A Survey of Life Sciences Applications on the Grid

Arun Krishnan

BioInformatics Institute,
21 Heng Mui Keng Terrace, Level 3, Singapore 119613

arun@bii.a-star.edu.sg

Abstract The availability of powerful microprocessors and improvements in the performance of networks has enabled high performance computing on wide-area, distributed systems. Computational grids, by integrating diverse, geographically distributed and essentially heterogeneous resources provide the infrastructure for solving large-scale problems. However, heterogeneity, on the one hand allows for scalability, but on the other hand makes application development and deployment for such an environment extremely difficult.

The field of life sciences has seen an explosion in data over the past decade. The data acquired needs to be processed, interpreted and analyzed to be useful. The large resource needs of bioinformatics allied to the large number of data-parallel applications in this field and the availability of a powerful, high performance, computing grid environment lead naturally to opportunities for developing grid-enabled applications. This survey, done as part of the Life Sciences Research Group (a research group belonging to the Global Grid Forum) attempts to collate information regarding grid-enabled applications in this field.

Keywords Grid Computing, Distributed Applications, Life Sciences

§1 Introduction

The development of improved DNA (deoxyribonucleic acid) sequencing technologies in the 1980s and 1990s heralded the start of a new era in biology. Biology and in fact biomedical science has slowly transformed into a multidisciplinary arena where there is a confluence of diverse fields of research such as

genetics, molecular biology, computer sciences, mathematics, biostatistics and bioinformatics. Thus departmental distinctions have started to blur as a result of which multidisciplinary programs have emerged to address specific problems. A vast amount of data is also being generated from DNA microarrays, mass spectrometry, DNA sequencing and structure analysis. In order to be useful, the data acquired with these technologies needs to be processed and interpreted. The need for processing this vast and exponentially growing information in a relatively short period of time has resulted in the development of high throughput techniques. This high-throughput mode however, requires computational aids such as sequence assembly tools that can effectively process the data obtained from DNA sequencers. In addition, it also requires large parallel computing resources.

The field of high performance computing has undergone even more rapid and drastic changes over the past few years. Improvements in the performance of processors and networks have made it feasible to treat collections of workstations, servers, clusters and supercomputers as integrated computing resources or *Grids*. However, heterogeneity, on the one hand allows for scalability, but on the other hand makes application development and deployment for such an environment extremely difficult. For grids to be widely useful, application design must be flexible enough to exploit widely differing and geographically distributed resources, matching application requirements and characteristics with grid resources. Grid computing, as a distributed computing framework, offers a powerful high performance computing environment, particularly for coarse-grained, data-parallel applications. The field of biosciences is one which lends itself easily to be adapted to the grid computing environment with its rich array of embarrassingly parallel applications.

The large resource needs of bioinformatics allied to the large number of data-parallel applications in this field and the availability of a powerful, high performance, computing grid environment lead naturally to opportunities for developing grid-enabled bioinformatics applications.

This survey was carried out on behalf of the Life Sciences Research Group[1)] which is a part of the Global Grid Forum[2)]. The survey was conducted in order to obtain information about currently existing grid-enabled/grid-aware applications in the life sciences area. The survey received responses from over 16 different grid organizations around the world. Although this survey does not claim to be exhaustive, it aims to provide a clear idea of the various disciplines

within the life-sciences area that use grid computing to solve complex problems.

This survey partitions the applications into the following broad categories by application area for purposes of discussion:

- Genomics/Proteomics
- Visualizations/ Simulations / Imaging
- Database Applications
- Molecular Modeling/Computational Chemistry

The rest of the paper is organized as follows. Section 2. discusses the characteristics of applications that can typically be grid-enabled. Section 3. presents the different grid-enabled applications in the field of life sciences. Section 4. encapsulates the observed trends in the different applications in this area while Section 5. summarizes the findings of this paper.

§2 Grid Applications

Despite the fact that faster networks and processors have made large-scale distributed computing a viable option, there still remains a dearth of real applications that can seamlessly run across distributed, heterogeneous domains. Grid application development remains limited to a small class of applications due to a number of issues. The foremost among these is problem granularity. As a result of the large latencies involved in a grid environment, parallel applications written for shared memory architectures or even for clusters are not amenable to being grid-enabled. Significant effort needs to be expended in order to re-engineer the problem by repartitioning the problem space to obtain a coarser level of granularity. As a result of this limitation, not all problems *can* be grid-enabled. In addition, grid applications need to have minimal inter-node communication since, large inter-node latencies on the grid as well as relatively lower (as compared to intra-cluster or intra-machine) bandwidths can lead to extremely low computation to communication ratios.

