7.

Referências bibliográficas

1 TALBI, E.; ZOMAYA, A. **Grid Computing for Bioinformatics and Computational Biology**. New Jersey, USA: Wiley, 2008. 392 p.

2 MANNHOLD, R.; KUBINYI H.; TIMMERMAN H. **Bioinformatics – From Genomes to Drugs**. Verlag GmbH, Weinhein (Federal Republic of Germany): WILEY-VCH, 2002. 668p.

3 KORF, I.; YANDELL M.; BEDELL, J. **BLAST**. Sebastopol, CA, USA: O'Reilly, 2003; 339p

4 GARDNER, M.; FENG, W.; ARCHULETA, J.; LIN, H.; XIAOSONG, M. Parallel Genomic Sequence-Searching on an Ad-Hoc Grid: Experiences, Lessons Learned, and Implications**.** In: **Proceedings of the ACM/IEEE Supercomputing 2006 Conference**, 2006, article 104.

5 BRAUN, R., PEDRETTI K., CASAVANT, T., SCHEETZ, T., BIRKETT C., ROBERTS, C. Parallelization of local BLAST service on workstation clusters. In: **Future Generation Computer Systems,** Volume 17, Issue 6, 2001, pp 745-754.

6 TANEMBAUM, A. **Redes de Computadores**. Rio de Janeiro: Campus, 1997. 968p.

7 EL-HEWINI, H.; ABD-EL-BARR, M. **Advanced Computer Architecture and Parallel Processing**. New Jersey, USA: Wiley, 2005. 288p.

8 GRID CAFÉ. What is "The Grid". Disponível em: http://gridcafe.web.cern.ch/gridcafe/whatisgrid/whatis.html. Acessado em: 22 set. 2008.

9 RIEFFEL, M.; GILL, T.; WHITE R. Bioinformatics Clusters in Action. Disponível em: http://www.paracel.com/pdfs/clusters-in-action.pdf. Acessado em 10 set. 2008.

PUC-Rio - Certificação Digital Nº 0711318/CA

10 LIFSCHITZ, S.; VALDURIEZ, P.; SOUZA D. BLAST Distributed Execution on Partitioned Databases with Primary Fragments. In: **Proceedings of the 8th International Conference – VECPAR 2008,** Toulouse, France, VECPAR 2008, 2008, pp 544-554.

11 UOL. Dicionário Houaiss. Disponível em: http://educacao.uol.com.br/dicionarios/. Acessado em: 10 nov. 2008.

12 BABYLON. Dicionário Babylon 7. Disponível em : http://www.babylon.com/definition/funcionalidade/Portuguese. Acessado em: 10 nov. 2008.

13 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 9126-1:2003** - Engenharia de software - Qualidade de Produto Parte 1: Modelo de Qualidade. Rio de Janeiro, 2003.

14 CAMPBELL, M.; DARGAN, K.; ELKIN, C.; GIBBS, R.; HAY, C.; HICKS, D.; JASPIN, M.; NEBURKA, J,; NICE, M.; WU, L.; CALDWELL, S.; BISERS, D. **Performance Measurement Manual for FY 2008.**

15 HETZEL, B. **Making Software Measurement Work: Building an Effective Measurement Program**. New Jersey, USA: Wiley, 1993. 304p.

16 PANDIAN C. **Software Metrics – A Guide to Planning, Analysis, and Aplication.** USA**:** Auerbach Publications, 2004. 312p.

17 FENTON, N.; PFLEEGER, S. **Software Metrics**. Boston, USA**:** PWS Publishing Company, 1997**.** 638p.

18 DARLING, A.; CAREY, L.; FENG, W. The Design, Implementation, and Evaluation of mpiBLAST**.** In: **Proceedings of ClusterWorld Conference & Expo and the 4th International Conference on Linux Clusters**: The HPC Revolution 2003, 2003.

19 SOUSA, D. **Estratégias de Balanceamento de Carga para Avaliação paralela do BLAST com Bases de Dados Replicadas e Fragmentos Primários.** 2007, 85p. Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de janeiro, 2007.

20 ANDRADE, J.; BERGLUND, S.; UHLEN, M.; ODEBERG, J. Grid-BLAST: A Grid implementation of the Sliding Window Algorithm Facilitates Whole Proteome Similarity Analysis. Disponível em: http://kthgridproxy.biotech.kth.se/grid_blast/index.html. Acessado em 01 jun. 2008.

PUC-Rio - Certificação Digital Nº 0711318/CA

21 KUMAR, S.; JOSHI, R. GBTK: A Toolkit for Grid Implementation of BLAST. In: **Proceedings of the High Performance Computing and Grid in Asia Pacific Region**, 2004, pp 378 – 382.

22 BAYER, M.; CAMPBELL, A.; VIRDEE, D. A GT3 Based BLAST Grid Service for Biomedical Research. In: **Proceedings of the UK e-Science All Hands Meeting**, 2004, pp 1019 - 1023.

23 SOUZA, M.; MELO, A. PackageBLAST: An Adaptive MultiPolicy Grid Service for Biological Sequence Comparison. In: **Proceedings of the 2006 ACM Symposium on Applied Computing**, 2006, pp 156 – 160.

24 CARVALHO, P.; GLÓRIA, R.; MIRANDA, A.; DEGRAVE, W. Squid – a Simple Bioinformatics Grid. **BMC Bioinformatics**, 2005, vol. 6: p 197.

25 KIM, T.; OH, S.; LEE, K.; ROH, D.; CHO, W. HGBS : A Hardware-Oriented Grid BLAST System. In: **Proceedings of the Cluster Computing and the Grid, 2005. CCGrid 2005.** IEEE International Symposium**,** 2005, vol. 1: pp 520 - 526.

26 DOWD, S.; ZARAGOZA, J.; RODRIGUEZ, J.; OLIVER, M.; PAYTON P. Windows .NET Network Distributed Basic Local Alignment Search Toolkit (W.ND-BLAST). In: **BMC Bioinformatics**, 2005, vol. 6: p 93.

27 ALBAYRAKTAROGLU, K.; JALEEL, A.; WU, X.; FRANKLIN, M.; JACOB, B.; TSENG, C.-W.; YEUNG, D. BioBench: A Benchmark Suite of Bioinformatics Applications**. Proceedings of the 2005 IEEE International Symposium on Performance Analysis of Systems and Software - ISPASS 2005**, 2005, pp. 2-9.

28 GARDNEN, M.; ARCHULETA, J.; LIN, H.; MA, X.; FENG W. Parallel Genomic Sequence-Searching on an Ad-Hoc Grid: Experiences, Lessons Learned, and Implications. **Proceedings of the 2006 ACM/IEEE conference on Supercomputing**, 2006, pp 104.

