Interoperability of GADU in using Heterogeneous Grid Resources for Bioinformatics Applications.

Dinanath Sulakhe[1] Alex Rodriguez[2] Michael Wilde[1, 2] Ian Foster[1, 2] Natalia Maltsev[1, 2]
[1] *Computation Institute, University of Chicago, Chicago, IL 60637, USA*
[2] *Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA*

***Abstract* - During the past decade, the scientific community has witnessed the rapid accumulation of gene sequence data and data related to physiology and biochemistry of organisms. Bioinformatics tools used for efficient and computationally intensive analysis of genetic sequences require large-scale computational resources to accommodate the growing data. Grid computational resources such as the Open Science Grid and TeraGrid have proved useful for scientific discovery. GADU is a high-throughput computational system developed to automate the steps involved in accessing the Grid resources for running bioinformatics applications. This paper describes the requirements for building an automated scalable system such as GADU that can run jobs on different Grids. The paper describes the resource-independent configuration of GADU using the Pegasus-based Virtual Data System that makes high-throughput computational tools interoperable on heterogeneous Grid resources. The paper also highlights the features implemented to make GADU a gateway to computationally intensive bioinformatics applications on the Grid. The paper will not go into the details of problems involved or the lessons learned in using individual Grid resources as it has already be published in our paper on GNARE and will focus primarily on the architecture that makes GADU resource independent and interoperable across heterogeneous Grid resources.**

Index Terms – Interoperability of Grid Resources, multiple grid resources, Bioinformatics, high-throughput computations.

1. Introduction

Bioinformatics is a "science of big numbers". It utilizes high-throughput computational analysis of genomic sequences for discovering evolutionary patterns underlying complex biological processes that produced the diversity of life on this planet. Essential for this approach is comparative and evolutionary analysis of a wide spectrum of phylogenetically diverse organisms, in order to that provide understanding of the adaptive mechanisms that led to diversification of biological systems on all levels of their organization: genomic, metabolic, and phenotypic.

When properly understood, these variations can present the answers to the following questions: How do differences observed on the genomic level affect the function of biological systems? What allows thermophilic organisms to sustain life at the temperatures above 100 degrees C? What differences on the genomic level lead to emergence of genetic diseases? Answering such questions is fundamentally important for further progress in biotechnology, medicine and bioremediation.

There has been an unprecedented accumulation of gene sequence data and data related to the physiology and biochemistry of organisms during the past decade. To date, 343 genomes have been sequenced, and genomes of more than 1500 organisms are at various levels of completion [1]. This wealth of genomic information will dramatically accelerate progress toward a comprehensive understanding of the genetic mechanisms involved in diverse biochemical processes pertinent to bioremediation, medicine, biotechnology and agriculture.

Efficiency and accuracy of genetic sequence analysis are achieved by the use of diverse CPU-intensive bioinformatics tools and algorithms (e.g., analysis of global similarities [2] [3] [4], domain and motif analysis [5] [6] [7], analysis of the relevant structural [8] [9], and functional data). Running these tools on the rapidly growing data is a time-consuming process and needs high-throughput computations to get results in a timely fashion. The aggregated and distributed computational and storage infrastructure of the Grids such as the Open Science Grid (OSG) [10] and TeraGrid [11] offers an ideal platform for mining biological information at this large scale.

We have developed a system called GADU [12], which has access to the OSG, TeraGrid and DOE Science Grid [13] resources. The opportunistic availability and the different architectures and environments of these resources make it extremely difficult to use them simultaneously through a single common system. GADU addresses these issues by providing a resource-independent system that can execute the bioinformatics applications as workflows simultaneously on these heterogeneous Grid resources. GADU is easily scalable to add new Grid resources or individual clusters into its pool of resources, thus providing more high-throughput computational power to its scientific applications.

2. GADU, the Genome Analysis Server

The Genome Analysis and Database Update system, GADU, is an automated, scalable, high-throughput computational workflow engine that executes computationally intensive workflows for the analysis of sequence data on the Grid and performs updates to the integrated database. The integrated database [12] warehouses sequence data and annotations from the monitored public databases as well as the results of data analyses using GADU.

