GNARE: An Environment for Grid-Based High-Throughput Genome Analysis

Dinanath Sulakhe,[1] Alex Rodriguez,[1] Mark D'Souza,[1] Michael Wilde,[1] Veronika Nefedova,[1] Ian Foster,[1,2] Natalia Maltsev[1]

[1] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

[2] Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

Abstract

Recent progress in genomics and experimental biology has brought exponential growth of the biological information available for computational analysis in public genomics databases. However, applying the potentially enormous scientific value of this information to the understanding of biological systems requires computing and data storage technology of an unprecedented scale. The Grid, with its aggregated and distributed computational and storage infrastructure, offers an ideal platform for high-throughput bioinformatics analysis. To leverage this platform, we have developed the Genome Analysis Research Environment (GNARE) – a scalable computational system for the high-throughput analysis of genomes, which provides an integrated database and computational backend for data-driven bioinformatics applications. GNARE efficiently automates the major steps of genome analysis, including acquisition of data from multiple genomic databases; data analysis by a diverse set of bioinformatics tools; and storage of results and annotations.

High-throughput computations in GNARE are performed by using distributed heterogeneous Grid computing resources such as Grid2003, TeraGrid, and the DOE Science Grid. Multistep genome analysis workflows involving massive data processing, the use of application-specific tools and algorithms, and updating of an integrated database to provide interactive Web access to results are all expressed and controlled by a "virtual data" model that transparently maps computational workflows to distributed Grid resources.

This paper describes how Grid technologies such as Globus, Condor, and the Gryphyn Virtual Data System were applied in the development of GNARE. We focus on our approach to Grid resource allocation and to the use of GNARE as a computational framework for developing bioinformatics applications.

1. Introduction

During the past decade, the scientific community has witnessed an unprecedented accumulation of gene sequence data and data related to the physiology and biochemistry of organisms. To date, 250 genomes have been sequenced, and genomes of more than 1,000 organisms are at various levels of completion [1]. In order to exploit the enormous scientific value of this information for understanding biological systems, the information must be integrated, analyzed, graphically displayed, and ultimately modeled computationally [2].

Comparative and evolutionary analysis of the wide spectrum of phylogenetically diverse organisms represents one of the most powerful approaches for interpreting genomes and for understanding how organisms adapt to environments. Such analysis allows for systematic exploration of mechanisms that have led to diversification of biological systems on all levels of their organization: genomic, metabolic, and phenotypic. A comparative approach, however, requires the development of high-throughput computational environments that integrate large amounts of genomic and experimental data, powerful tools and algorithms for knowledge discovery and data mining, tools for collaborative analysis of biological data by the experts residing in remote locations, and scalable computational backends.

The efficiency and accuracy of genetic sequence analysis is achieved by using various bioinformatics tools and algorithms (e.g., analysis of global similarities [3] [4] [5], domain and motif analysis [6] [7] [8], analysis of the relevant structural [9] [10], and functional data). Acquiring and integrating the needed information can be extremely tedious, time-consuming, and prone to human error if done by manually. Reliable execution of such multistep analytical processes could be achieved, however, by controlled workflows and analytical pipelines. Another problem that emerges in high-throughput bioinformatics is related to the fact that most of the tools and algorithms used for analysis of genomic data are CPU-intensive, requiring computational resources beyond those available to researchers at a single location. The aggregated and distributed computational and storage infrastructure of the Grid offers an ideal platform for mining biological information at this large scale. A number of groups are working on utilizing Grid technologies for bioinformatics purposes. Examples include the Integrative Genome Annotation Pipeline (iGAP), which has been used by the international consortium "Encyclopedia of Life" [11] for the extensive annotation of protein sequence data; myGrid [12], a large-scale Grid-based European effort; the North Carolina

BioGRID [13]; EUROGRID [14]; and the Asia Pacific BioGrid Initiative [15].

We have developed and continue to extend a system that uses Grid technology to address the needs of high-throughput genetic sequence analysis. This Genome Analysis Research Environment (GNARE) is a high-performance, scalable computational environment that allows efficient automation of the major steps of genome analysis, including data acquisition from diverse genomic databases and analysis by several bioinformatics tools and algorithms. GNARE also expedites the process of storing the results of analyses and annotations. It is based on Grid technology (specifically the Globus Toolkit® [16], Condor [17], and the GriPhyN virtual data system [18]) and uses the computational resources of GRID2003 [19], TeraGrid [20], and the DOE Science Grid [21] to perform high-throughput computations. GNARE's flexible architecture allows users to tailor the genome analysis process to their individual needs. The system can function in an automated mode as well as interactively through a Web-based interface.

