International Journal of Genetic Engineering and Biotechnology.
ISSN 0974-3073 Volume 1, Number 1 (2010), pp. 23--33

http://www.irphouse.com

'Blast those Genes': A Gridified Framework for Bioinformatics Blast using the A^{3p}viGrid

Avinash Shankaranarayanan and Christine Amaldas

School of Chemical and Biotechnology, SASTRA UNIVERSITY,
Tirumalaisamudram, Thanjavur - 613 401. Tamil Nadu, India.
E-mail: avigrid@gmail.com
Business Information Systems, Royal Melbourne Institute of Technology (RMIT),
Hanoi, Vietnam E-mail: christine.amaldas@rmit.edu.vn

Abstract

BLAST is one of the most widely used bio informatics tools; largely used in finding matching gene sequences from genome databases from an input query string (sequence). Dynamic programming and heuristic search mechanisms in BLAST can rapidly identify similar sequences (similarity matching); the performance starts to deteriorate when there is an exponential increase in the size of the data set (input query) which remains to this day an important research problem. As the size of genome databases and input query size increases there is decrease in compute performance. We propose improvements to the BLAST application performance using distributed mini grid architecture called A^{3p}viGrid that significantly improves the applications performance by applying agent coalition algorithms using the power server model of computing. Unlike many Grid based middleware architectures such as the Globus toolkit, the A^{3p}viGrid tries to provide enhanced performance and lower implementation costs through the deployment of commonly used web scripting languages using a power server model architecture. Comparative analysis of our previous implementations shows that there is a significant improvement in the performance of BLAST searches using the A^{3p}viGrid platform. A Fasta formatted sequence database (human DNA sequence from clone RP11-10K8 on chromosome 1) was used to evaluate the BLAST searches which indicates improvements in the search over a simple 10 node mini-Grid system. The future direction of this research would focus on evaluating the scalability of this system.

Keywords: Agents, Blast, Coalition, Mini Grid, Quality of Service and Scalability.

Introduction

Biologists often require sequence comparison and alignment applications such as BLAST [12], ClustalW [8] and repeat finding algorithms [9], which are effectively utilized for processing large sets of gene sequences for similarity matching. These tools were previously extensively investigated [4] and observed. Most of the existing bioinformatics tools were either too low level (complicated); too expensive for most laboratories to afford; and inflexible to customization on heterogeneous networks and computational environments without significant technical expertise. The exponential rise in the size of datasets increases the problems related to the scalability of existing bioinformatics programs and tools. One approach to solving this problem is to break the problems into a number of sub-problems which could be done easily either in the algorithmic level (re-programming for parallel processing such as MPI) or through dataset parallelism where the data is broken down depending on the number of available processors. Bioinformatics applications along with well defined outputs heavily rely on various methods of pattern recognition and statistical methods of information processing (based on sequences). The paper is subdivided into the following sections: Section II will give an overview of current Blast Literature with insights into high performance distributed systems; Section III will highlight and discuss the limitations of current systems and the need for new mechanisms; in Section IV we talk about the A^{3p}viGrid [4] architecture and show how the Blast application can be run as a Grid service; finally in Section V we discuss about the results obtained from running Blast in parallel using our mini-grid test bed and conclude the paper with future discussions and enhancements to our research.

Basic Local Alignment Search Tools

Basic Local Alignment Search Tools (BLAST) are a set of programs used for searching sequence databases with that of the input query sequence for similarity matching. BLAST is a heuristic search method which makes assumptions about the data based on experience. This implies that it is not guaranteed to find the best alignment in all possible circumstances. It sacrifices some accuracy for a greater increase in speed. The BLAST algorithm is an improvement over the Smith-Waterman algorithm [15], which is slow but guaranteed to get the best possible alignments given certain input parameters. BLAST uses a special database format to speed up the search operations. Several pre-packaged databases exists, and the most notable is the “nr” database which is the non-redundant database consisting of all sequences in GenBank. As the size of the dataset increases the time taken for meeting the alignment conditions increases exponentially. Hence there is a need to parallelize the processing steps of the algorithm as follows:

Hardware Acceleration

Hardware accelerators parallelize a single query sequence to a single database entry. As symmetric multiprocessors (SMPs) and symmetric multi threaded systems (SMTs) cannot support the level of parallelism required, custom hardware must be used. Singh et al. introduced the first hardware accelerator for BLAST [13]. More recently, Time

Logic[1] has commercialized a field-programmable gate array (FPGA) based accelerator called the DeCypher BLAST hardware accelerator [2]. FPGAs are reprogrammable hardware devices that support the custom computing needs that are characteristic of data intensive problems. FPGAs can be reprogrammed to utilize powerful sequence alignment algorithms such as the Smith Waterman algorithm [6].