29 YANG, C.; HSIUNG, Y.; KAN, H. Implementation and Evaluation of a Java Based Computational Grid for Bioinformatics Applications**.** In: **Proceedings of the 19th International Conference on Advanced Information Networking and Applications**, Taipei, Taiwan. IEEE Computer Society 2005, 2005, volume 1: pp 298- 303.

30 BAYER, M.; SINOT, R. Distributed BLAST in a Grid Computing Context. In: **Lecture Notes in Computer Science, 2005, Computational Life Sciences, First International Symposium, CompLife 2005**, Konstanz, Germany, 2005, vol. 3965: pp 241-252.

PUC-Rio - Certificação Digital Nº 0711318/CA

31 NOGUEIRA, L.; YAMIN, A.; VARGAS, P.; GEYER, C. Balanceamento de Carga em Sistemas Distribuídos: Uma Proposta de Ambiente para Avaliação. Disponível em: http://www.inf.ufrgs.br/~kayser/papers/clei2001-robinhood/RH.pdf. Acessado em: 15 nov. 2008.

32 NORONHA, M. **Controle da Execução e Disponibilização de Dados para Aplicativos sobre Seqüências Biológicas: o Caso BLAST.** 2006, 83p**.** Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, 2006.

33 BRYAN, L. The TPC-C database benchmark -- What does it really mean?. Disponível em: http://sunsite.uakom.sk/sunworldonline/swol-08-1997/swol-08-database.html. Acessado em 18 mar. 2008.

34 NCBI. BLAST. Disponível em: http://www.ncbi.nlm.nih.gov/BLAST. Acessado em:18 mar. 2008.

35 WU-BLAST. Disponível em: http://blast.ustl.edu. Acessado em: 18 mar 2008.

36 FASTA. Disponível em: http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml. Acessado em: 18 mar. 2008.

37 KOPPER, K. **The Linux Enterprise Cluster. Build a Highly Available Cluster with Commodity Hardware and Free Software**. San Francisco, CA, USA: No Starch Press, 2005. 464p.

38 VRENIOS, A. **Linux Cluster Architecture**. Indianapolis, Indiana, USA: Sams Publishing; 2002. 264p.

39 SOTOMAYOR, B.; CHILDERS, L. **Globus Toolkit 4** – Programming Java Services. San Francisco, CA, USA: Elsevier, 2006. 506p

40 UPDATERS. Large Hadron Collider. Disponível em : http://updateordie.com/updates/geral/2008/02/large-hadron-collider. Acessado em: 25 mar. 2008

41 FOSTER, I.; KESSELMAN, C.; TUECKE, S.; NICK, J. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems. Disponível em: http://www.globus.org/research/papers.html. Acessado em 10 jun. 2008.

PUC-Rio - Certificação Digital Nº 0711318/CA

42 FOSTER, I.; KISHIMOTO, H.; SAWA, A.; BERRY, D.; DJAOUI, A.; GRIMSHAW, A.; HOM, B.; MACIEL, F.; SIEBENLIST, F.; SUBRAMANIAM, R.; TREADWELL, J.; REICH, J. The Open Grid Services Architecture version 1.5. Global Grid Forum, Lemont, Illinois, U.S.A., 2006. Disponível em: http://www.ggf.org/documents/GFD.80.pdf. Acessado em 07 mar. 2009.

43 TUECKE, S.; CZAJKOWSKI, K.; FOSTER, I.; FREY J.; GRAHAM, S.; KESSELMAN, C.; MAQUIRE, T.; SANDHOLM, T.; SNELLING, D.; VANDERBIT, P. Open Grid Services Infrastructure (OGSI). Global Grid Forum 2003. Disponível em: http://www.ggf.org/documents/GFD.30.pdf. Acessado em: 07 mar. 2009.

44 PROKOSCH, T. Status of Grid Middleware and Corresponding Emerging Standards for Potential Usage in Sharing Scientific Instruments via (International) Networks. Disponível em http://www.ringrid.eu/public/deliverables/RINGRID-WP3-D3_2-JKU-Middleware5.pdf. Acessado em: 7 mar. 2009.

45 ANDRADE, J. **Grid and High Performance Computing for Applied Bioinformatics**; 2007, 47p, Tese (Doutorado), Royal Institute of Technology, School of Biotechnology, Stockholm, 2007.

46 DOUGLIS, F.; FOSTER, I. The Grid Grows Up. **IEEE Internet Computing**, jul./ago. 2003, Guest Editors' Introduction. Disponível em: http://www2.computer.org/portal/web/csdl/doi/10.1109/MIC.2003.1215656. Acessado em: 07 mar. 2009.

47 COSTA , R.; **Alocação de Dados e Distribuição de Carga para Execução Paralela da Estratégia BLAST de Comparação de Seqüências**; 2002, 75p. Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, 2002.

48 LASZEWSKI, G.; FOSTER,I., GAWOR, J., LANE P.; A Java Commodity Grid Kit, Concurrency and Computation: Practice and Experience. **ACM Java Grande Conference 2000**, vol.13, number 8 e 9: pp 645- 662, 2001.

49 SOARES, Luiz Fernando Gomes ; LEMOS, G. ; COLCHER, S. **Redes de Computadores: Das LANs, MANs e WANs às Redes ATM**. Rio de Janeiro, Campus, 1995, vol. 1, 700 p.

PUC-Rio - Certificação Digital Nº 0711318/CA

50 Spendolini, M. J. **The Benchmarking Book**. New York, USA**:** Amacom Press, 1992, 256p.

PUC-Rio - Certificação Digital Nº 0711318/CA

Apêndice 1

A Ferramenta gradeBLAST

O crescimento das bases de dados biológicas motivou vários estudos que levaram à implementação de ferramentas capazes de executar o BLAST em ambientes distribuídos, de modo a tornar a sua execução mais eficiente.

Basicamente existem duas maneiras de tornar a execução do BLAST paralela: a primeira consiste em replicar uma base de dados pelos nós de um *cluster* ou de um *grid*, dividir as consultas entre esses nós, e submetê-las; a segunda consiste em segmentar uma base de dados, alocar cada um dos segmentos a um determinado nó de um Cluster ou de um Grid, e submeter todas as consultas a todos os nós.

O gradeBLAST foi criado para permitir a paralelização da execução do BLAST utilizando a replicação da base de dados pelos nós de um Grid, que utiliza o *Globus Toolkit* como *middleware*, ou ainda de um *cluster* que possua o *Globus* instalado. Ele foi baseado em dois trabalhos: o primeiro foi o Grid-Blast que foi criado a partir de um estudo efetuado no *Royal Institute of Technology* [35]; o segundo foi o Squid [39], desenvolvido no Rio de Janeiro pelo Instituto Oswaldo Cruz.

O presente texto tem como objetivo descrever o funcionamento do gradeBLAST.