This paper describes the GNARE implementation and its automated database update pipeline GADU, our experiences using GNARE on the Grid, and our efforts to increase the application's computational power and speed through further Grid integration and enhancement. We also describe scientific applications that use GNARE for high-throughput analysis and annotation of genomes.

2. System Overview and Design

GNARE comprises three main components, as illustrated in Figure 1: GNARE Architecture. GADU is the main engine that executes computationally intensive workflows on the Grid and performs the Integrated Database updates. The Integrated Database (see Section 4) warehouses sequence data and annotations from the monitored public databases as well as the results of data analyses from the GADU update engine. The third component is the set of Web-based applications that use the Integrated Database (see Section 5.2) and GADU's analysis services.

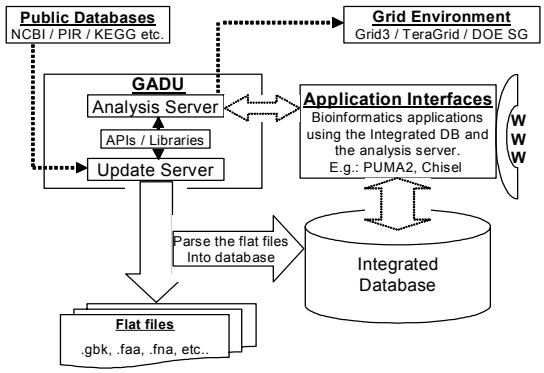

Figure 1: GNARE Architecture

GADU serves as the heart of the system. It acts as a gateway to the Grid, handling all computational analysis for the GNARE system. It is an automated, scalable, high-throughput computational workflow engine that enables the Grid execution of bioinformatics tools. The interpretation of every newly sequenced genome involves the analysis of sequence data by a workflow pipeline composed of multiple bioinformatics tools, the execution of result and annotation parsers, and other intermediate data-transforming scripts. The GADU implementation comprises two modules: an Analysis Server and an Update Server. The Analysis Server automatically creates workflows in the abstract Virtual Data Language, based on predefined templates (Section 3.5), which it then executes on distributed Grid resources such as Grid2003 and TeraGrid. The Update Server updates the Integrated Database with recently changed data from a set of monitored public databases (currently including NCBI RefSeq [22], PIR [23], InterPro [6], and KEGG [24]).

In the following sections, we describe the implementation details of each of the components of GNARE.

3. The GADU Analysis Server

We start with the GADU Analysis Server, which is responsible for executing bioinformatics analyses on the Grid. Its components are shown in Figure 2 and described below.

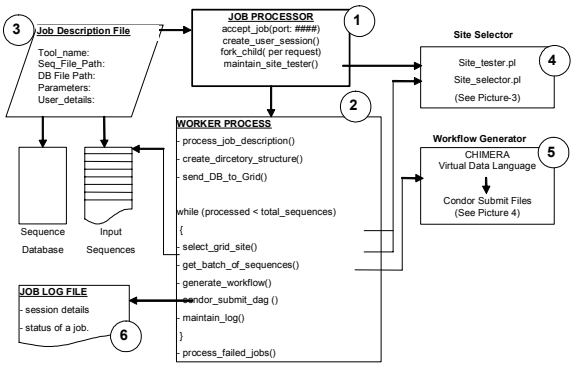

Figure 2: GADU Analysis Server

3.1. Job Description File

The *Job Description File* (item 3 in Figure 2) describes a job that may involve simply running a bioinformatics tool on the local machine, or alternatively the execution of a predefined complex workflow on the Grid.

For example, the analysis of a set of protein sequences using BLAST [25] against some database is represented in the Job Description File with all the information required to perform this analysis, including the tool name (BLAST), path of the input sequence file, path of the database sequence file, and parameters to be used for BLAST. Similarly, every analysis (e.g., BLOCKS [26], PFAM [27], TMHMM [28]) can be represented by a Job Description File appropriate for the tool. Using the information from this file, the Job Processing Server along with Workflow Generator creates the actual workflow in a Virtual Data Language and eventually in the form of a condor DAG.