Another challenge for grid application developers lies in the absence of generic schedulers for grid environments. As a result, application developers need to incorporate aspects of scheduling into the applications. Although application level schedulers[3)] and problem solving environments are being developed, these are not ubiquitous enough for generic application development.

In addition to the above mentioned problems for grid-enabling applications, the wide disparity in operating systems (OS) and system architectures provides additional challenges for writing grid-applications. Varying architec-

tures and OS require applications to be compiled with different and sometimes incompatible libraries.

Keeping in mind the challenges mentioned above, it seems reasonable to classify grid applications that can be developed using currently available technology into the following main classes, viz.,

1. Single Program Multiple Data (SPMD): This category also includes the class of parameter sweep applications. The primary characteristic of this category of applications is the very minimal inter-node communication. These applications typically tend to be embarrassingly parallel in nature.
2. Hierarchical Parallelism: Applications belonging to this category are those that have a mix of fine-grained and coarse-grained parallelism. These applications can typically be re-engineered so that fine-grained parallelism at the intra-cluster level and coarse grained parallelism at the inter-node level coexist.
3. Work Flow: These applications are typically made up of a number of distinct stages that can be run independently of one another. In a sense, these applications can also be said to be embarrassingly parallel since the individual components making up a pipeline are mostly independent.
4. Data-driven applications: Applications belonging to this class are not computationally intensive. They deal with enabling the access of data from remote locations.

Furthermore in developing an application for the grid, it is essential to have a unifying middleware that can provide a transparent interface to the underlying protocols. Globus[4] has emerged as a de facto standard for grid middleware. Most applications that are discussed in the following sections have been developed using Globus as the underlying middleware.

§3 Life Sciences Applications

3.1 Genomics/Proteomics

The field of bioinformatics, which is the application of computing and mathematical sciences to the analysis of biological data, has led to a revolution in the life sciences industry thereby leading to an explosion of data. This enormous

growth in data in recent years is due in large part to the explosive growth in genomic data resulting from major advances in laboratory automation. Some estimate that life sciences data volume is doubling every six months; even faster than Moore's law. The explosion of data from the decoding of the human genome has created a need for enormous compute power. Grid computing is a source of enormous untapped power that could revolutionize life sciences research. The nature of genomic and proteomic data analysis is such that it can be easily adapted to the grid computing environment.

The data in genomic sequence analysis consist primarily of sequences of either nucleotides (DNA or RNA) or amino acids. Genomic analysis essentially involves either looking for patterns in individual sequences, comparing pairs of sequences, or working with small families of sequences to reveal patterns that can be compared to other sequences[5]. Each of these approaches is similar in the sense that the computation is local; that is, it requires working with only a small subset of a vast collection of sequences. Since the above mentioned approaches consist of a number of largely independent problems, the task of analyzing large sets of data can be easily subdivided. It is this embarrassingly parallel nature of genomics applications that has led to the large-scale adoption of high-throughput techniques in bioinformatics research environments.

[1] Sequence Analysis

Genomics applications thus, are the most widely visible bioinformatics applications on the grid. Perhaps the most ubiquitous grid application in this area has to be grid-enabled BLAST. BLAST (Basic Local Alignment Search Tool)[6] is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA. BLAST uses a heuristic algorithm that seeks local as opposed to global alignments and is therefore able to detect relationships among sequences that share only isolated regions of similarity[6]. BLAST can be run locally as a full executable and can be used to run BLAST searches against private, local databases, or downloaded copies of the NCBI[7] databases. BLAST binaries are provided for Macintosh, Win32 (PC), LINUX, Solaris, IBM AIX, SGI, Compaq OSF, and HP UX systems[8].