O gradeBLAST

Escrito em bash, o gradeBLAST é composto por um programa que roda em um dos nós de um *grid* ou de um *cluster*, e é responsável por submeter os serviços para os diversos nós, verificar se os serviços estão prontos, e ao final, elaborar um

PUC-Rio - Certificação Digital Nº 0711318/CA

relatório contendo os resultados da execução do BLAST. Adicionalmente utiliza alguns scripts que tem por objetivo facilitar chamadas ao *Globus Toolkit*.

O programa, em si, é composto por cinco módulos que são responsáveis por: verificar as condições de execução; montar os arquivos contendo as consultas a serem submetidas; verificar a existência de nós disponíveis para submissão de um serviço; submeter os serviços para os nós disponíveis; e consolidar o relatório com os resultados obtidos pela execução do BLAST nos diversos nós.

Um mecanismo de tolerância a falhas está também disponível, de modo a possibilitar a re-execução de qualquer serviço em que haja ocorrido algum problema que possa ser sinalizado pelo *Globus*. Não é possível a verificação de erros de execução do BLAST, e por isso os resultados obtidos devem ser verificados posteriormente.

O módulo de verificação existente no programa efetua uma checagem que considera: a existência de um arquivo contendo a identificação dos nós em que o BLAST será rodado; a existência, em cada um dos nós, do diretório contendo a base formatada; a formatação da base de dados em cada um dos nós; a existência de um arquivo contendo as seqüências de consulta no nó mestre (aquele em que o programa é rodado); e o formato FASTA das seqüências de consulta.

O módulo de execução do BLAST é responsável por submeter aos nós as consultas desejadas, de acordo com a disponibilidade de nós livres para a execução. A verificação de nós livres é feita pela indicação de um código de retorno do *Globus Toolkit*. Na realidade este código indica se uma execução do BLAST foi efetuada com sucesso ou com algum erro. Caso a execução tenha terminado com êxito aquele nó é disponibilizado para um novo serviço, senão um mecanismo de re-submissão é disparado em outro nó, e o nó que apresentou o problema não é mais utilizado. Para a submissão de serviços, as consultas são montadas em tarefas, sendo que cada tarefa constitui um arquivo contendo um número de seqüências pré-definido como parâmetro na linha de comando. Cada serviço é composto por uma tarefa que é submetida a um determinado nó identificado como livre.

PUC-Rio - Certificação Digital Nº 0711318/CA

O módulo de consolidação de resultados simplesmente efetua a concatenação dos resultados obtidos pela execução do BLAST em cada um dos nós. Para isso, os resultados são movidos para o nó mestre, e depois concatenados. Ao contrário das estratégias fragmentadas não há necessidade de qualquer acerto nos resultados obtidos.

Complementando o programa foram elaborados alguns scripts que tem por finalidade tornar menos confuso o código quando efetuando chamadas ao *Globus Toolkit*. Basicamente são utilizados quando copiando arquivos entre os nós do *grid* ou do *cluster*., ou quando da submissão de serviços aos nós remotos. Esses comandos normalmente fazem alusões a endereços URL pertencentes aos diversos nós, e tornam os comandos de cópia muito grandes, podendo ocupar com facilidade duas ou mais linhas, o que pode tornar o código ilegível.

Na listagem abaixo são apresentados os passos executados pela ferramenta gradeBLAST durante a sua execução. A linha de comando para a sua execução contém os seguintes parâmetros: o arquivo que contém os equipamentos do *cluster* ou do *grid* em que o BLAST será executado; a base de dados biológica; o programa BLAST que será executado; o arquivo que contém as seqüências de consulta; o nome do arquivo de resultado; e o número de seqüências incluídas em cada tarefa submetida a um determinado nó.

A listagem mostra ainda as informações que são apresentadas durante a execução. É possível observar, a partir das informações fornecidas que, inicialmente, os dados fornecidos na linha de comando são verificados, para saber se a execução nos diversos nós será bem sucedida. Em seguida, as tarefas começam a ser distribuídas pelos nós do *cluster* ou do *grid*, e a execução do BLAST é então iniciada. em cada um dos nós. A medida que as tarefas são prontificadas, novas tarefas são distribuídas, até que não existam mais tarefas a executar. Ao final os resultados obtidos em cada um dos nós são consolidados em um único relatório, que constitui o resultado final da execução.

PUC-Rio - Certificação Digital Nº 0711318/CA

```
[pgomes@n00 ~]$./gradeBLAST bootGradeBLAST nr blastp SQ_RP_NR_500_500 ResulBlast
24
Validando Dados ...
Geral ...
Hosts ...
n01 ...
n02 ...
n04 ...
n05 ...
Iniciando Blast ...
Executando ...
tarefa 0 submetida n01 ...
tarefa 1 submetida n02 ...
tarefa 2 submetida n04 ...
tarefa 3 submetida n05 ...
tarefa 4 submetida n04 ...
tarefa 5 submetida n05 ...
tarefa 6 submetida n01 ...
tarefa 7 submetida n02 ...
tarefa 8 submetida n05 ...
tarefa 9 submetida n04 ...
tarefa 10 submetida n01 ...
tarefa 11 submetida n02 ...
tarefa 12 submetida n01 ...
tarefa 13 submetida n05 ...
tarefa 14 submetida n04 ...
tarefa 15 submetida n02 ...
tarefa 16 submetida n01 ...
tarefa 17 submetida n05 ...
tarefa 18 submetida n04 ...
tarefa 19 submetida n02 ...
tarefa 20 submetida n01 ...
Execucao do Blast  Terminada ...
Processando Resultados ...
Consolidando Resultados ...
Programa Encerrado
```

Listagem 1: *Execução do gradeBLAST.*

PUC-Rio - Certificação Digital Nº 0711318/CA

Alguns arquivos de *log* são gravados durante a execução do gradeBLAST, que fornecem dados estatísticos sobre a execução. São eles:

a) *gradeBLAST_Verificacao*, que fornece o tempo que a ferramenta levou para analisar os dados fornecidos em linha de comando;

b) *gradeBLAST_Blast*, que fornece o tempo de execução total da ferramenta BLAST, incluindo todos os nós em que foi executado.

c) *gradeBLAST_Preparacao*, que fornece o tempo total para efetuar as movimentações de arquivos de resultados dos nós para o nó mestre;

d) *gradeBLAST_Consolidacao*, que fornece o tempo para concatenar os

diversos relatórios contendo os resultados das execuções do BLAST, produzindo um único relatório.

e) *gradeBLAST_Execucao*, que fornece o tempo total de execução do gradeBLAST;

f) *gradBLAST_Tarefas_Nó*, que fornece o número de tarefas executadas por cada um dos nós.

Conclusão

O gradeBLAST não foi desenvolvido com a finalidade específica de se tornar uma ferramenta BLAST em paralelo para a utilização em pesquisas reais, mas sim como um esforço de pesquisa no sentido de tornar possível a utilização de uma ferramenta simples para avaliação com ferramentas que utilizam uma estratégia fragmentada, como é o caso do balaBLAST e do mpiBLAST. No entanto, apesar de não ser tão eficiente quanto outras ferramentas, pela sua simplicidade e fácil transformação do código para utilização sem a exigência de uma implementação do MPI ou do GLOBUS, em ambientes de cluster, pode tornar-se uma opção atraente em alguns casos.