Query Segmentation

Query segmentation provides the most elaborate method of parallelization of BLAST by splitting up a query string so that each node in a cluster or Grid searches a fraction of the sequence database. Multiple BLAST searches can be executed in parallel on different query sets. However, such an approach typically requires the entire database to be replicated on to each of the nodes local storage system [12]. If the database does not fit the memory requirements of the node under utilization, the node suffers from disk I/O access times leading to increase in access time of the database.

Database Segmentation

*"BLAST searches tend to become too slow due to increase in size of databases that run out of core memory" [5].*Database segmentation is an orthogonal approach to query segmentation. Query segmentation utilizes individual query segments to match against an entire database on each individual node whereas database segmentation keeps the queries intact and distributes individual database segments to each of the nodes for query processing. The biggest challenge using this approach would be to ensure that the statistical scoring is properly produced as this depends on the size of the database subset. Database segmentation has been implemented in a closed-source commercial product by TurboWorx, Inc. called TurboBLAST.

High Performance Distributed Computing

Distributed technologies can be classified into a number of technological streams such as, High Performance Cluster / Grid [10, 11], Multi-Agents [14] and Peer-to-Peer (P2P) [3] systems. BLAST users can take advantage of low-cost, efficient Linux cluster architectures such as Beowulf. Unfortunately, the efficiency declines when scaled to hundreds of nodes because of serial result-merging and output domination. To tackle such scalability challenges the message passing interface (MPI) was introduced. mpiBLAST is a parallel implementation of BLAST using the MPI interface. It functions with database-segmenting technique [16] where the number of compute nodes increases from 1 to 4 and mpiBLAST achieves a speed-up of nearly 10. It achieves super linear speed-up by segmenting a BLAST database where each of the nodes in the computational cluster search through a unique segment of the database. Database segmentation permits each node to search a smaller portion of the database, eliminating disk I/O and vastly improving BLAST performance. Database segmentation does not create heavy communication demands with respect to a cluster.

Darling [5] evaluated the performance of BLAST using a 300-KB input query file against a 5.1-GB uncompressed 'nt' database which took 1346 minutes (or 22.4

[1]http://www.timelogic.com

hours) to compute on a single compute node. The same query was run within 8 minutes on 128 nodes on the Green Destiny supercomputing cluster. Green Destiny is a supercomputing infrastructure having negligible turnaround time. When comparing smaller sequences with large genomes the efficiency of mpiBLAST decreases as the number of nodes increases. If we take the speed-up and divide it by the number of nodes available, the efficiency of mpiBLAST across four nodes is 2.31 (9.23/4) and drops all the way down to 1.33 (170.41/128) when run across 128 nodes. The reason for this drop is due to the trade off that exists when segmenting the database into many small fragments. There is significant overhead in searching extra fragments. The ideal database segment will typically be the largest fragment that can fit into the memory and not cause any swapping of the disk. Making fragments smaller than the available memory simply adds overhead [5]. A more detailed performance analysis and evaluation can be found in the green destiny paper [5].