BLAST can however only run a single query against a database. Wrapper programs are available that can run high-throughput versions of BLAST on a cluster[9, 10, 11]. Typically, high-throughput BLAST applications in a dis-

tributed framework resort to "task farming" in order to distribute queries across the various nodes of the cluster. The same methodology can also be applied to distribute jobs (queries) to the different nodes on a grid. Metery et al [12] provided a proof of concept for running a high-throughput BLAST application on a Globus-enabled grid. They also studied a number of scenarios with respect to the staging of files, executables and databases from remote locations. Krishnan [13] created a standalone application called GridBLAST that can be installed on any Globus-enabled machine as well as studied the effect of problem sizes on the efficacy of porting such applications on the grid. Krishnan *et al* also created a similar high-throughput gene-finding application called GridWise by utilizing the same methodology as GridBLAST.

OBIGrid[14], a bioinformatics grid in Japan, has developed a number of applications for the grid. OBIGrid has tested out the application of BLAST[15] on a distributed grid environment where the databases are not replicated on every server. Servers within the same country NFS mounted the databases while servers in another country utilized Grid Security Infrastructure Self-certifying File System (GSI-SFS) to access the remote databases. The BLAST executables were pre-installed on the servers and the jobs were spawned on the remote nodes using the Grid infrastructure (Globus).

Another project that deals with sequence analysis and comparative genomics is the Encyclopedia of Life (EOL)[16] project being carried out by the San Diego Supercomputer Center (SDSC). SDSC has developed and implemented a grid-enabled pipeline for the EOL project. Their effort rides on the success achieved by APPLeS[3], the grid-level scheduler for scheduling jobs on the grid. The pipeline consists of a number of stages that are linearly dependent on one another, with the output of one stage being the input to the succeeding stage. The pipeline has been grid enabled by distributing tasks to the grid nodes using APST to schedule the jobs.

[2] DNA Assembly

DNA assembly deals with composing long sequence fragments from shorter ones through analyzing their common elements. PROGRESS[17], a common enterprise of several Polish scientific institutions and Sun Microsystems, has implemented a grid-enabled version of a DNA assembly program called ASM[18]. The computational problem arises because current biological techniques result in short fragments of genetic material. Blazewicz *et al*[18] created a distributed ver-

sion of the DNA assembly algorithm based on the master-slave approach. The master process, besides performing the sequential portion of the computation also acts a supervisor of the entire process. It distributes the tasks to the slaves, coordinates I/O operations with the slaves, as well as collates results from the different slave processes and returns it to the user.

[3] Proteomics

Although the amount of data accumulated in terms of protein sequences and structure is markedly less than that for genomic sequence data, it is probably more important since the holy grail of genomics and proteomics is the determination of protein structure (and hence the biological functions of proteins). Many functional properties of proteins have been found to be dependent on some typical parts of their three dimensional structure. Families of proteins share a common underlying three-dimensional structure and their identification often leads to an understanding of their functionality. Thus the problem of identifying the common substructure shared by two protein molecules, based on the similarity of their three dimensional structure has received considerable attention.

Apostolico *et al*[19] developed a protein structure comparison methodology based on indexing techniques that use geometric properties of the proteins to build and query a hash table where the proteins are stored in a redundant way. The building of the hash table is compute-intensive; the retrieval of candidate matches between a query protein and the proteins stored in the database is fast and efficient. They developed a grid-enabled version of the software that essentially incorporates the two phases of hash table construction and querying. The hash table is partitioned across the grid nodes by using a master-slave methodology. Each slave node, receives its complement of proteins from the master node and builds and maintains its portion of the table.

The search for similarity of a query protein follows the same master-slave paradigm. The master distributes the query protein to the slaves and then waits for the slaves to run the query against their portions of the hash table. On completion, the master collates the results from all the slaves.

Another application that has been grid-enabled by Koita *et al* is InterProScan[20]. InterProScan is a genomic application program that combines different protein signature recognition methods native to the member databases into one resource with lookup of corresponding InterPro and GO annotation. InterProScan uses

protein sequences as input data, and executes multiple scanning tools with multiple databases against the inputs. The scanning tools can be executed independently and hence this makes it ideally suited for being grid-enabled. In order to grid-enable the suite of applications, Koita *et al* first parallelized and grid-enabled HMMER[21] by using MPICH-G2. The grid-aware version of InterProScan, obtains node information from GIIS and then schedules the jobs on the grid nodes based on certain heuristics. The input files and databases as well as the results are copied to and from the initiating node using GridFTP.