PUC-Rio - Certificação Digital Nº 0711318/CA

Apêndice 2

Alguns Tópicos sobre Clusters e Grids

Ferramentas BLAST, *Basic Local Alignment Search Tool*, são hoje amplamente empregadas na comparação entre seqüências desconhecidas e conhecidas de Proteínas, DNA e RNA, na busca por similaridades que apontem para homologias ou predições de funções de genes [3].

No entanto, o crescimento exponencial das bases de dados biológicas, tem causado uma grande preocupação quanto ao desempenho dessa ferramenta. Alternativas, tais como a otimização do gerenciamento de *buffer* em memória, como, por exemplo, o BioProvider [32], ou a paralelização, utilizada pelo mpiBlast [18], constituem soluções que podem ser utilizadas no sentido de prover uma maior capacidade de processamento e, assim, acelerar a sua execução.

A paralelização do BLAST tem por objetivo dividir a sua execução entre vários recursos computacionais disponíveis, incrementando, assim, o seu desempenho. Para isso, atualmente, utilizam-se *cluster*s que interligam computadores localmente utilizando um sistema de passagem de mensagens, tal como o MPI e PVM, ou então *grids*, que consideram recursos computacionais geograficamente espalhados interligados através a utilização de um *middleware*, tal como o *Globus Tookit*, capaz de prover o seu gerenciamento.

O objetivo do presente texto é apresentar alguns tópicos concernentes à tecnologia de *clusters* e *grids*, visando o melhor entendimento quanto à utilização do BLAST nestes ambientes.

Alguns Tópicos Sobre Clusters

De uma forma bem simplificada, pode-se dizer que um *cluster* é um grupo de computadores que roda como se fosse apenas um [37, 38]. Esta definição

PUC-Rio - Certificação Digital Nº 0711318/CA

admite que um *cluster* possa ser utilizado como um recurso unificado, utilizando um sistema computacional local, que compreende um conjunto de computadores independentes, conectados através uma rede. Assim, um *cluster* deve ser "transparente" a quem o esteja utilizando, incluindo usuários, programadores e até outros nós do Cluster.

Em [37] são descritas quatro propriedades que um *cluster* deveria ter, à luz da definição acima:

a) Um usuário não deve saber que está utilizando um *cluster*:

   Se for diferente, ele utilizará um *cluster* como se fosse uma rede, e não como um único computador;

b) Os Nós em um *cluster* não devem "saber" que pertencem a um *cluster*:

   Ou seja, não é necessário modificar o sistema operacional dos equipamentos para que eles rodem como um *cluster*. Além disso, uma falha em qualquer dos nós de um *cluster* não deve influenciar o processamento de uma tarefa;

c) As aplicações não devem saber que estão executando em um *cluster*:

   Uma aplicação não deve ser modificada para rodar em um *cluster*, caso contrário, a aplicação não enxerga o *cluster* como um único recurso;

d) Servidores em um *cluster* não devem saber que estão servindo um nó do *cluster:*

   Assim, servidores de impressão, *email*, etc., não necessitam ser reescritos para rodar em um *cluster.*

O que se deseja, ao montar uma estrutura de *cluster* com vários computadores individuais, é obter um maior desempenho e confiabilidade do que seria possível em apenas um computador. No entanto para alcançar esses objetivos três são os atributos necessários [38]:

a) Escalabilidade:

   Quanto maior o número de nós maior será o desempenho obtido;

PUC-Rio - Certificação Digital Nº 0711318/CA

b) Disponibilidade:

A existência de falhas em alguns dos nós de um *cluster* não significa a parada do serviço, mas apenas uma degradação no desempenho;

c) Tolerância a Falhas:

O conjunto de nós do *cluster* compartilha a responsabilidade pela execução de um serviço. Caso algum dos nós venha a ter alguma falha os outros nós não devem ser afetados. Ademais, algum sistema monitor pode reiniciar o serviço naquele nó, ou atribuir a outro nó a sua execução.

No entanto quando a Escalabilidade é abordada, a realidade mostra que o incremento no desempenho é limitado pelo *overhead* provocado pela latência nas comunicações efetuadas entre computadores [38]. A fixação de um limite é um processo puramente interativo, em que a cada adição o ganho de performance tem que ser avaliado.

A figura 1 mostra a arquitetura de um *cluster* homogêneo, que é constituído por equipamentos de igual porte, que utilizam um mesmo sistema operacional.

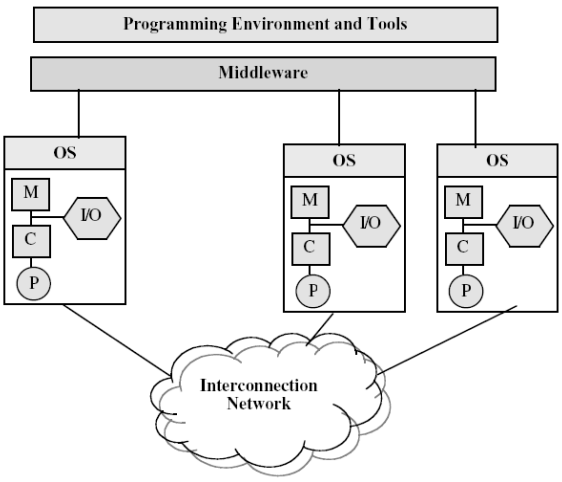

Figura 1: *Arquitetura de um cluster homogêneo [7]*.

PUC-Rio - Certificação Digital Nº 0711318/CA

Pode-se observar na figura uma camada que constitui o *middleware*. O *middleware* possibilita que um *cluster* pareça ao usuário uma única máquina paralela que é denominada *Single System Image* – SSI, que entre outras finalidades, permite que um usuário tenha acesso ao *cluster* como um todo, e não a um nó em particular [7].

Além do *middleware*, existe outra camada que aparece na figura, e que constitui o ambiente de programação. Esse ambiente tem por finalidade prover as ferramentas e bibliotecas para o desenvolvimento de aplicações paralelas, como o *Message Passing Interface* – MPI, e o *Parallel Virtual Machine* – PVM [8].

Assim como os computadores mais modernos, *clusters* são capazes de efetuar "*overlap*". Entende-se por *overlap* a capacidade de execução de mais que uma função simultaneamente [38]. Dessa maneira conseguem alcançar um substancial incremento de desempenho, principalmente se comparado ao acréscimo que pode ser obtido utilizando-se um equipamento isolado. Este fato pode ser considerado como a essência da utilização paralelismo em um *cluster.*

O paralelismo é, portanto, possível à medida que, em uma determinada tarefa, existam porções que permitam a sua execução em paralelo. Caso existam, essas porções podem ser processadas em diferentes nós do *cluster*. Após o término do processamento, os resultados obtidos em cada nó podem ser combinados a partir de um processamento serial [38].

