A GT3 based BLAST grid service for biomedical research

Micha Bayer[1], Aileen Campbell[2] & Davy Virdee[2]

[1]National e-Science Centre, e-Science Hub, Kelvin Building, University of Glasgow, Glasgow G12 8QQ

[2]Edikt, National e-Science Centre, e-Science Institute, 15 South College Street, Edinburgh EH8 9AA

Abstract

BLAST is one of the best known sequence comparison programs available in bioinformatics. It is used to compare query sequences to a set of target sequences, with the intention of finding similar sequences in the target set. Here, we present an implementation of BLAST which is delivered using a grid service constructed with the Globus toolkit v.3. This work has been carried out in the context of the BRIDGES project, a UK e-Science project aimed at providing a grid based environment for biomedical research. To achieve maximum performance in a grid context, the service uses parallelism at two levels. Single query sequences can be parallelised by using mpiBLAST, a BLAST version that fragments the target database into multiple segments which the query sequence can be compared against in parallel. Multiple query sequences are partitioned into sub-jobs on the basis of the number of idle compute nodes available and then processed on these in batches. To achieve this, we have provided our own java based scheduler which distributes sub-jobs across an array of resources.

1 The BRIDGES Project

The BRIDGES project (Biomedical Research Informatics Delivered by Grid Enabled Services [1]) is a core e-Science project aimed at developing grid-enabled bioinformatics tools to support biomedical research. Its primary source of use cases is the Cardiovascular Functional Genomics Project (CFG), a large collaborative study into the genetics of hypertension (high blood pressure). Hypertension is partly genetically determined, and investigation of the genes and biological mechanisms responsible for high blood pressure is of great importance.

BRIDGES aims to aid and accelerate such research by applying grid-based technology. This includes data integration tools (see reference [2], this volume) but also grid support for compute intensive bioinformatics applications.

2 BLAST

The comparison of biological sequences (DNA or protein) is an important part of any functional genomics research project - the initial research is generally carried out using an animal model, such as mouse or rat, and results from these species then are applied to humans. This generally involves sequence similarity searches, often on a very large scale, with query sequences being compared to several million target sequences.

BLAST - the Basic Local Alignment Search Tool [3] is a widely used search algorithm that is used to compute alignments of nucleic acid or protein sequences with the goal of finding the *n* closest matches in a target data set. BLAST takes a heuristic (rule-of-thumb) approach to a computationally highly intensive problem and is one of the fastest sequence comparison algorithms available, yet it still requires significant computational resources. It does therefore benefit from being run in a grid computing context.

There are a number of public sites [4, 5] that provide users with web based access to BLAST, and these generally use dedicated clusters to provide the service. However, the growth of sequence data over the last decade has been exponential [21], and consequently searches take increasingly longer. Given that most biologists use these public sites rather than running the computation on their own machines (BLAST is freely available as a command line tool), the load on the purpose built clusters has increased dramatically and now significant queuing times are becoming increasingly common. A typical use of BLAST will usually involve searching against the *nt* database from NCBI - a data set that contains several of the main sequence repositories and is currently approx. 7 gb in size (over 2,220,000 target sequences).

The current work therefore serves two main purposes:

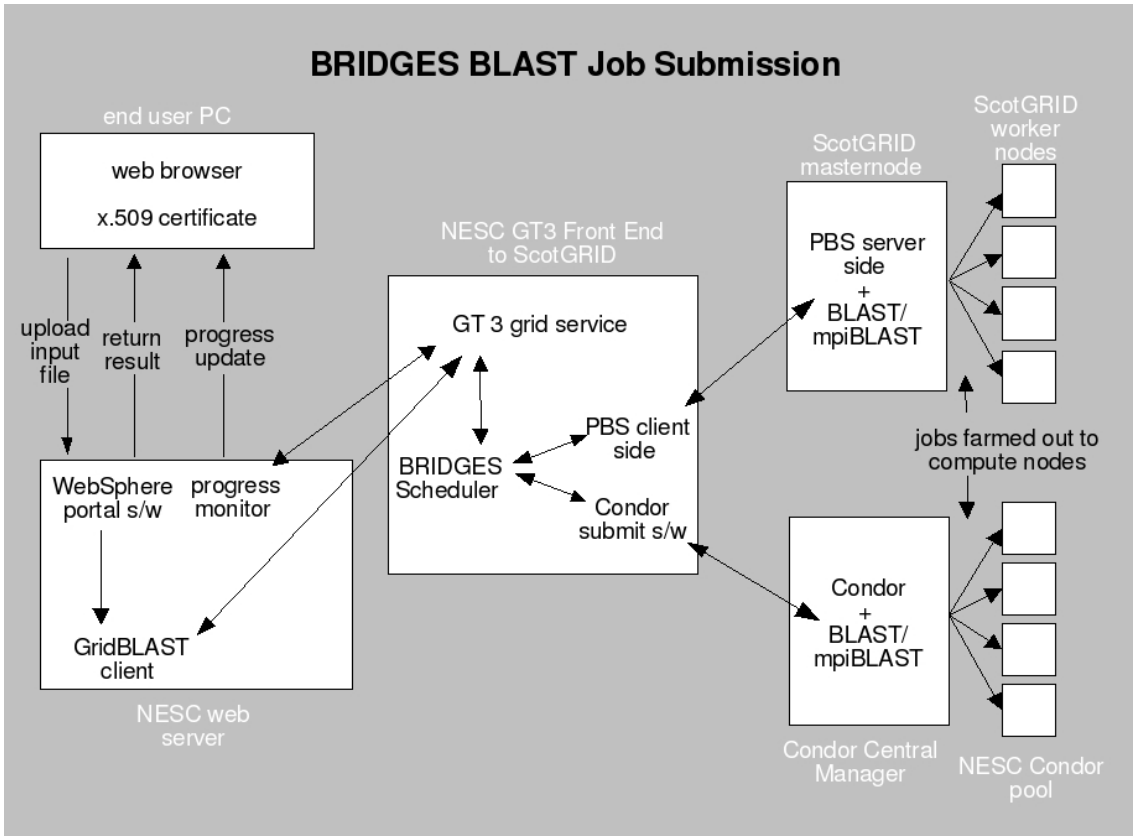

Figure 1: Schematic of system architecture

1. to provide CFG members with access to a BLAST service that – ideally – provides a faster response time than the publicly available services (e.g. NCBI, EBI)
2. to test the usefulness of GT3 as a front end for bioinformatics applications

3 Parallelising BLAST

In order to benefit from being deployed in a grid environment an application has to be parallelised, i.e. single jobs must be partitioned into independent sub-jobs that can be run on remote resources separately. In the case of BLAST (and possibly similar sequence comparison programs) parallelisation can be achieved at three different levels [6, 7]:

1. A single query can be compared against a single target sequence using several threads in parallel, since there are $O(n \times m)$ possible alignments for a query sequence of length n and a target sequence of length m.
2. Input files with multiple query sequences can be parsed to provide individual query sequences or blocks of sequences, and these can then all be compared against multiple identical instances of the target data file in parallel.
3. With input files containing only a single query sequence, the target data can be segmented into n copies for n available compute nodes, and then multiple identical instances of the query sequence can each be compared to a different piece of the target data in parallel.

There are several existing implementations which take approach 2 [7, 8] or 3 [9, 10, 11], but to the best of our knowledge there are none that do both. In order to provide maximum speed and flexibility we decided to implement both approaches. For the purpose of this project mpiBLAST was chosen as an implementation of approach 3 because it is the most comprehensive package, freely available and contains its own data formatting program. We provided our own implementation of query file segmentation in our implementation of BLAST (see section 4.2).

4 BRIDGES GT3 BLAST service

There is a growing number of grid based implementations of BLAST [12, 13, 14, 15], based on various grid middleware but we here

present the first implementation based on version 3 of the Globus toolkit. Globus v.3 is based on the web services programming model but services have been adapted to operate in a grid computing context, with service data, client notification mechanisms and statefulness as added functionality. Our GT3 based implementation of BLAST is used in conjunction with our own metascheduler that allows jobs to be farmed out to remote clusters and the results to be recombined.