Arun Krishnan in his paper [1], talks about applying BLAST to the Globus Grid platform using Perl scripts called GridBLAST on a mini-grid test bed. When looking at the computational aspects of BLAST, typically a full scale BLAST job across whole genomes is highly computationally intensive due to the size of the databases queried upon. It was observed that embarrassingly parallel applications such as BLAST improve execution times when the data set is broken down and processed individually on individual processors. Currently, GridBLAST seems to be the only standalone version of BLAST that can be run on a Globus enabled Grid platform. It is a high throughput task farming application that distributes independent tasks or queries to remote static nodes available on the Grid. The BLAST problem is embarrassingly parallel in nature and belongs to the SPMD class of parallel applications where sequence comparison takes place using parallel datasets. GridBLAST utilizes the Globus toolkit [10] for managing remote files and remote execution of programs. It contains two Perl scripts; one running on a local node (Initiator) and the other on each of the remote nodes (Static). The Initiator utilizes a Static schedule mechanism to query each of the nodes using two methods namely Proportional Scheduling and Minmax Scheduling. The main drawback of Proportional Scheduling is that it does not account for the dynamic changes in latency, bandwidth and load-balancing while distributing the workload to various nodes. Minmax Scheduling optimizes the variability of resources using parameters of historical data and statistical averaging methodology based on the maximum predicted time spent on each node using a fixed window size.

Limitations of Current Mechanisms

- *Optimal resource allocation* strategies need to be incorporated for job processing due to the highly dynamic nature of resource requirements of a job namely load, CPU and latency.
- When looking at *scalability issues* over a Grid as compared to a cluster, remote computational nodes do not share a file system and pose problems related to fault-tolerance, state-information updates; latencies and overloading of remote nodes.

- *Fault tolerance* is a major issue that needs to be addressed in GridBLAST. The resubmission of jobs with respect to timeout or failure detection is based on the intelligence of the scheduling mechanisms.
- *Static sets of nodes* are assumed as the algorithms do not provide mechanisms for adding dynamic sets of nodes and resources.
- All remote nodes are assumed to be running the local grid daemon (E.g. Globus) which is platform dependent most of the time.
- *Latency and bandwidth issues* are not considered during scheduling and remote execution of tasks.
- Restricted to the *size of the dataset (database)* being distributed across remote nodes limits size of node set.
- *Coarse grained load-balancing* achieved through database segmentation involves heavy overhead due to database fragmentation.
- *Centralized failure* of head node or initiator nodes.
- *No automated mechanisms* for discovering optimal nodes and their respective resources.
- Currently, there is *no standalone implementation of BLAST over a dynamic Grid environment*.

Running Blast on the A^{3p}viGrid Experimental Framework

The term, "Grid" first came into use over half-a-decade ago as a synonym for metacomputing. It has evolved over the years to refer to an infrastructure that combines heterogeneous resources over a network to provide for an efficient problem-solving environment.

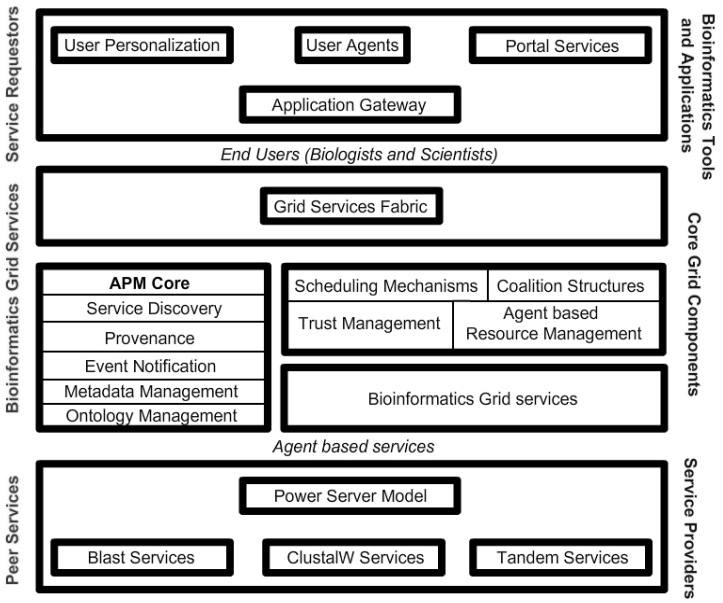

Figure 1: A^{3p}viGrid an Experimental Grid Framework [4]