Várias aplicações, como por exemplo, o BLAST, permitem o paralelismo.

Alguns Tópicos Sobre Grids.

De um modo geral, *grid* pode ser entendido como um serviço destinado ao compartilhamento do poder computacional e capacidade de armazenamento de dados de equipamentos diversos, sobre a internet, com o objetivo de tornar a rede mundial de computadores em um vasto recurso computacional [8].

PUC-Rio - Certificação Digital Nº 0711318/CA

Outras definições sobre o que seja um *grid* podem ser ainda citadas: um conjunto de *clusters* independentes, em que cada um deles constitui um nó de um supercomputador virtual, cujos recursos espalham-se dentro de uma organização; ou ainda, um supercomputador virtual que utiliza recursos geograficamente dispersos focados em uma atividade científica, matemática ou acadêmica.

Um sistema de *grid* ideal deveria considerar a utilização de milhões de equipamentos, incluindo: estações de trabalho, *mainframes*, supercomputadores, grandes repositórios de dados, além de outros, tais como telescópios e sensores meteorológicos, ao redor do mundo, pertencentes a muitas diferentes pessoas ou organizações, conectados via internet, atuando em conjunto, como um "super" poderoso computador. Assim, caso alguém necessitasse executar uma aplicação, não teria, necessariamente, que usar a sua própria máquina, poderia fazê-lo em outra, e o próprio grid poderia se encarregar de verificar a melhor localização, instalar naquele local e rodar o programa [8].

Vantagens e desvantagens podem advir da utilização de um Grid, Uma grande vantagem é que cada nó pode ser adquirido como uma *commodity*, ou seja, um computador comum, mas que quando combinado em um conjunto de nós, pode produzir recursos similares a um supercomputador multiprocessado, porém a um baixo custo. Além disso, a escalabilidade de *grids* dispersos geograficamente é geralmente favorável, principalmente em razão da internet. Entre as desvantagens pode-se citar a diferença de performance entre as conexões, dificuldade de desenvolver programas que possam ser executados em um ambiente de supercomputador, e a confiabilidade das computações efetuadas em virtude de um possível mau funcionamento ou então de utilizadores maliciosos.

Um exemplo bastante citado na literatura diz respeito ao *Large Hadron Collider* (LHC) [39], que é um grande acelerador e "colisor" de partículas. O LHC é um anel de aproximadamente 16,7 milhas e nele duas partículas são aceleradas a velocidades próximas a da luz, em direções opostas, e então colidem, gerando energia e outras partículas, muitas delas desconhecidas [40]. As informações produzidas pelo equipamento alcançam a ordem de 10 Petabytes (cada Petabyte equivale a 10^{7} Gigabytes) no decorrer de um ano. Evidentemente, com a tecnologia atual poder-se-ia dizer impossível processá-los em apenas um

PUC-Rio - Certificação Digital Nº 0711318/CA

site (um local com capacidade de executar aplicações por si só). A solução encontrada foi efetuar o processamento dessas informações em um *grid*, incluindo inúmeras organizações na Europa, de modo a satisfazer os requisitos exigidos de poder computacional e capacidade de armazenamento.

Um dos conceitos mais importantes quando se trata de *grids* é o da virtualização. A virtualização consiste basicamente na utilização de recursos de outras organizações, e que, portanto, não possuem controle local, e que, no seu conjunto, formam uma organização virtual [39]. A virtualização permite que se tenha uma visão unificada dos recursos, como se fossem locais, embora sabidamente eles estejam geograficamente dispersos. Dessa maneira, o usuário tem a ilusão de estar um grande computador virtual capaz de executar enormes tarefas [7].

Pelo que foi exposto, pode-se concluir que, o que se procura, ao utilizar um *grid*, é encontrar um melhor desempenho através a associação de recursos pertencentes a várias organizações. Para isso esses recursos são reunidos, formando organizações virtuais (conjunto de indivíduos e/ou organizações definidos por regras de compartilhamento), com o objetivo de resolver problemas específicos [39].

Foster, Kesselman, Nick e Tuecke, em 2002, apresentaram a proposta inicial do que seria o *Open Grid Services Architecture* (OGSA) [41], mostrada na figura 2.2, cujas idéias foram estudadas pelo *Grid Group Forum* (GGF), que produziu um documento denominado "*The Open Grid Services Architecture version 1.5*" [42], e que define uma arquitetura comum, padrão e aberta para o desenvolvimento de aplicações baseadas em Grids. O seu objetivo é padronizar todos os serviços que são normalmente encontrados em um sistema de grid, tais como*: job management service,* que permite a submissão de *jobs* ao grid; *resource management service,* que permite descobrir os recursos adequados para a execução de um determinado *job*, *virtual organization management service*, que permite identificar quais nós e usuários pertencem a uma organização virtual*;* e *security service,* com o objetivo de prover segurança na execução de *jobs; etc*., especificando as interfaces padrões para os mesmos [39].

PUC-Rio - Certificação Digital Nº 0711318/CA

A figura 2 deixa clara a separação das funcionalidades de cada camada. Assim na camada de mais alto nível encontram-se as aplicações e serviços que se utilizam dos serviços da plataforma OGSA e da infra-estrutura básica oferecida pelo *Open Grid Service Infrastructure* (OGSI) [43].

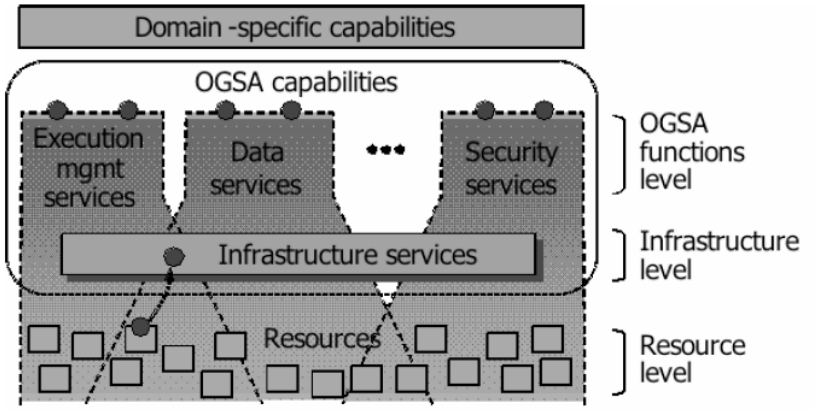

Fig 2: *Open Grid Service Architecture* [42]