3.2. Job Processing Server

The *Job Processing Server* (item 1 in Figure 2) accepts a Job Description File and creates a worker process (item 2) to handle the job. Other stand-alone services are also invoked at this time, namely, the Site Selector (item 4) and Workflow Generator (item 5). In addition to creating the worker process, the server also takes care of creating a session for each job and controls the "Site Selector" so as to keep an updated list of good working sites for job submission. The site selector is explained later.

3.3. Worker Process

The worker process (item 2 in Figure 2) determines how to handle each job based on the information in the Job Description File. The worker process first creates the directory structure for the job and then sends the sequence database (e.g., in the case of BLAST) to all usable sites on the Grid. A list of usable sites is collected from the "Site Selector" (item 4).

The next step is to create a batch job where we select a batch of query sequences to be submitted to a selected Grid site. The worker process first asks for a "good performing site" from the site selector. Based on the information it gets back (explained in next subsection), it picks a batch of sequences from the original input sequence file and sends them to the site selected. For example, if the Site Selector picked a site with 20 CPUs to be used for 1,000 sequences to be processed for BLAST, then the worker process would pick a batch of 1,000 sequences from the main input file after the last sequence processed. Next, the worker process calls the Workflow Generator (item 5 in Figure 2, described in detail in Section 3.5), which encodes the workflow in an abstract Virtual Data Language (VDL). Condor represents a workflow in the form of a DAG and executes the workflow on the selected Grid site. Once the Worker Process submits a batch job, or rather a Condor DAG, to the selected Grid site using Condor, it writes all the details of the job to a log file (item 6 in Figure 2) and goes back to generate the next batch job. The worker process continues in this manner until all sequences have been processed.

3.4. Site Selector

One challenge in using the Grid reliably for high-throughput analysis is monitoring the state of all Grid sites and how well they have performed for job requests from a given submit host. If we are executing a workflow that may submit large numbers of jobs to different Grid sites over a period of several days, it is important to keep track of which sites are available to run jobs at different times. We view a site as "available" if our submit host can communicate with it, if it is responding to Globus Toolkit job-submission commands, and if it will run our jobs promptly, with minimal queuing delays.

To address this issue, we have developed a Site Selector (item 4 in Figure 2, and also Figure 3) that uses information collected at the submit host to determine which sites meet the required responsiveness criteria.

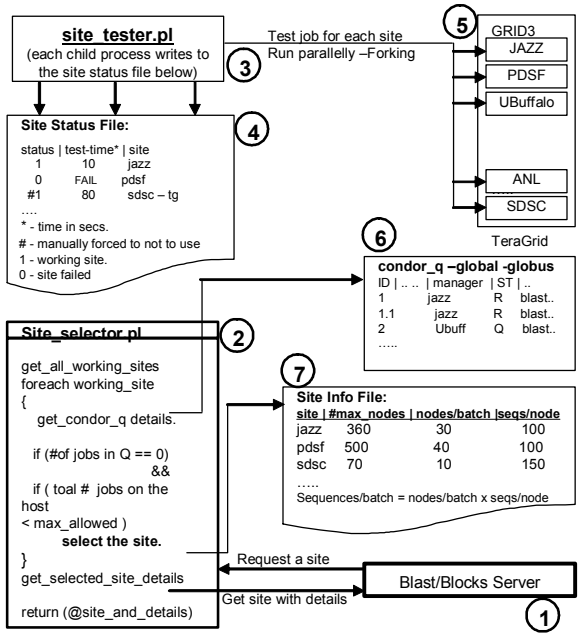

Figure 3: Implementation of the Site Selector

The site tester ("probe") script (item 3 in Figure 3) can be started and controlled manually or by using the Job Processing Server. The script submits a small test job to all Grid sites of interest and then records for each site whether the site responded and, if so, its performance (i.e., response time). The resulting Site Status File (item 4) notes for each site a status (either the site responded correctly, or the site responded incorrectly or failed to respond before a timeout period) and the time taken for the site to respond. We update this Site Status File regularly to maintain the current status of all the sites.