The ability to invoke a program or workflow say, a java applet or servlet using a web server can be effectively utilized towards distributed processing of data. This is termed as the *"power server model"* of computing whereby a web server and servlets are used for consuming services (similar to the web services architecture). The advantage is the simplicity of the model where the client connects to a bunch of web servers to enable the consumption of remote services using web pages. The default method of connecting remote programs with users transparently is through the use of the Common Gateway Interface (CGI) over the HTTP protocol. This method of computing differs from conventional computing where clients work with a centralized server model of computing. The main goal of power servers is to share CPU power to other users. Every system has a Web server running a servlet as the method of program execution. A client connects to the server and the servlet executes a client by dividing a single computational task into multiple subtasks queued by the scheduler of the system. The application program then invokes a servlet on the server and transfers a small part of the task to the servlet. The servlet then computes the task and sends the results back to the client upon completion. The advantage of the model is that any user can download and install a web server to make his/her machine into a power server. The primary problem of the model is maintenance. When we need to update the software side of the web server we need to manually update each of the web servers. This can be automated using a web server upload page to update when necessary. The maintenance can also be minimized if all the nodes are gathered under one domain and update as required. This way only one copy of the web server and servlet is needed and each of the clients can invoke it on booting thus streamlining the maintenance job. A^{3p}viGrid [4] works on the principle of the power server model of computing. Each of the clients runs the A^{3p}viGrid server which is a simplistic http web server running services in the form of CGI/Perl Scripts. The client side coding model enables the developer to develop services using the common gateway interface (CGI) and can use any of the languages that support CGI scripting. For the sake of simplicity and rapid development of services we have used Perl as the language of choice due to its availability and portability for most platforms. Discussed below are the different components that enable the functioning of the A^{3p}viGrid system.

Components of A^{3p}viGrid System

Peer-to-peer Service Discovery

Any peer (node) needs to transmit some form of state information updates or discovery messages to a known set of nodes. The basic requirements of discovering a remote peer is by identifying the peer by its unique id namely the IP address or domain name and its services offered. The A^{3p}viGrid uses a decentralized directory structure to enable peers to register and de-register peers and their respective services. A good example of this would be Web services definition language (WSDL) which uses UDDI for publishing web services. The A^{3p}viGrid system relies on a similar service namely the Agent based Peer Manager [APM] where each peer hosts a set of service agents running a specific service. Each peer is also given a unique id with respect to the APM registration service which identifies the peer. The unique id ensures that the peer remains unique to that of the APM even if its configuration and

services changes (i.e. changes to IP address and location specific information changes).

The Agent based Peer Manager Service [APM]

The primary aim of the APM is to store location and service information in the form of categorized listing. Similar to P2P file sharing applications such as Bit torrent, which uses the .torrent style structure for seeding files, the APM stores location and services information in the form of servicename_location.APM file.

Example File: GridBlast_abc.apm

APM location: Singapore

Service Name: Grid Blast

Service Description: Takes numeric values as input and executes Blast services in parallel.

Remote results are concatenated back to originator

# Peers	URL	Port (opt)
192.168.1.90	http://192.168.1.90/blastg.pl	80
192.168.1.91	http://192.168.1.91/blastg.pl	80
192.168.1.92	http://192.168.1.92/blastg.pl	8080
192.168.1.93	http://192.168.1.93/blastg.pl	8888
192.168.1.94	http://192.168.1.94/blastg.pl	80
192.168.1.95	http://192.168.1.95/blastg.pl	80
192.168.1.96	http://192.168.1.96/blastg.pl	80
192.168.1.97	http://192.168.1.97/blastg.pl	8888
192.168.1.98	http://192.168.1.98/blastg.pl	9000
192.168.1.99	http://192.168.1.99/blastg.pl	9999

-END

The Initialization Phase

- All peers are capable of being a head node by running the A^{3p}viGrid Server.
- All the peers download a list of peers based on the requirements of the user from the APM.
- Initially an ideal set of peers are initially computed for job processing.
- An optimal coalition is then formulated from the ideal list using the coalition formula and the most appropriate coalition list is finalized.
- Services defined by the user are checked with APM for registration / de-registration.
- Job submission and scheduler are initialized for receiving remote jobs.

Computing Ideal Set of Peers

The most ideal coalitions can be formed with a given set of peers if and only if the latency of the peers is minimal. By applying a latency test, the ideal set of nodes along with their quality of service parameters are processed initially by the client. This is

not a necessity that the client needs to compute the set of nodes. A user can obtain the set as a file which is read by the script as the ideal set. The most important step is to compute the optimized coalition list which determines the closest nodes needed for job processing.