[4] Phylogenetic Analysis

Phylogenetic trees are very useful for studies related to molecular biology and evolution, particularly to predict the function of a new gene, to identify important regions in genomic sequences, to study evolution at the molecular level and to determine the phylogeny of species. Various techniques exist for determining phylogenetic trees that differ by their speed and accuracy. The reliability of a phylogenetic tree is determined by evolving a consensus tree (procedure known as bootstrapping) from a set of generated trees.

The parallelization of the process of building phylogenetic trees can be useful either for calculating a number of independent trees (using a fast process) or for building a consensus tree through the bootstrapping procedure that is accurate, albeit slow. Silvestre and Duret[19] have developed a grid-enabled application for constructing phylogenetic trees. Each individual tree calculation is launched on a node of the grid. The distributed bootstrapping procedure involves the collation of the different trees generated and the choice of a consensus candidate tree from among them. The authors have tested the performance of the application on a small scale but are yet to test its efficiency on a larger grid. They have also developed a an easy-to-use, client-server application called *PhyloJava* that provides controlled access to phylogenetic applications on the grid.

3.2 Visualizations/ Simulations / Imaging

Visualizations, simulations and imaging are essentially applications that are targeted at the needs of the medical community. Medical institutions in general suffer from a lack of computing hardware, software and expertise. Hence, a grid environment provides a near ideal platform for a resource-poor, problem-rich community. In a typical scenario[22], a medical end-user would interact with

the grid through a grid-enabled client software, a visualization tool for images and/or geometric models and some sort of a graphical user interface. On the server side, the client requests are handled by a corresponding interface which then forwards the requests to a high-end, compute-intensive, possibly distributed application.

[1] Telescience for Advanced Tomography Applications

Ellisman *et al*[23] have developed a distributed infrastructure that gives enhanced control over data collection, data processing and data management as well as increased access to specialized resources such as high-voltage electron microscopes and data analysis and visualization in a grid computing framework. The *Telescience* system, merges technologies for remote control of devices with grid computing and federated digital libraries of multiscale, cell-structure data. The Telescience Portal, the interface for the Telescience alpha project, allows the user to accomplish the complex process of remote data acquisition via telemicroscopy; Globus-enabled parallel tomographic reconstruction; advanced visualization, segmentation, and data processing tools; and transparent deposition of data products into federated libraries of cellular structure.

[2] Monte Carlo Cellular Microphysiology on the Grid

Monte Carlo simulations are often used to analyze the microscopic structure of brain tissue. One of the most successful monte carlo simulators of cellular microphysiology is MCell[24]. MCell produces highly realistic 3-D simulations of subcellular architecture and physiology, allowing unexplored aspects of neural signaling to be quantitatively modeled. Berman*et al*[25] have developed a grid-enabled version of MCell that is based on using the AppLeS Parameter Sweep Template (APST)[3] in order to spawn jobs on the grid. APST integrates many grid software components as back-ends, including NetSolve and Globus, which can be used to deploy MCell on an increasingly large number distributed resources.

[3] MRI Simulation on the grid[26]

The simulation of Magnetic Resonance Imaging (MRI) has become important in order to obtain a theoretical understanding of the complex technology. The increased interest in MRI image analysis methods along with the compute-intensive nature of simulation has led to the necessity of parallel implementations

of such applications. MR imaging[27] involves the nuclear spin system of an object to be imaged, the magnetic fields generated by the imager, the resonance of and relaxation phenomena arising from the interaction of nucleii and magnetic fields as well as signal detection and reconstruction.

Benoit *et al*[26] implemented the simulation as a number of modules dealing with individual components such as virtual objects manipulation, implementation of MRI sequences, image reconstruction and magnetization of the computation kernel. They grid-enabled the compute-intensive module dealing with the magnetization of the computation kernel, using MPICH-G running on the European data grid (that is built on top of Globus). Their solution, which involved a master-slave type approach is scalable and follows a simple divide-and-conquer scheme for parallelization.