The Site Selector then uses the information in the Site Status File to select the site to which we can submit our next job. The selector scripts takes into consideration the state of the Condor queue at the submit host (item 6 in Figure 3) in order to make a decision. Whenever there is request from the "Job Processing Server" to select a good site, the Site Selector selects all the sites from the Site Status File that have been flagged as acceptable by the tester script. For each site it then looks at the submit host's Condor-G queue (using the condor_q command) for the number of jobs that have already been submitted to that site. If all the previously submitted jobs to this site are in "Running" state or if there are no jobs submitted to this site, then it selects the site for the next job. If, however, even one job at this site is waiting to run, then the site in consideration is not selected, and the selector scripts looks at the next site. Once a job has been submitted, the selector script makes sure that the same site is not selected again for a specified period, so that the newly submitted job has time to show up in the Condor queue.

Apart from selecting a site, the site selector also returns job-specific information to the requester. Based on the statistics of the previously executed jobs, it creates a configuration file, the Site Info File (item 7). This file records the maximum number of nodes (or CPUs) at each site, the number of nodes to be requested for each batch job submitted at this site, the number of sequences to be processed at each node, and other site-specific information. Based on this configuration file, the selector script also calculates the number of sequences that should be processed in a single job at the selected site and the number of nodes to be requested for that job. This information is all returned to the requester, which in our case is the Worker Process.

3.5. Workflow Generator

Having determined the site to which a job should be directed, the worker process assembles a complete description of the job that is to be executed and passes this information to the Workflow Generator (item 5 in Figure 2). The Workflow Generator is then responsible for producing a workflow suitable for execution in the Grid environment. This task is accomplished through the use of the GriPhyN virtual data system's "virtual data language" (VDL) [29]. VDL provides simplified, abstract access to large-scale Grid computation and storage resources. It also provides the ability to

- track accurately the provenance of results of the workflows, describing how they were obtained from transformations of input data;
- discover data through tools that search for specific transformations;
- produce new analysis work based on previously executed work, which allows for the comparison of transformation patterns executed at different times; and
- audit and disseminate results.

Figure 4 illustrates the six-stage workflow produced for a simple comparative analysis of 100 protein sequences, grouped into five sets, through the BLAST tool. The stages include transferring data to and from Grid storage servers, partitioning input data for the subsequent BLAST process, parsing specific information that the user wants to capture from protein sequences, and concatenating the final results.

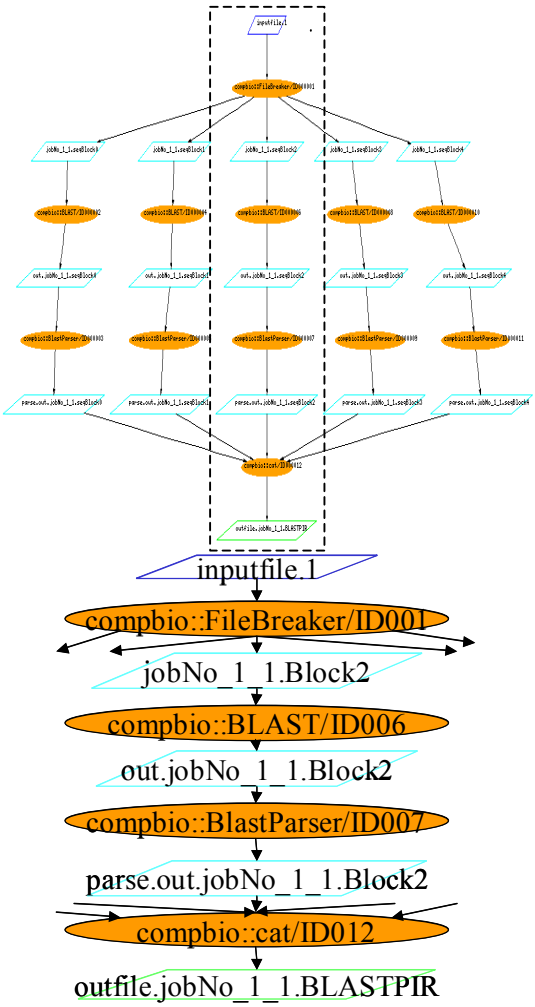

Figure 4: BLAST workflows
Top, a 6-step workflow with 5-way parallelism; below: the center-path details.

This workflow is represented in VDL via a set of *transformation* definitions (abstract interfaces that describe an application program such as BLAST, Blocks, or result-parser) and a set of *derivations*—in effect, function calls that specify inputs such as genome sequence files, output files from comparative analysis tools, and textual parameters: in other words, all the information coming from the worker process.

