantidade de memória RAM ser suficiente para armazenar o arquivo de sequências inteiramente na memória, pode ser a configuração que foi usada nos *scripts* de preparação do BioProvider, de acordo com ela um bloco estaria ocupando o anel inteiro.

Com isto, muitos processos têm que disputar entre si para alterar o mesmo bloco. Os processos que terminarem de processar este bloco não terão alternativa a não ser aguardar até que todos os processos mais lentos também terminem de o ler para poder se atualizar o bloco. Assim a velocidade de execução do BLAST seria reduzida devido a este tempo de espera.

Por causa disto o *script* de preparação do BioProvider foi modificado para, só para estes testes, dividir o anel em quatro blocos. Um dos problemas desta mudança com o BioProvider atual é que devido à necessidade de se fazer as permutações, considerando um anel de 100MBytes e um arquivo de sequências de 940MBytes, seriam precisos 40 permutações do banco de sequências que contêm este arquivo. Logo para se fazer isto com estas configurações foi preciso gastar mais de 40GBytes de HD e umas 3~4 horas de processamento em uma máquina com um processador de quatro núcleos e 512MBytes de memória RAM.

Outro problema que aconteceu na única máquina que tinha espaço no disco rígido suficiente para estas permutações foi o de os processos BLAST "pararem" de trabalhar. Ao se olhar no gerenciador de tarefas, pode-se notar que a quantidade de uso de recursos do processador por estes processos se tornaram zero. Quando o BioProvider é finalizado, estes processos voltam a trabalhar e algum tempo depois exibem uma mensagem de erro no lugar de exibir o resultado pois o local de onde o banco estava sendo lido deixou de ser acessível.

Analisando este comportamento, aparentemente este é um problema de alocação de recursos. Ou seja, por alguma razão os processos BLASTs não recebem a informação que necessitam e ficam "eternamente" esperando por ela, sem fazer nada. Devido a isto não foi possível fazer um teste com mais blocos na memória para este relatório.

Conclusões

Com estes testes se pode confirmar que o que foi dito durante o estudo da criação do provedor de dados continua válido em máquinas mais atuais. Também se confirmou que o BioProvider e o seu driver podem funcionar em versões anteriores de *kernel* (9.1 e 9.04).

O uso do BioProvider é realmente proveitoso quando o banco de sequências a ser consultado não cabe inteiramente na memória, pois a manipulação genérica oferecida pelo sistema operacional não pode otimizar o acesso a estas informações como o BioProvider faz. Caso contrário o BioProvider tornaria a execução do BLAST mais lenta do que a sua execução normal. Também é preciso saber escolher corretamente os parâmetros para a execução do BLAST em cima do BioProvider, como o número de blocos na memória e o tamanho do anel de *buffer*.

Seria bom para seu uso efetivo encontrar e corrigir o problema de alocação de recursos que ocorre em algumas ocasiões, encontrar uma maneira de se dispensar a necessidade de permutar o banco de sequências e fazer uma pequena mudança no *script* de preparação para que se possa escolher a quantidade de blocos existentes dentro de um anel.

Referências

1 - NORONHA, Maíra Ferreira de. **Controle da Execução e Disponibilização de Dados para Aplicativos sobre Seqüências Biológicas: o Caso BLAST.** Rio de Janeiro, 2007. 83p. Dissertação de Mestrado, Departamento de Informática, PUC-Rio.

2 – KORF, Ian; YANDELL, Mark; BEDELL, Joseph. **BLAST**. O'Reilly Media, 2003, 368pp.

3 – http://www.c2b2.columbia.edu/infrastructure/NCBI_BLAST_DB.htm

4 – http://www.ncbi.nlm.nih.gov/BLAST/docs/formatdb.html

5 – CORBET, Jonathan; RUBINI, Alessandro; KROAH-HARTMAN, Greg. ***Linux Device Drivers, Third Edition***. O'Reilly Media, 2005, Cap. 1-6.

6 – CAMACHO, Christian; COULOURIS, George; AVAGYAN, Vahram; MA, Ning; PAPADOPOULOS, Jason; BEALER, Kevin; MADDEN, Thomas. **BLAST+: architecture and applications.** BMC Bioinformatics, 2009, 9pp.