[4] Medical Image Content based queries on the Grid

Montagnat *et al*[28] developed an image indexing and querying application over large databases. The application allows the users to register a set of medical images and their associated data on a grid data manager and to search this data by making hybrid queries involving both metadata and image content. The image content-based queries involve classical image similarity measures. Queries over image content involves triggering automatic processings. The image databases are transported to the individual computation nodes and similarity measures are then used to discover images that show similar content to a given sample. The most similar images are then transported back to the user's machine for display.

A prototype has been developed and deployed on the European datagrid. The user can query the database through a graphical interface and search for images similar in content to the query image. One job is started for each pair of images to compare. The grid middleware receives job description, processes the jobs and returns the result in the form of a simple similarity score to the user. Since all similarity measurements are independent and can be processed in parallel on several grid nodes, this application is eminently suitable for the grid computing environment.

3.3 Database

There has been an enormous growth in medical data in recent years as a result of which there is an increased need for automatic analysis. However,

the sensitive nature of medical data makes it exceedingly difficult to widely distribute medical applications over computational grids. Hence, there is an imperative need to develop techniques for distributed data management that takes into account the special requirements with respect to security of medical data.

Grid technologies offer significant benefits that can be utilized for this purpose, such as, increased computing power for complex modeling and simulation algorithms, a distributed environment with opportunities for resource sharing as well as a common architecture to access heterogeneous data. However, any application (middleware) that deals with distributed data must take into account the nature of medical data, which, though weakly structured, have strong semantics and utilize metadata to describe content[29]. Moreover, data management and replication mechanisms [30] mainly deal with flat files. This infrastructure does not cater to metadata and patient record distribution.

A distributed data management system for sensitive medical data must provide secure and reliable data storage as well as synchronization of the data across the distributed nodes. In addition, it should also provide an easy to use query service and encryption facilities to restrict access to authorized users. All these requirements make developing such a system an extremely challenging task.

There have been a few efforts to develop and deploy distributed data management systems for the life sciences and we describe them below.

[1] DM^2: A distributed medical data manager for grids

Duque *et al*[29] designed DM^2 system as a complex system involving multiple grid service interfaces and interacting processes geographically distributed over a heterogeneous environment. DM^2 acts as a gateway to the grid as well as an intermediary between the grid and trusted medical sites. The architecture allows external applications to interact with the local system and to execute locally on the same node or remotely. Thus, the DM^2 system allows users secure access to the data and to send hybrid requests over huge databases.

[2] Data Management using Storage Resource Broker (SRB)

SRB[31] is a middleware that provides applications, a uniform API to access heterogeneous, distributed storage resources including, filesystems, databases systems and archival storage systems. It uses a metadata catalog service to offer

a "collection" oriented view of data. Thus, data items belonging to a single collection can be stored in a distributed fashion across organizational domains. SRB has been used to support data-driven life science projects of which the Visible Embryo Project[32] is the most widely known. The project's purpose is to demonstrate how leading-edge information technologies in computation, visualization, collaboration, and networking can expand capabilities in science and medicine for developmental studies, clinical work and teaching.

The data collected for this project is stored initially as lossless TIFF files. The metadata for these files are then transferred via software generated XML documents. Using conversion tools resident on the UNIX machine, smaller lossy JPEG sets are created of the 2X maps for use in regular net viewing tools. The metadata as well as the lossy images are stored on a local server which provides search and visualization capabilities for users via html. The images and metadata are transferred to the central server using SRB which coordinates the transfer process as well as distributed storage of the data.

3.4 Molecular Modeling

Molecular modeling has emerged as a popular methodology for drug design by combining aspects of computational chemistry with computer graphics. Drug design using molecular modeling techniques involves screening a large number of ligand records or molecules of compounds in a chemical database to identify potential drugs through a process known as molecular docking. Docking each molecule in the target chemical database is both compute and data-intensive. However, since molecular modeling can be implemented using a master-slave paradigm, it is possible to use a distributed environment like clusters or the grid. Moreover, the large size of the chemical databases implies that there is a need to replicate the databases across distributed nodes. Hence, a distributed grid environment, with its abundance of computing resources, can play an increasingly important role in this area.