The VDL fragments below illustrate the notation. The first two statements define the transformations FileBreaker and BLAST shown in Figure 4. These transformation definitions act as function definitions and specify the formal arguments to an application, as well as the details of how those arguments are passed to and from the application represented by the TR definition. In the case of FileBreaker, the arguments include a genome sequence input file, the subsequent output files, and the number of computer resources to use in the Grid environment.

```
TR FileBreaker(input filename, none nodes,
output sequences[], none species) {
  argument = ${species};
  argument = ${filename};
  argument = ${nodes};
  profile globus.maxwalltime = "300";
}

TR BLAST( none OutPre, none evalue, input
query[], none type ) {
  argument = ${OutPre};
  argument = ${evalue};
  profile globus.maxwalltime = "300";
}

DV jobNo_1_1separator->FileBreaker(
  filename=@{input:"inputfile.1"|rt},
  nodes="5",
  sequences=[@{output:"job1.0":"tmp"},
             @{output:"job1.1":"tmp"},
             @{output:"job1.2":"tmp"},
             @{output:"job1.3":"tmp"},
             @{output:"job1.4":"tmp"} ],
  species="Aeropyrum_Pernix"
)
```

Figure 5: VDL for BLAST workflow

The third statement specifies a FileBreaker derivation. Derivations, defined by "DV" statements, specify the actual arguments to be passed to a transformation. File names used as arguments in DV statements are "logical names," mapped to physical file names at run time.

Data transfer for VDL is performed automatically and transparently for the user. For example, the physical file for the logical filename "inputfile.1" will be transferred automatically to the site selected for execution of the FileBreaker transformation via GridFTP [30], which provides secure, efficient data movement in Grid environments. Input files to transformations are automatically located in the Grid by searching for physical copies of a logical file in a replica location service such as RLS [31. Output files are automatically cataloged in the same location service for use in subsequent transformations and workflows. In the transformation "BLAST," above, we use the "profile" feature of VDL to specify the run-time limit for that process. VDL profiles permit parameters to be passed to components of the run-time environment.

Once the VDL for the workflow is written, the worker process invokes the GriPhyN virtual data system to execute the workflow at the Grid site selected previously by the site selector. This task is achieved via the DAGman (Directed Acyclic Graph Manager), a metascheduler for Condor, which submits jobs to remote site via Condor-G in an order determined by specified interdependencies, which in this case are derived from the VDL specification.

4. Update Server and Integrated Database

As shown in Figure 1, the Update Server helps in the acquisition of genomic data from the public databases like NCBI, PIR, PDB [32], KEGG and parses and uploads the data into the Integrated Database. It also uploads the parsed results of the various analysis tools and workflows into the integrated database.

The Update Server uses its library of parsers for all the different databases (flat files) that it downloads from the public databases. The acquisition process can be automatically executed at a predefined interval of time, in which case the server checks for new updates to download, or it can be started manually whenever there is a new release of updated data. Once the databases are downloaded and parsed, the Update Server uploads the data into the integrated database. The uploading process has been parallelized locally by forking appropriate number (based on the maximum permitted number of database handles) of child processes that upload the data using SQLLoader. Similarly whenever we have parsed results from the various analysis tools and predefined workflows that are executed by the Analysis Server, the Update Server uploads the data into the Integrated Database.

The volume and complexity of the data, as well as the diversity of the applications being developed at Mathematics and Computer Science Division, require the data to be stored in a highly integrated fashion. We have developed an integrated relational database that serves as a platform on which we can efficiently develop bioinformatics applications. The Integrated Database includes sequence and annotation data from public databases NCBI, SwissProt [33], PIR, UniProt [34], and Interpro, as well as metabolic pathway information from EMP [35] and KEGG. The database also contains the results obtained from applying different bioinformatics tools to the sequence data, for example, BLAST, Blocks, and TMHMM. The GADU Update Server ingests and integrates all of this data into an integrated warehouse database, automatically cross-referencing related entities from the various data sources.

5. Results

In this section we describe the two significant benefits that have resulted from developing and deploying the GNARE system: the high-throughput analysis of updated genome sequence data that we have achieved with GADU, and the power that GADU and the Integrated Database have provided for building powerful interactive applications.