Service Registration and Job Scheduler

The basic operation of any scheduler is to create an ideal job queue for job processing. The job scheduler takes care of job id creation, time of creation etc. It is the duty of job scheduler to log the statuses of the jobs such as job history, success or failure of jobs, compute trust based on job history, de/register services, etc.

Registering a Service

A service is registered by calling the register programs which takes the IP address, hostname, location information, service name and service location/path. It is assumed that each node is represented by one or more agents and a common coalition formation construct is used based on a set of rules. The rules could be different for each agent based on its environment and self-interests. The commonality of service factor plays a vital in the collaboration and formation of coalition in agents.

De-Register a Service

When an agent tends to leave a coalition due to unavailability, resource problems etc, a set of leaving rules are defined based on the subscription rules of the coalition group. Rules like group losses, individual losses, re-negotiation and higher payoff for the stay of the agent, will have to be taken into account for the agent leaving the group.

Running Blast Grid service

A random set of 10 machines where used for job processing. A workflow was built as follows:

- All the nodes ran A^{3p}viGrid web servers
- The Blast_gu.apm file was downloaded by all the peers as part of the initialization phase.
- Each of the nodes computed the ideal set of nodes using a basic ping test based on the Blast grid service list.
- As all the nodes where capable of receiving jobs, one of them were randomly chosen for job execution (Originator).
- A Fasta formatted Sequence database (human DNA sequence from clone RP11-10K8 on chromosome 1) was used to evaluate the Blast searches.
- The input query file is obtained, and the next step is to prepare a set of jobs for job processing using the optimal coalition list.
- Based on QoS Characteristics namely Latency, Load and CPU time, the Originator of the job computes the most optimal coalition.
- Once the coalition list is computed the data files are migrated using the POST method to all the members of the coalition.
- Each of the coalition members start to search using the input query files and

output the results.

- The output of the Search Phase is appended to a file using POST back to originator where the results are formatted using the Blast format perl script and stored as a file or displayed in the browser of the originator.

Results and Conclusion

Ten user agents were selected in random to run the experiments in parallel. To cater to a heterogeneous environment and make it truly a peer-to-peer model of computing, all nodes where connect by DSL or Cisco routers using wireless hotspots and Cable modem lines. We assume N data distributed over tasks **d = log P**, with N an integer multiple of .The computation costs comprise of the initial sort and the comparisons performed during the communication phase. The former involves a total of comparisons $\mathbf{P = 2^d}$, while the latter requires at most **(Nd (d + 1) / 2)** comparisons.

Tables 2 and 3: Turn around times recorded for the experiments performed.

	Processor 0	Processor 1	Processor 2	Processor 3
Seq Blastcode	43.0763	65.8885	48.526	83.8911
Grid Blastcode	58.633	58.6393	58.6436	58.6373
	Processor 5	Processor 6	Processor 7	Processor 8
Seq Blastcode	46.8615	41.2549	49.1826	41.5324
Grid Blastcode	58.6441	58.6389	58.6326	58.6406

	Processor 0	Processor 1	Processor 2	Processor 3
Seq Blastcode	52.575	53.1896	86.0856	118.026
Grid Blastcode	72.4102	72.4286	72.5143	72.4188
	Processor 4	Processor 5	Processor 6	Processor 7
Seq Blastcode	51.9337	52.769	51.5515	51.2998
Grid Blastcode	72.4297	72.4361	72.4246	72.4203

Because the algorithm is perfectly balanced, we assume that idle time is negligible. Our results were obtained by running Gridblast code on Linux Clusters (Fedora Core 3) with 1.8 GHz CPU's and 1GB RAM. While a similar heterogeneous set of peers (6 nodes running Linux Fedora core 3 and 4 running Windows) having different configurations were used for running the algorithm as a Grid service using the A^{3p}viGrid framework. In this project, human DNA sequence (GenBankID: AL611946) has been used as the database. The size of this sequence is 44,921base pairs (bp). The time of execution was taken as the average value of the two experiments with the same settings and parameters in place as observed in the tables 2 and 3.