4.1 Basic design

A GT3 core based grid service is used as a thin middleware layer to our application (Figure 1). The service itself has deliberately been kept very basic to allow easy porting to other platforms since grid computing is still very much in flux. It therefore only uses limited GT3 functionality, with no client notification or service data having been implemented. The only significant GT3-specific functionality used is the service factory pattern. This is a useful feature because it provides an easy way of handling concurrency in the context of this application, with a service instance representing a single job submitted by a single user. The actual BLAST computation is carried out by NCBI BLAST which is freely available from NCBI [5]. The software components contributed by the authors have been implemented in Java.

4.2 Scheduler and input data segmentation

For the purpose of this project we decided to provide our own scheduler since at the time of designing the application established grid meta-schedulers were rare and none were available that satisfied our requirements with respect to the particular combination of back end OS and batch submission system on our resources (*n.b.* "resource" here denotes a computational backend such as a compute cluster).

Our scheduler can take an unlimited number of resources as an argument and uses the following algorithm to distribute subjobs across these:

```
parse input and count no. of query sequences
poll resources and establish total no. of idle
      nodes = no. of sub-jobs to be run
calculate no. of sequences to be run per sub-job
      n (= no. of idle nodes/no. of sequences)
while there are sequences left
      save n sequences to a sub-job input file
if the number of idle nodes is 0
      make up small number of sub-jobs
            (currently hardcoded to 5) and evenly
            distribute these into queues across
            resources
else assuming the number of idle nodes is i,
      send i subjobs to the resource as separate
      threads
when results are complete save to file in the
      original input file order
return this to the user
```

Thus, the system will always make use of the maximum number of idle nodes across resources if several query sequences exist. Load balancing is achieved by assigning only as many sub-jobs to a resource as there are idle nodes.

4.3 Database segmentation

At the time of writing this segment of functionality had not been implemented yet. The design for this is to use mpiBLAST [10, 16] for single query sequences which cannot be parallelised using the above scheduling algorithm. mpiBLAST splits the target database into segments which means that a single query sequence can be compared against potentially all fragments of the database in parallel, with each comparison running as a sub-job on a single node. Benchmarking by the mpiBLAST developers has shown that this results in at least linear and sometimes even superlinear speedup [10].

4.4 Deployment and configuration

The grid service is deployed using Apache Tomcat running on Linux server with GT3. To allow easy modification of the set of compute resources available to the service, resource details are held in an XML configuration file which is read by the service at deploy time and an array of resources is initialised accordingly. The details of interest include the type of batch submission system, the domain name of the resource's head node, the number of compute nodes, memory etc.

4.5 Back end compute resources in this implementation

The main compute resource is the ScotGRID compute cluster [17] at Glasgow University, a 250 cpu Linux cluster which was originally commissioned for the particle physics community but now includes a 20% bioinformatics share. It uses OpenPBS [18] as its batch submission system. The second resource employed in our implementation is a local Condor [19] pool of 25 laptop PCs. This

may be extended to include other machines in the future.

4.6 Front end and user access

A web portal has been designed for the BRIDGES project to allow the CFG users easy web based access to resources and in order to avoid having to install grid software on end user machines. This has been implemented using IBM Websphere, currently one of the more sophisticated portal packages. Users use a secure login and are then presented with a basic GUI based submission and result handling interface to the service.

4.7 Target databases

Grid based file transfer is potentially time-consuming and the favourable option is to keep frequently used data cached locally, close to the computation. In the case of BLAST this is readily achievable since most users blast against a small number of publicly available target databases (available as flat text files for ftp download, e.g. from NCBI [5] or EBI [4]). These can then be stored on the local NFS of the compute resource and updated regularly.

At the time of writing the databases available to the service were the *nt* and *nr* databases (the most commonly used standard nucleotide and protein databases) as well whole chromosome databases for human, mouse and rat. Typical sizes of these databases are several gigabytes, with the *nt* and *nr* databases growing exponentially.