5.1. GADU Throughput

GADU has been used extensively by the computational biology group at Argonne National Laboratory as well as by the SEED [36] project, the NIH Midwest Center for Structural Genomics (MCSG) [37], and the NIH Great Lakes Regional Center of Excellence [38]. We have developed and continue to develop automated analytical pipelines for the applications in order to manage and submit computationally intensive jobs to the Grid. One such pipeline is a very basic one designed to use the BLAST tool for comparative analysis computations on the complete protein sequence universe. There are constant updates to the protein sequence universe; at this point it comprises over 2.1 million amino acid sequences. These sequences are analyzed individually to find out similarities among them that could give clues about how sequences are related with each other. The large number of protein sequences makes these computations very computer intensive. The following results demonstrate how a Grid environment for the computations increases the efficiency of the GNARE environment.

Several of the groups mentioned above have used the BLAST pipeline for their sequence analysis and annotations. One of the first instances of using this BLAST pipeline was for the analysis of the data for the SEED project, which at the time consisted of pairwise comparisons of a database of 1.8 million protein sequences against itself. The complete analysis was performed in 84 hours using 250 nodes from the DOE Science Grid site at the Argonne Laboratory Computing Resource Center. Typically, depending on the platform and size of the database, a single BLAST process for one sequence may take anywhere from 20 seconds to 60 seconds.

Since that first run, many more BLAST pipeline invocations have been performed, and the runs have become more efficient with the addition of more Grid resources (e.g., GRID2003 and TeraGrid). The size of the protein sequence database has also increased to over 2.1 million sequences. Unlike the first run, more recent runs have been performed with the Site Selector mechanism. This mechanism decreased the labor of the earlier runs in having to manually select the sites to which to send the analysis. Indeed, even though the quantity of the protein sequence universe has increased, the real time taken to complete the whole database has decreased, as shown in Table 1. This time reduction is due mainly to the ability of the Site Selector to test the availability of a Grid site before submitting a job to that site. One of the main problems in the earlier runs was the time wasted

recomputing many of the protein sequences submitted to unavailable resources.

Table 1: Runtime results with and without the Site Seclector

Runtime (hours) / Pipeline	DOE Science w/o Site Select 1.4 million seq	GRID3 w/o Site Select 1.5 million seq	GRID3, TeraGrid w/ Site Select 1.7 million seq
BLAST	*170*	*184*	*108*
Blocks	*216*	*224*	*120*

The BLAST pipeline is not the only pipeline run using the Grid environment; however, it is the most common one that we run today. Other bioinformatics tools are also run for the complete protein sequence database using different tools (e.g. Blocks, TMHMM) to extract different vector information about the individual protein sequences. tools are as computer intensive as the BLAST pipeline and take a similar amount of time to analyze.

5.2. GNARE Applications

The use of GNARE has been essential in developing the following bioinformatics applications:

PUMA2 [39], a system for high-throughput analysis and metabolic reconstruction of genomes from the sequence data, provides a platform for interactive genome functional annotation, metabolic reconstruction and the study of evolution of metabolism and biological function. It contains analyses of over 1,000 completed and partially sequenced organisms through precomputed sequence analysis results using GNARE.

Pathos [40] provides the bioinformatics support to members of the NIH/NIAID Great Lakes Regional Center of Excellence in Biodefense research.

TarGet [41], a computational environment supporting the NIH Midwest Center for Structural Genomics (MCSG) serves researchers working on selection of protein targets of biomedical importance for the determination of 3D structure.

Sentra [42] provides an interface to a database of prokaryotic Signal Transduction proteins.

MetaGenome serves researchers of the DOE Microbial Genome program [43]. It provides bioinformatics support for the identification and characterization of organisms present in environmental samples taken from the Hanford site.

Chisel [44] provides function prediction, evolutionary and high-resolution analyses of genetic sequence data for enzymatic functions.

In order to provide up-to-date data and analyses to the researchers involved in these projects, given the growth of genomic sequence data, the use of state-of-the-art computational technologies such as distributed computing is essential.

6. Summary and Future Work

The use of GNARE has proved essential in developing applications in evolutionary analysis of genomes (PUMA2, the SEED), biodefense research (PathosDB), structural biology (TarGet DB), and bioremediation (MetaGenome), all of which depend on the availability of up-to-date annotations and rely on comparative analysis of large sets of phylogenetically diverse organisms. Use of GNARE dramatically reduced the time and human resources required for genome analysis. The increase in efficiency and speed of genome analysis enabled the expert biologists involved in these application projects to concentrate on essential biological problems without wasting time and effort on data processing.