References

[1] A. Krishnan, "GridBLAST: a Globus-based high-throughput implementation of BLAST in a Grid computing framework", Concurrency and Computational Practice: Practice and Concurrency Computation Practice and Experience, 2005, 17:1607 1623.

[2] A. Shpuntof, and C. Hoover, Personal Communication, August, 2002.

[3] A. Shankar, F. Dehne, A. P. Shankar, and G. Subramanian, "Applying Coalition Concepts to Service Oriented Multi-Agent Load Balancing Systems - A^{3p}viLoad", PDPTA 2005, Monte Carlo Resort, Las Vegas, Nevada, U.S.A., June 2005, pp. 27-30.

[4] A. Shankaranarayanan, F. Dehne, and A. Lewis, "Applying Agent/Grid Coalition Concepts to Service Oriented Multi-Agent Systems - A3pviGrid", IEEE Computer Society, CIMCA-IAWTIC'06, Volume 2, 28 - 30 November, Vienna - Austria, 2006, pp. 315-320.

[5] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST", ClusterWorld Conference & Expo, 2003.

[6] Burt, "Written Statement of Dr. Stanley Burt for the Senate Committee on Commerce, Science, and transportation", Subcommittee on Technology, Innovation, and Competitiveness, 2006.

[7] C. Gibas, and P. Jambeck, "Developing Bioinformatics Computer Skills", O'Reilly & Associates, Inc., U.S.A, 2001.

[8] D. G. Higgins, J. D Thompson, and T. J Gibson, "Using CLUSTAL for multiple sequence alignments.", Methods Enzymol, 1996, 266, pp. 383-402.

[9] G. Benson and L. Dong, "Reconstructing the Duplication History of a Tandem Repeat", Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB-99), 1999, pp. 44-53.

[10] I. Foster, C. Kesselman, "Globus: A metacomputing infrastructure toolkit", The International Journal of Supercomputer Applications and High Performance Computing 1997, 11(2):115-128.

[11] I. Foster, C. Kesselman, S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". International Journal of Supercomputer Applications, 2001

[12] R. Braun, K. Pedretti, T. Casavant, T. Scheetz, C. Birkett,and C. Roberts, "Parallelization of Local BLAST Service on Workstation Clusters", Future Generation Computer Systems, 2001, 17(6):745-754.

[13] R. K. Singh, W. D. Dettloff, V. L. Chi, D. L. Hoffman, S. G. Tell, C. T. White, S. F. Altschul, B. W. Erickson, BioSCAN: A dynamically reconfigurable systolic array for bio-sequence analysis. http://citeseer.ist.psu.edu/163511.html

[14] S. Franklin, and A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Institute for Intelligent Systems, University of Memphis, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996

[15] T. F. Smith, and M. S. Waterman, "Identification of common molecular subsequences" Journal of Molecular Biology, 1981, 147: 195 - 197.

[16] W. Feng, "Green Destiny + mpiBLAST = Bioinfomagic", 10th International Conference on Parallel Computing, 2002.

Biography

Avinash Shankaranarayanan has been actively involved in publishing several research papers and periodicals in International Conferences and Journals of high standards such as the IEEE and ACM. He has presented and presided as Program Committee Member and Program / Conference Chair over a number of International Conferences and has been invited to give guest lectures in Singapore, India, Australia, USA, Japan, Mauritius Europe, the United Kingdom and many parts of South Asia. His research areas include High performance Grid Computing, Bioinformatics, Material Flow Management, and Renewable Energy Systems.

Christine Amaldas has been an active researcher and academic at the Ritsumeikan Asia Pacific University since 2003. She has been the author of several Conferences and Journals spanning the Asia Pacific region. She has presented and presided over a number of conferences and has given guest lectures in Singapore, India, Japan, USA, Mauritius and Australia. She specializes in Asia Pacific Studies, IT Governance, Security and Fraudulence, Ethics and Ethical Governance in ICT, Holistic Medicine and Energy Healing and Governance (Corporate, Public, IT and Dynamic). She is currently an academic at the Royal Melbourne Institute of Technology and is the Director for the Journal.