[1] Folding@Home

Perhaps the most widely used and certainly the most well known distributed application for molecular modeling is Folding@Home[33]. Folding@Home works by decomposing the protein folding problem into a number of subproblems, each of which can be run independently on a different node of the grid. For every molecule, each simulation is started from a completely different extended

state which is done to avoid the possibility of biasing the initial state toward the native state of the molecule. For each run, M clone processes, each simulating folding in atomic detail with molecular or stochastic dynamic simulations are used. Once one of these clones makes a transition, the resulting configuration is copied over to *all* the processors and the simulations are started afresh from the new configuration. This process is performed several times over several runs. Folding@Home involves running client applications on remote nodes that then communicate with the master node in order to obtain new configurations for further simulation runs.

[2] Virtual Lab

Buyya *et al*[34] have built an application called *Virtual Lab*(VL) that transforms the molecular docking process into a parameter sweep application (using Nimrod[35]) for executing jobs for docking molecules in chemical databases in parallel on distributed resources. The parametrized application contains multiple independent jobs, each screening different compounds to identify their drug potential. Since applications built with this sort of a task-farming approach have very low computation to communication ratios, they are ideal for being grid-enabled. The VL application builds on existing grid technologies and tools for performing data intensive computing on distributed resources. VL is built by using DOCK[36] software for molecular modeling, Nimrod[35] for enabling DOCK as a parameter sweep application, Nimrod-G resource broker[37] for scheduling jobs on the grid and chemical database(CDB) management tools.

[3] Protein tertiary structure determination

Tanimura *et al*[38] have developed a grid-wide version of a parallel simulated annealing with genetic algorithm (PSA/GA) algorithm for determining the tertiary structure of proteins. They implemented the grid-aware version on top of *EVOLVE/G*, a medium for the development of evolutionary computation (EC) systems on the computational grid. EVOLVE/G has a tree topology of data communication. In EVOLVE/G, there is an Agent and some Workers. Since data can be transferred between Agent and Worker, any logical model of EC can be implemented by the EVOLVE/G. For the PSA/GA algorithm, several processes of SAs run independently. After some steps, new search points are generated by genetic crossovers and the SA runs are restarted. This logical model is implemented using a single Agent and a host of Worker processes in the

Basic model and by two types of Agents and two different stages in the *Hybrid* model. In the hybrid model, the fine grained problem is solved on a cluster while the coarse grained problem is solved across clusters (on the grid).

§4 Life Sciences Applications Trends

One of the primary aims of the survey was to not only obtain a listing of the applications being developed in the life sciences area, but also to understand the technologies involved in developing grid-aware applications. The survey found that the main programming languages used in developing these applications were C/C++, Java, PERL and Fortran. Combinations of C/C++ with Java or C/C++ with PERL are also used. The operating systems used were Linux, UNIX (various flavors, WinOS and MacOS with Linux dominating. There seems to be a general trend towards commodity grids built using open source software. The preponderance of Linux machines in the grids surveyed is perhaps indicative of this trend. These grid organizations were composed of a truly heterogeneous mix of resources including single CPU machines, clusters, SMPs and NUMA machines. There were a wide variety of schedulers that were represented such as OpenPBS, PBSPro[39], SGE[40], LSF[41], Maui[42] and other proprietary schedulers.

The survey also found that most of the problems being solved on the grid are either embarrassingly parallel in nature or make use of domain decomposition techniques. This is expected in a sense, since slow network speeds remain the main bottleneck for running grid-enabled applications. Parallelization is mostly achieved by making use of MPI, threads or Unix processes.

In view of the applications presented in the preceding sections, we can come to a few generalizations about the life sciences applications that have been grid-enabled. As discussed in Section 2, only a certain class of applications can be grid-enabled keeping in mind the current state of grid computing technology. For the most part, the applications belonging to this category are embarrassingly parallel in nature. This is obvious from a cursory look at the list of grid-enabled applications, especially in the Genomics/Proteomics area. Even in other areas such as molecular modeling, where there is a mix of fine-grained and coarse-grained parallelism, it is only the embarrassingly parallel, coarse-grained part that is solved in a distributed grid environment. A consequence of this is that the most widely used computing approach seems to be a master-slave approach.

Master-slave approach combined with a parameter-sweep methodology

is another variant that is often employed in order to grid-enable applications. There are a few applications that use remote procedure calls (RPC) or RPC-like programming paradigms (eg Ninf[43]) but for the most part, applications are being written in easy to use scripting languages like PERL or Python or else by using the grid version of MPI, viz., MPICH-G[44].