GNARE's modular architecture is especially useful for annotation and analysis of newly sequenced genomes. Availability of new experimental results concerning functions of proteins previously annotated as hypothetical, as well as improvements in the sensitivity and accuracy of bioinformatics tools, requires periodic revisiting of previously annotated genomes and reassignment of functions using this newly acquired knowledge. The increased efficiency of genome analysis offered by the GNARE system and the Grid considerably simplifies the analysis of newly sequenced genomes and the reannotation of previously annotated genomes. GNARE can be an interface to leverage Grid resources for all biologists interested in performing such complex computations. It can hide the complex technologies involved in using distributed Grid resources and help users perform faster and better analyses.

In future work, we plan to advance further the use of Grid technology for the needs of bioinformatics applications in two main areas. First, we will provide services to bioinformatics community via a Web-based gateway, thus allowing users to submit and analyze their data by various tools and algorithms using GNARE as an entry port to the Grid environment. This server will also enable users to create customized controlled workflows using GNARE's interface. The development of such a server will allow access to advantages offered by the Grid to wide community of researchers who will not be able to use the Grid otherwise because of the lack of necessary expertise or resources. A prototype of the GNARE Web server has already developed.

Second, we will implement a virtual data warehouse that will use the Grid environment for navigation and analysis of data residing in remote locations.

Acknowledgments

We extend special thanks to the following individuals who contributed valuable advice and support: Elizabeth Glass, Jens Voeckler, Miron Livny, Zachary Miller, Alain Roy, Susan Coghlan, and the systems support groups of MCS, GRID2003, Globus, Condor, and iVDGL VDT. This work was supported in part by the U.S. Department of Energy under Contract W-31-109-ENG-38, and by the National Science Foundation under grants 86044 (GriPhyN), 122557 (iVDGL), and the NCSA Alliance Expedition "A PACI Petascale Data Quest" (PDQ).