5 GT3, porting issues and future work

The initial design of the current system included GT3 GRAM functionality, i.e. the wrappers to OpenPBS and Condor, as well as job handling facilities. However, difficulties in using GT3 GRAM, especially the lack of useful documentation, forced a redesign and subsequently only the GT3 core was used for the final implementation. We developed our own java based wrappers to OpenPBS and Condor as part of this implementation.

Another consideration in the redesign was the emerging new standard of WSRF [20], which made a port to another package a likely future requirement. Therefore the involvement of GT3 in the current system was deliberately kept minimal and only a small segment of GT3 functionality was used (service factories). High emphasis was placed on reusability of the code, and as a result the entire job submission mechanism and the scheduler itself can very easily be ported to other middleware, including a WSRF-compliant service.

Future extensions could include support for other backend schedulers and other bioinformatics applications which have similar requirements.

6 References

[1] BRIDGES project website: http://www.brc.dcs.gla.ac.uk/projects/bridges/

[2] Sinnott R, Atkinson M, Bayer M, Berry D, Dominiczak A, Ferrier M, Gilbert D, Hanlon N, Houghton D, Hunt E & White D. 2004. Grid Services Supporting the Usage of Secure Federated, Distributed Biomedical Data. Proceedings of the UK e-Science All Hands Meeting, Nottingham, UK, August 2004.

[3] Altschul SF, Gish W, Miller W, Myers EW & Lipman DJ. 1990. Basic Local Alignment Search Tool. J Mol Biol 215: 403-410.

[4] EBI BLAST website: http://www.ebi.ac.uk/blastall/index.html

[5] NCBI BLAST website: http://www.ncbi.nlm.nih.gov/BLAST/

[6] Pedretti KT, Casavant TL, Braun RC, Scheetz TE, Birkett CL, Roberts CA. 1999. Three Complementary Approaches to Parallelization of Local BLAST Service on Workstation Clusters (invited paper). Proceedings of the 5th International Conference on Parallel Computing Technologies, p.271-282, September 06-10.

[7] Braun RC, Pedretti KT, Casavant TL, Scheetz TE, Birkett CL & Roberts CA. 2001. Parallelization of local BLAST service on workstation clusters. Future Generation Computer Systems 17: 745-754.

[8] Clifford, R & Mackey AJ. 2000. Disperse: a simple and efficient approach to parallel database searching. Bioinformatics 16: 564-565.

[9] Mathog DR. 2003. Parallel BLAST on split databases. Bioinformatics 19: 1865-1866.

[10] Darling AE, Carey L & Feng W. 2003 The design, implementation and evaluation of mpiBLAST. Proceedings of ClusterWorld Conference & Expo and the 4th International Conference on Linux Clusters: The HPC Revolution.

[11] Hokamp K, Shields DC, Wolfe KH & Caffrey DR. 2003. Wrapping up BLAST and other applications for use on Unix clusters. Bioinformatics 19: 441-442.

[12] GridBlast at Keck BioCenter, University of Wisconsin: http://bioinf.ncsa.uiuc.edu/

[13] GridBlast at A-Star Bioinformatics Institute Singapore: http://www.bii.astar.edu.sg/infoscience/dcg/gridblast/index.asp

[14] North Carolina BioGrid: http://www.ncbiogrid.org/tech/apps.html

[15] RIKEN GridBlast implementation: http://big.gsc.riken.jp/big/Members/fumikazu/Activity_Item.2004-02-02.0425

[16] mpiBLAST website. http://mpiblast.lanl.gov/

[17] ScotGRID website. http://www.scotgrid.ac.uk/

[18] OpenPBS website. http://www.openpbs.org/

[19] Condor website. http://www.cs.wisc.edu/condor/

[20] Globus WSRF web page: http://www.globus.org/wsrf/default.asp

[21]GenBank statistics web page: http://www.ncbi.nih.gov/Genbank/genbankstats.html