Para atender o seu objetivo, a OGSA especifica um conjunto de interfaces padrões para os serviços citados acima, permitindo que eles se comuniquem entre si, e que devam obedecer a determinados requisitos, dentre os quais, os seguintes podem ser apontados como mais relevantes [42]:

a) Interoperabilidade e suporte a ambientes dinâmicos e Heterogêneos

   A interoperabilidade entre recursos diversos, heterogêneos e distribuídos deve ser garantida pelo OGSA, assim como a redução da complexidade de administrar tais recursos;

b) Compartilhamento de Recursos entre Organizações

   O OGSA tem como um dos principais propósitos suportar o compartilhamento e a utilização de recursos entre diversos domínios administrativos sejam eles unidades de trabalho dentro de uma organização ou entre organizações diferentes;

c) Otimização:

   O OGSA deve suportar a utilização de recursos da forma mais eficiente possível, evitando a subutilização dos mesmos;

PUC-Rio - Certificação Digital Nº 0711318/CA

d) Assegurar Qualidade de Serviços (QoS):

Entendido que qualidade de serviço não está restrito somente à disponibilidade de recursos, segurança e desempenho. Expectativas nesse sentido devem ser expressas através de termos mensuráveis;

e) Execução de *Jobs*

O OGSA deve prover funções que permitam o gerenciamento de serviços. Dessa maneira, deve fornecer suporte a funções tais como: *schedule*, controle de *jobs*, alocação de recursos, e manipulação de exceções, ainda que o *job* esteja sendo executado em um ambiente distribuído e heterogêneo; e

f) Segurança

O OGSA tem que prover autenticação e autorização, detecção de intrusão, isolamento, *secure logging*, etc.

Assim como em um *cluster*, a base de aglutinação dos recursos computacionais de um *grid* é o *middleware*. Foster e Douglis [46] definiram um *grid* como sendo uma infra-estrutura de *middleware*, ferramentas e aplicações concernentes com a integração de recursos computacionais geograficamente distribuídos. Logicamente, em se tratando de recursos descentralizados e heterogêneos, o *middleware* capaz de aglutinar esses componentes é, sem dúvida, muito mais complexo do que aquele utilizado em um *cluster* [7].

Um *grid middleware* consiste, basicamente, em uma camada de *software* capaz de encapsular serviços e tarefas comuns em uma interface padrão, escondendo as diferenças existentes de *software* e *hardware*, enquanto mantém as suas próprias características. Ele é capaz de implementar serviços fundamentais, tais como: serviços de informação, descoberta e monitoramento de recursos, submissão e gerência de *jobs*, segurança, gerência de dados e de recursos, etc. [44].

O Globus Toolkit, hoje em sua versão 4, é um *middleware* bastante conhecido. Não se pode dizer que ele seja exatamente uma implementação da OGSA, mas inclui, de fato, a maioria dos requisitos ali definidos, como:

PUC-Rio - Certificação Digital Nº 0711318/CA

monitoração e descoberta de recursos, infra-estrutura de submissão de serviços, infra-estrutura de segurança, serviços de gerenciamento de dados, etc. [39].

Tal como acontece em um *cluster,* tarefas em paralelo podem ser executadas em um *grid* implementado com uma arquitetura *Single Process Multiple Data*, ou SPMD. Assim, as consultas para uma aplicação, digamos o BLAST, podem ser divididas em sub-consultas, que são então processadas simultaneamente em diferentes nós, de forma a obter os resultados mais rapidamente [45].

Conclusão

O presente texto teve como objetivo mostrar alguns conceitos básicos sobre a execução de aplicações em ambientes de cluster e de grid. Várias ferramentas estão sendo atualmente desenvolvidos para ambientes de *cluster* e de *grid* e que tem apresentado bons resultados, como, por exemplo, o mpiBLAST [18] e o balaBLAST [19]. Ambas as ferramentas procuram aumentar a eficiência de execução do BLAST implementando o paralelismo de sua execução a partir da fragmentação de uma base de dados biológica e da distribuição desses fragmentos através dos nós de um *cluster*, ou de um *grid*, nos quais o BLAST é executado.

PUC-Rio - Certificação Digital Nº 0711318/CA

Apêndice 3

Roteiro para Avaliação e Comparação de Desempenho de Ferramentas BLAST em Paralelo

O presente texto tem por objetivo propor um conjunto de ações, sob a forma de um roteiro, a ser seguido para efetuar a avaliação e comparação do desempenho de ferramentas BLAST em um ambiente distribuído. O roteiro aqui proposto é calcado nos fatores e critérios de avaliação de ferramentas de software apresentados no Capítulo 2, e inclui algumas das idéias citadas no Capítulo 3.

PUC-Rio - Certificação Digital Nº 0711318/CA

Insumos

Incluídas no escopo do presente trabalho, foram efetuadas duas avaliações considerando a ferramentas balaBLAST. A primeira, constante do Capítulo 4, teve por objetivo complementar os testes para verificar se o balaBLAST efetuava de fato o balanceamento de carga durante a sua execução. A segunda, incluída no Capítulo 5, teve por alvo avaliar o desempenho do balaBLAST em relação a outras ferramentas BLAST, voltadas ao ambiente distribuído.

Na primeira avaliação, a estratégia adotada incluíu a utilização de máquinas heterogêneas em ambientes de *cluster* e de *grid*, e também o particionamento das bases de dados biológicas, de forma normal, balanceada e randômica. A utilização dos três tipos de particionamento pode ser explicada pelo fato de que possibilitam considerar a distribuição do serviço entre equipamentos de diferentes capacidades de processamento, além de ponderar o efeito do que poderia ser um pré-balanceamento de carga da base de dados a partir da utilização de algum critério na ordenação das seqüências ali existentes. Assim, uma vez estabelecida uma métrica para efetuar a avaliação, no caso o Número de Tarefas computadas por

cada equipamento do *cluster* ou do *grid*, foram então efetuados alguns testes de modo a verificar se o balanceamento estava sendo executado.

Na segunda avaliação o alvo foi a análise de desempenho da ferramenta balaBLAST frente a outras ferramentas BLAST em paralelo. Considerando fatores e critérios especificados no capítulo 2, foram selecionados para tal o mpiBLAST, e o gradeBLAST. Para os testes foram utilizadas a base de dados não redundante de proteínas NR e a base Swissprot. A base NR foi escolhida em virtude de oferecer um nível adequado de verificação em virtude do seu tamanho, aproximadamente 4 GB, e da sua complexidade em termos de comparação e alinhamento em relação à uma base de dados de nucleotídeos, como por exemplo, a NT. A base Swissprot serviu à verificação do desempenho das ferramentas em análise para bases de dados menores.

A métrica básica de confrontação utilizada para efetuar a análise de desempenho foi o Tempo de Execução, da mesma forma que nos trabalhos relacionados nos Capítulo 3, medido em situações diversas em um ambiente de *cluster* e de *grid*.

No entanto, apesar de considerada a métrica acima citada como básica para efetuar a medição do desempenho das ferramentas, ponderou-se outras métricas, também, necessárias à análise, assim como a adoção de alguns critérios de avaliação e procedimentos. Um exemplo disso foi a utilização do Número de Leituras e Gravações em Disco, efetuados durante o tempo de execução de uma ferramenta, para examinar o seu comportamento em determinadas situações de execução.