References

1. GOLD: http://wit.integratedgenomics.com/GOLD/
2. Ideker, T., Galitski, T., Hood, L. (2001) A new approach to decoding life: systems biology. *Annu. Rev. Genomics Hum. Genet.,* **2**, 343–372..
3. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389–3402.
4. Pearson, W. R. (1994) Using the FASTA program to search protein and DNA sequence databases. *Methods Mol Biol*., **24**, 307–331.
5. Shpaer, E. G., Robinson, M., Yee, D., Candlin, J. D., Mines, R., Hunkapiller, T. (1996) Sensitivity and selectivity in protein similarity searches: A comparison of Smith-Waterman in hardware to BLAST and FASTA. *Genomics*, **38**, 179–191.
6. Mulder, N. J., Apweiler, R., Attwood, T. K., Bairoch, A., Barrell, D., Bateman, A., Binns, D., Biswas, M., Bradley, P., Bork, P., et al. (2003) The InterPro Database, 2003 brings increased coverage and new features. *Nucleic Acids Res*., **31**, 315–318.
7. Bateman, A., Birney, E., Cerruti, L., Durbin, R., Etwiller, L., Eddy, S. R., Griffiths Jones, S., Howe, K. L., Marshall, M., Sonnhammer, E. L. (2002) The Pfam protein families database. *Nucleic Acids Res*., **30**, 276–280.
8. Henikoff, S., Henikoff, J. G., Pietrokovski, S. (1999) Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, **15**, 471–479.
9. Pearl, F. M., Bennett, C. F., Bray, J.E., Harrison, A. P., Martin, N., Shepherd, A., Sillitoe, I., Thornton, J., Orengo, C. A. (2003) The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Res*., **31**, 452–455.
10. Lo Conte, L., Brenner, S. E., Hubbard, T. J., Chothia, C., Murzin, A.G. (2002) SCOP database in 2002: refinements accommodate structural genomics. *Nucleic Acids Res*., **30**, 264–267.
11. "Encyclopedia of Life" (http://eol.sdsc.edu/)
12. Goble, C., Pettifer, S. and Stevens, R. Knowledge Integration: In silico Experiments in Bioinformatics. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2004.
13. North Carolina BioGRID (http://www.ncbiogrid.org/)
14. EUROGRID, (http://www.eurogrid.org/)
15. Asia Pacific BioGrid Initiative (http://www.apbionet.org/apbiogrid/).
16. Foster, I. , Kesselman, C. (1997) Globus: A metacomputing infrastructure ioolkit. *International Journal of Supercomputer Applications* **11**(2) 115–128.
17. Litzkow, M.J., Livny, M., Mutka, M. W. Condor – a hunter of idle workstations. In *Proc. 8th International Conference on Distributed Computing Systems*, 1988, 104–111.
18. Foster, I., Voeckler, J., Wilde, M., Zhao, Y., The virtual Data Grid: A new model and architecture for data-intensive collaboration. Conference on Innovative Data Systems Research, 2003.
19. Foster, I., et al., The Grid2003 Production Grid: Principles and practice. In *Proc. IEEE International Symposium on High Performance Distributed Computing*, 2004, IEEE Computer Science Press.
20. Catlett, C. The TeraGrid: A primer, 2002. www.teragrid.org.
21. DOE Science Grid, www.doesciencegrid.org
22. *The NCBI handbook* [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; 2002 Oct. Chapter 17, The Reference Sequence (RefSeq) Project. Available from http://ncbi.nlm.nih.gov/entrez/query.fcgi?db=Books.
23. Wu, C. H., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Robert, S. (2002) The Protein Information Resource: An integrated public resource of functional annotation of proteins. *Nucl. Acids Res*. **30**(1) 35–137.
24. Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., Kanehis, M., (1999) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl. Acids Res*. **27**(1).
25. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J. (1990) *Basic local alignment search tool. J. Mol. Biol.* **215**:403–410.
26. Henikoff, J. G., Henikoff, S. (1996) Blocks database and its applications. *Meth. Enzymology* **26:**88–105.
27. Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Finn, R. D., Sonnhammer, E. L. (2000) The Pfam protein families database. *Nucl. Acids Res.* **28:** 260–262.
28. Krogh, A., Prediction of transmembrane helices in proteins, http://www.cbs.dtu.dk/services/TMHMM/
29. Foster,I., Voeckler, J., Wilde, M., Zhou, Y. (2220) Chimera: A virtual data system for representing, querying, and automating data derivation. In *Proc. 14th Conference on Scientific and Statistical Database.*
30. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Nefedova, V., Quesnel, D., Tuecke, S. (2002) Data management and transfer in high-performance computational Grid environments. *Parallel Computing* **28**(5):749–771
31. Chervenak,A., Palavalli, N., Bharathi, S., Kesselman, C., Schwartzkopf, R. (2004) Performance and scalability of a replica location service. In *Proc. International Symposium*

on High Performance Distributed Computing Conference (HPDC-13).
32. *PDB, The Protein Data Bank,* http://www.rcsb.org/pdb/
33. *Swiss-Prot, The Swiss-Prot Protein Knowledgebase*, http://us.expasy.org/sprot/
34. Bairoch A, Apweiler R, Wu C. H., Barker W. C., Boeckmann B., Ferro S, Gasteiger E, Huang H, Lopez R, Magrane M, Martin MJ, Natale DA, O'Donovan C, Redaschi N, Yeh L. S. The Universal Protein Resource (UniProt).
35. Burgard, A.P., Maranas, C.D. (2001) Review of the enzymes and metabolic pathways (EMP) database. *Metab. Eng.* **3:**193–194.
36. Overbeek, R., Disz, T., Stevens, R. (2004) The SEED: A peer-to-peer environment for genome annotation. *Comm. ACM* **47**(11):46–51.
37. Midwest Center for Structural Genomics (MCSG) http://www.mcsg.anl.gov
38. Great Lakes Regional Center of Excellence for Biodefense & emerging Infectious Diseases Research, http://www.glrce.org.
39. PUMA2 System, http://compbio.mcs.anl.gov/puma2
40. Pathos System, http://compbio.mcs.anl.gov/pathos
41. TarGet Environment, http://compbio.mcs.anl.gov/target
42. Maltsev, N., Marland, E., Yu, G.-X., Bhatnagar, S., Lusk, R. (2002) Sentra, a database of signal transduction proteins. *Nucl. Acids Res.* **30**(1): 1349-1350. http://www-wit.mcs.anl.gov/sentra
43. DOE Microbial Genome Program, http://microbialgenome.org
44. Chisel, http://compbio.mcs.anl.gov/CHISEL