One of the surprising findings of the survey is with regard to the number of database access and query applications that are being grid-enabled. Since much of the life sciences area is highly data dependent, such a trend is heartening despite the many bottlenecks associated with slow networks and small bandwidths.

§5 Summary

This survey was commissioned by the Life Sciences Grid Research Group (LSG-RG) which is a part of Global Grid Forum (GGF) with a charter to explore issues and technologies related to the integration of IT with Life Sciences on a grid infrastructure. One of the tasks of the LSG-RG is to create and maintain a survey of grid-enabled /grid-aware applications or databases in the life sciences area. With this aim in mind, the life sciences grid application survey was created at `http://www.bii.a-star.edu.sg/ggf` and the results presented at GGF7 in Tokyo, Japan.

The survey was also meant to understand the technologies involved in grid-enabling life-science applications so that the members of the community can benefit from each other's work. To this extent the paper has tried to put down in brief, the methodologies involved in developing the grid-enabled applications described above.

§6 Acknowledgments

The author would like to acknowledge the inputs given by Dr. Abbas Farazdel (IBM Corp.) for designing the survey. The author would also like to thank the Global Grid Forum and the Life Sciences Research Group for their help in providing a platform for conducting the survey. Last, but not the least, the author would like to extend his grateful appreciation to all the scientists and researchers who responded to the survey as well as provided documentation about their respective projects.

References

1) GGF. Life Science Grid Research Group. http://people.cs.uchicago.edu/ dangulo/LSG/.
2) GGF. Global Grid Forum. http://www.ggf.org.
3) Francine D. Berman, Rich Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao. Application-level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.
4) Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
5) Platform Computing. http://www.platform.com/industry/lifesciences/index.asp.
6) S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410.
7) National Center for Biotechnology Information. NCBI Homepage. http://www.ncbi.nlm.nih.gov/.
8) NCBI. BLAST Homepage. http://www.ncbi.nlm.nih.gov/BLAST/.
9) Raphael Clifford and Aaron J. Mackey. Disperse: a simple and efficient approach to parallel database searching. *Bioinformatics*, 16:564–565, 2000.
10) Mpiblast. http://mpiblast.lanl.gov/.
11) Arun Krishnan. Scatter: High throughput version of blast on a cluster. http://scatter.bii.a-star.edu.sg.
12) R. Metery, F. Hernandez, C. Imbaud, N. Jacq, and Y. Legre. Blast on the experimental grid. Technical report. WP10, DataGrid-10-TED-0105-2_4.
13) Arun Krishnan. GridBlast: A globus based implementation of blast in a grid computing framework. *Submitted for publication*, 2003. http://gridblast.bii.a-star.edu.sg.
14) Open Bioinformatics Grid. http://www.obigrid.org.
15) Shoji Hatano, Yoshihiro Ichiyanagi, Juncai Ma, Hongyu Shi, Yoshiyuki Kidoand Susumu Date, Toshiyuki Okumura, Hideo Matsuda, and Shinji Shimojo. Construction of a grid-computing network for life sciences in china. *To be Published*, 2003.
16) San Diego Supercomputer Center. Encyclopedia of Life. http://eol.sdsc.edu.
17) PROGRESS: Poznan Supercomputing and Networking Center. http://progress.psnc.pl.
18) J. Blazewicz, M. Kasprzak, B. Nowierski, T. Redlinski, R. Styszynski, L. Szajkowski, M. Werla, and P. Widera. ASM - DNA Assembly Application. Technical report, Poznan Supercomputing and Networking Center. http://www.cs.put.poznan.pl/asm/.
19) CNRS. DataGrid: Report on the 1^{st} Bio-Testbed Release. Technical report, Lead Partner: CNRS. Document Identifier: DataGrid-10-D10.3-1-1.
20) Takahiro Koita, Yasuyo Kofune, Yusuke Inoue, and Akira Fukuda. Implementation and evaluation of resource allocation for a genomic application program on the grid. *Proc. of the IASTED Int'l Conf. on Parallel and Distributed Computing and Networks(AI2003)*, pages 524–528, 2003.
21) HMMER: Profile Hidden Markov Models for Biological Sequence Analysis. http://hmmer.wustl.edu/.