Na realidade, existem alguns fatores que podem ser considerados para avaliar se uma ferramenta BLAST é melhor do que alguma outra, de acordo com a motivação que se tenha em fazer tal avaliação. Por exemplo, se a motivação é efetuar um benchmark de ferramenta BLAST em ambiente distribuído, como foi o caso mostrado no Capítulo 5, então se pode pensar em alguns fatores a considerar, como por exemplo:

PUC-Rio - Certificação Digital Nº 0711318/CA

a) Multiplicidade de execuções:

Pode-se considerar que diferentes condições possam ocorrer em diferentes execuções de uma ferramenta. Logo, é desejável que sejam efetuadas, pelo menos, duas execuções em iguais condições, em uma avaliação. O melhor resultado pode ser considerado para a análise de resultado;

b) Igualdade de condições de execução:

Para efetuar uma avaliação entre diversas ferramentas é necessário que todas as execuções efetuadas, em uma determinada situação, tenham exatamente as mesmas condições em termos de hardware e software utilizados,

c) Adequação ao ambiente de execução:

O ambiente em que a ferramenta é capaz de ser executada é um fator primordial na avaliação de uma ferramenta de software, em particular das ferramentas BLAST em paralelo. Devem-se considerar, em uma avaliação, apenas aquelas que sejam adequadas a uma instalação.

d) Latência:

É fato que em ambientes de *grid* existe um período relativamente grande nas comunicações entre os diversos nós remotos. Diferentes meios de comunicação determinam diferentes tempos de transmissão, e que acarretam pequenos atrasos, mas que no conjunto, determinam um período de execução diferenciado, como aquele que foi observado no Capítulo 5.

e) Motivação.

A visão que um simples usuário possui quanto à avaliação de desempenho de uma determinada ferramenta é, sem dúvida, diferente daquela de um desenvolvedor. Normalmente, este último irá procurar identificar na estratégia utilizada pelo programa de melhor desempenho,

PUC-Rio - Certificação Digital Nº 0711318/CA

alguma maneira de acelerar a sua própria ferramenta, ou de outrem, a partir do uso, por exemplo, de estratégias desenvolvidas para SGBDs.

Roteiro Proposto

O roteiro aqui proposto é composto por um conjunto de passos, ou ações, que tem por objetivo orientar o processo de avaliação e comparação de ferramentas BLAST em ambientes distribuídos, e que são descritos a seguir:

a) Verificar que ferramentas BLAST em paralelo são compatíveis com a instalação existente.

   Nem todas as ferramentas conseguem ser executadas em todos os ambientes. Deve-se verificar, portanto, se a ferramenta BLAST é compatível com o Hardware e o Software existentes na instalação. Por exemplo, o W.nd Blast não poderá ser executado em uma instalação que utilize somente o Linux como Sistema Operacional.

b) Caso algumas das ferramentas compatíveis possuírem algum custo de aquisição, manutenção ou instalação deve-se verificar a adequabilidade em se obter essas ferramentas.

   A grande maioria das ferramentas BLAST existentes, apesar de possuírem licença GNU, apresentam um desempenho bastante satisfatório, e, portanto, somente casos excepcionais podem justificar a aquisição de algum software pago. Por outro lado, se a instalação e/ou manutenção de algumas das ferramentas exigirem a contratação de pessoal especializado, ou ainda a compra de novos equipamentos, então os custos de aquisição devem ser analisados quanto à sua viabilidade.

c) Efetuar uma avaliação preliminar das ferramentas, após a seleção inicial, contemplando alguns dos fatores e critérios citados no Capítulo 2.

   Fatores tais como abrangência, acurácia, funcionalidade, facilidade, entre outros, citados nos Capítulo 2, devem ser previamente avaliados.

PUC-Rio - Certificação Digital Nº 0711318/CA

Por exemplo, não adianta uma ferramenta ter um desempenho excelente, superior às demais ferramentas analisadas, caso os resultados obtidos não estejam corretos, ou então, se a cada vez que a ferramenta é executada, ela forneça um resultado diferente.

d) Selecionar a métrica, ou métricas, adequadas ao objetivo a alcançar. Em caso de desempenho, então a métrica a ser adotada poderá ser o Tempo de Execução.

   A escolha de uma métrica adequada é fundamental quando se deseja efetuar a avaliação de uma ferramenta, ou ainda uma comparação entre diversas ferramentas. Ao final dos testes de uma avaliação, as medições obtidas, utilizando aquela métrica escolhida, permitirão a análise dos resultados obtidos segundo o objetivo a ser alcançado.

   Pode acontecer que a adoção de uma única métrica não seja suficiente para se efetuar a análise dos resultados, e nesse caso devem-se escolher outras métricas, No Capítulo 5, durante a avaliação de desempenho entre ferramentas BLAST, foi necessária a utilização da quantidade de E/S para tornar possível a análise dos resultados.

e) Elaborar um programa de testes, que possua uma abrangência suficiente para cobrir todos os possíveis casos de execução na instalação.

   Em outras palavras, é importante que todos os casos possíveis de execução sejam testados, de modo a viabilizar uma análise mais completa, consciente e segura. È importante que cada um dos testes executados seja repetido, principalmente em se tratando de ambientes não controlados, como é o caso de *grids*, e nesse caso, apenas o melhor resultado deve ser aproveitado.

f) Se possível, solicitar ao suporte da instalação o acesso exclusivo aos equipamentos para execução dos testes. Em caso de *cluster*, normalmente é mais fácil obter tal acesso por tratar-se de um ambiente controlado, o que não é verdade para um *grid*.

PUC-Rio - Certificação Digital Nº 0711318/CA

O objetivo é obter um ambiente controlado para efetuar a avaliação. Normalmente, outros programas podem estar sendo executados no momento do teste, e esse fato pode ocasionar resultados distorcidos, e, portanto duvidosos, o que inviabiliza a correta avaliação dos resultados obtidos.

g) É importante, em cada situação, que a execução seja feita utilizando-se os mesmos equipamentos. Caso haja alguma impossibilidade, procurar, pelo menos utilizar equipamentos de mesmo porte em termos de hardware e software.

   A utilização de diferentes equipamentos para a realização de testes em uma avaliação de desempenho entre diferentes ferramentas pode promover resultados inconsistentes, mesmo quando possuem o mesmo poder computacional e mesmo sistema operacional. No entanto, se acontecer algum tipo de impossibilidade e a repetição dos testes utilizando outro equipamento representarem um custo muito alto, então uma opção é verificar a existência de algum outro pelo menos de mesmo porte. Caso não existam tais equipamentos então a avaliação de desempenho entre as ferramentas deverá ser reiniciada.

h) Analisar os resultados das execuções, e elaborar gráficos que evidenciem o desempenho das ferramentas analisadas em cada situação. Eles serão bastante úteis em demonstrar a capacidade de cada uma delas, principalmente quando for necessário apresentar os resultados para ratificação por terceiros.

   Normalmente a utilização de gráficos é bem mais elucidativa que um relatório, permitindo um bom entendimento dos resultados. Programas como o Excel facilitam a elaboração de gráficos a partir da importação de dados constantes em relatórios, normalmente em formato texto, produzidos durante os testes. A escolha dos dados a serem incluídos nos gráficos também é fundamental, e devem produzir um significado real em relação aos objetivos que se pretende alcançar. No caso de um benchmark, habitualmente, os tempos de execução das ferramentas em

PUC-Rio - Certificação Digital Nº 0711318/CA

cada execução efetuada, em diferentes condições, deve constar dos gráficos.

Conclusão

No presente anexo foi apresentado uma proposta de roteiro contendo um conjunto de ações destinadas a auxiliar no processo de avaliação de desempenho de ferramentas BLAST em paralelo, O roteiro proposto foi fundamentado em fatores e critérios estabelecidos no Capítulo 2 do presente trabalho, e serviu de base à avaliação apresentada no Capítulo 5.

PUC-Rio - Certificação Digital Nº 0711318/CA

Apêndice 4

Testes para Avaliação do Balanceamento de Carga

BASE	Frag.	Maq.	Seq.	Bases/ Aminoac.	Tipo 1	Tipo 2	Formatação
Ecoli	8	8	1000	500	4	4	Normal
Ecoli	8	8	1000	500	4	4	Balanceada
Ecoli	8	8	1000	500	4	4	Randômica
Ecoli	8	8	1000	1500	4	4	Normal
Ecoli	8	8	1000	1500	4	4	Balanceada
Ecoli	8	8	1000	1500	4	4	Randômica
Ecoli	8	8	5000	1500	4	4	Normal
Ecoli	8	8	5000	1500	4	4	Balanceada
Ecoli	8	8	5000	1500	4	4	Randômica
Swissprot	8	8	500	1500	4	4	Normal
Swissprot	8	8	500	1500	4	4	Balanceada
Swissprot	8	8	500	1500	4	4	Randômica
Swissprot	8	8	1000	1500	4	4	Normal
Swissprot	8	8	1000	1500	4	4	Balanceada
Swissprot	8	8	1000	1500	4	4	Randômica
Swissprot	8	8	5000	1500	4	4	Normal
Swissprot	8	8	5000	1500	4	4	Balanceada
Swissprot	8	8	5000	1500	4	4	Randômica
Swissprot	16	16	5000	1500	8	8	Normal
Swissprot	16	16	5000	1500	8	8	Balanceada
Swissprot	16	16	5000	1500	8	8	Randômica
NR	12	12	500	500	4	8	Normal
NR	12	12	500	500	4	8	Balanceada
NR	12	12	500	500	4	8	Randômica
NR	12	12	500	1000	4	8	Normal
NR	12	12	500	1000	4	8	Balanceada
NR	12	12	500	1000	4	8	Randômica
NR	12	12	1000	500	4	8	Normal

PUC-Rio - Certificação Digital Nº 0711318/CA

NR	12	12	1000	500	4	8	Balanceada
NR	12	12	1000	500	4	8	Randômica
NR	12	12	1000	1000	4	8	Normal
NR	12	12	1000	1000	4	8	Balanceada
NR	12	12	1000	1000	4	8	Randômica
NT	16	16	500	500	5	11	Normal
NT	16	16	500	500	5	11	Balanceada
NT	16	16	500	500	5	11	Randômica

PUC-Rio - Certificação Digital Nº 0711318/CA

Apêndice 5

Testes para Avaliação de Desempenho

BASE	Grid.	Ferram.	Maq.	Frag.	Seq.	Bases / Aminoac.	PUC	UFF
NR	**Nao**	**balaBLAST**	**4**	**4**	1000	1000	5	0
NR	**Não**	**balaBLAST**	**4**	**4**	500	500	5	0
NR	**Nao**	**balaBLAST**	**4**	**8**	500	500	5	0
NR	**Nao**	**balaBLAST**	**8**	**8**	1000	1000	5	0
NR	**Não**	**balaBLAST**	**8**	**8**	500	500	5	0
NR	**Nao**	**gradeBLAST**	**4**	**4**	1000	1000	5	0
NR	**Não**	**gradeBLAST**	**4**	**4**	500	500	5	0
NR	**Não**	**gradeBLAST**	**8**	**8**	1000	1000	9	0
NR	**Não**	**gradeBLAST**	**8**	**8**	500	500	9	0
NR	**Não**	**mpiBLAST**	**4**	**4**	1000	1000	6	0
NR	**Não**	**mpiBLAST**	**4**	**4**	500	500	6	0
NR	**Não**	**mpiBLAST**	**4**	**8**	500	500	6	0
NR	**Não**	**mpiBLAST**	**8**	**8**	1000	1000	10	0
NR	**Não**	**mpiBLAST I**	**8**	**8**	500	500	10	0
SP	**Não**	**balaBLAST**	**4**	**4**	1000	1000	5	0
SP	**Não**	**balaBLAST**	**4**	**4**	500	500	5	0
SP	**Não**	**gradeBLAST**	**4**	**4**	1000	1000	5	0
SP	**Não**	**gradeBLAST**	**4**	**4**	500	500	5	0
SP	**Não**	**mpiBLAST**	**4**	**4**	1000	1000	6	0
SP	**Não**	**mpiBLAST**	**4**	**4**	500	500	6	0
NR	**Sim**	**balaBLAST**	**4**	**4**	1000	1000	1	4
NR	**Sim**	**balaBLAST**	**4**	**4**	500	500	1	4
NR	**Sim**	**gradeBLAST**	**4**	**4**	1000	1000	1	4
NR	**Sim**	**gradeBLAST**	**4**	**4**	500	500	1	4
NR	**Sim**	**mpiBLAST**	**4**	**4**	1000	1000	2	4
NR	**Sim**	**mpiBLAST**	**4**	**4**	500	500	2	4
NR	**Sim**	**balaBLAST**	**8**	**8**	1000	1000	1	8
NR	**Sim**	**balaBLAST**	**8**	**8**	500	500	1	8
NR	**Sim**	**gradeBLAST**	**8**	**8**	1000	1000	1	8
NR	**Sim**	**gradeBLAST**	**8**	**8**	500	500	1	8

PUC-Rio - Certificação Digital Nº 0711318/CA

NR	**Sim**	**mpiBLAST**	**8**	**8**	1000	1000	2	8
NR	**Sim**	**mpiBLAST**	**8**	**8**	500	500	2	8
NR	**Nao**	**NCBI BLAST**	**1**	**1**	500	500	1	0

PUC-Rio - Certificação Digital Nº 0711318/CA