22) European Commission and IST. Grid Enabled Medical Simulation Services. http://www.ccrl-nece.de/gemss/.

23) Mark Ellisman. TeleScience Project. http://ncmir.ucsd.edu/Telescience/.

24) T. Bartol and J. Stiles. MCell: General Monte Carlo Simulator of Cellular Microphysiology. http://www.mcell.cnl.salk.edu/.

25) Fran Berman and Henry Casanova. MCell: A Parameter Sweep Usage Scenario. Technical report, San Diego Supercomputer Center. http://dast.nlanr.net/GridForum/Apps-WG/scenarios/3_m*cell.html*.

26) H. Benoit-Cattin, F. Bellet, J. Montagnat, and C. Odet. Magnetic Resonance Imaging (MRI) Simulation on a Grid Computing Architecture. *Proc. of IEEE CCGrid03*, 2003. Biogrid '03, Tokyo, Japan.

27) Z. P. Liang and P. C. Lauterbur. *Principles of Magnetic Resonance Imaging. A signal processing perspective*. IEEE Press, 2000. New York.

28) J. Montagnat, H. Duque, J. M. Pierson, V. Breton, L. Brunie, and I. E. Magnin. Medical Image Content-Based Queries using the Grid. *Proc. of IEEE CCGrid03*, 2003. Biogrid '03, Tokyo, Japan.

29) H. Duque, J. Montagnat, J. M. Pierson, L. Brunie, and I. E. Magnin. DM^2: A Distributed Medical Data Manager for Grids. *Proc. of IEEE CCGrid03*, 2003. Biogrid '03, Tokyo, Japan.

30) H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and object replication in data grids. 10^{th} *IEEE Symposium on High Performance and Distributed Computing (HPDC2001)*, August, 2001.

31) Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. The SDSC Storage Resource Broker. *Proc. CASCON'98 Conference*, Nov.30–Dec.3 1998.

32) National Library of Medicine. Visible Embryo: Human Embryology Digital Library and Collaboratory Support Tools. http://netlab.gmu.edu/visembryo/overview.htm.

33) Stefan M. Larson, Christopher D. Snow, Michael R. Shirts, and Vijay S. Pande. *Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology*. Richard Grant, editor, Horizon Press, 2002.

34) Rajkumar Buyya, Kim Branson, Jon Giddy, and David Abramson. The Virtual Laboratory: Enabling Molecular Modeling for Drug Design on the World Wide Grid. *The Journal of Concurrency and Computation: Practice and Experience*, 2002.

35) Abramson D., Sosic R., Giddy J., and Hall B. Nimrod: A tool for performing parametrized simulations using distributed workstations. *Proceedings 4th IEEE symposium on High Performance Distributed Computing, Virginia*, August, 1995.

36) Ewing A. DOCK Version 4.0 Reference Manual. Technical report, University of California at San Fransisco (UCSF), USA, 1998. http://www.cmpharm.ucsf.edu/kuntz/dock.html.

37) Buyya R., Abramson D., and Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. *Proceedings 4th IEEE symposium on High Performance Computing in Asia Pacific Region (HPC Asia2000)*, 2000.

38) Yusuke Tanimura, Tomoyuki Hiroyasu, Mitsunori Miki, and Keiko Aoi. Discussion on Evolutionary Computing on the Computational Grid. *IPSJ SIGNotes High Performance Computing*, 2002.

39) R. Henderson and D. Tweten. Portable Batch System: External reference specification. Technical report, NASA Ames Research Center, 1996.

40) Sun grid engine: White paper available at. http://wwws.sun.com/software/gridware/sgeee53/wp-sgeee/index.html.

41) S. Zhou. LSF: Load sharing in large-scale heterogeneous distributed system. *Proc. Workshop on Cluster Computing*, 1992.

42) Maui Scheduler. http://supercluster.org/maui/.

43) Satoshi Sekiguchi, Mitsuhisa Sato, Hidemoto Nakada, and Umpei Nagashima. Ninf: Network based information library for globally high performance computing. *Parallel Object-Oriented Methods and Applications (POOMA)*, Feb 1996.

44) N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing (JPDC) (to appear)*, 2003.