The interpretation of every newly sequenced genome involves the analysis of sequence data by a variety of computationally intensive bioinformatics tools, the execution of result and annotation parsers, and other intermediate data-transforming scripts. GADU acts as a gateway to the Grid, handling all the high-throughput computations. GADU is implemented in two modules, analysis server and an update server. The analysis server automatically creates workflows in the abstract Virtual Data Language, based on predefined templates that it executes on distributed Grid resources. The update server updates the integrated database with recently changed data from a set of monitored public databases (currently including NCBI RefSeq [14], PIR [15], InterPro [6], and KEGG [16]).

GADU has successfully used Grid resources with different architectures and software environments like the 64-bit processors in TeraGrid and 32-bit processors in the Open Science Grid or DOE Science Grid. GADU executes its parallel jobs simultaneously on these different Grid resources. It expresses the workflows in the form of a directed acyclic graph (DAG) and executes it on a specified Grid site using Condor-G [17]. GADU uses the GriPhyN Virtual Data System [18] to express, execute, and track the results of the workflows that helps in using the grid resources.

3. Use of the Virtual Data System in GADU

In this section we introduce the use of the Virtual Data System (VDS) [18] to generate, execute, and control the workflows in Grid environments. We look at the features of VDS that makes GADU a resource-independent system that can use multiple Grids of different architectures and environments.

VDS provides tools to express, execute, and track the workflows that consist of application invocations. The workflows are expressed in a location-independent, high-level abstract language called Virtual Data Language (VDL). VDL frees the workflow from specifying details of the location of files and programs in a distributed environment.

VDS uses Pegasus [19] as a planner component to generate executable forms of the workflow expressed using VDL. Pegasus maps an abstract workflow expressed in VDL to a specified Grid resource. It generates a concrete workflow or a fully planned Condor DAG from the initial abstract workflow. These concrete workflows generated using VDS can be executed in a variety of environments ranging from the desktop to Grids such as the Open Science Grid and TeraGrid or any individual computing cluster.

3.1. Representing Transformations and Resources

The Virtual Data Language expresses an abstract workflow as transformations and derivations. Transformations are general descriptions of each executable and provide information about the input and output parameters required for each executable. The transformation does not provide the physical locations of the input and output files or the actual parameters going into the executable. Rather, it provides a template to the derivations that provides the physical locations and parameters of the applications.

```
#SITE      Transformation   PFN                          TYPE
ANL_Jazz     BLAST     /soft/apps/BLAST/bin/blastall   null
ANL_Jazz     Blocks    /soft/apps/run-Blocks.pl        null
ANL_Jazz     Chisel    /soft/apps/chisel/runChisel.pl  null
ANL_Jazz     IPRSCAN   /soft/apps/iprscan_wrapper.pl null
ANL_Jazz     globus-url-copy /soft/apps/packages/globus-
2.2.4/bin/globus-url-copy
GLOBUS_LOCATION=/soft/apps/packages/globus-2
.2.4/;LD_LIBRARY_PATH=/soft/apps/packages/globus-
2.2.4/lib;PATH=/soft/apps/packages/globus-2.2.4/bin
```

Figure 1: Sample Transformation Catalog on Jazz Cluster

```
pool ANL_Jazz {

  lrc "rls://gnare.mcs.anl.gov"
  gridftp "gsiftp:// jmayor1.lcrc.anl.gov:2812/soft/apps/gadu"
  gridlaunch "/soft/apps/gadu/bin/kickstart"
  workdir "/soft/apps/gadu/vdldata"
  universe vanilla "jmayor1.lcrc.anl.gov:2121/jobmanager-pbs"
  universe globus "jmayor1.lcrc.anl.gov:2121/jobmanager-pbs"
  universe transfer " jmayor1.lcrc.anl.gov:2812/jobmanager-fork"
}
….
```

Figure 2: Sample Site Catalog on Jazz Cluster

In VDS, each transformation or executable of a workflow installed on a high-throughput computing resource and its path on the remote resource is listed in a Transformation Catalog (TC) file as shown in the Figure 1. The TC file contains transformation information for each grid location, including the Grid site name,

transformation name, and physical location on the Grid. Each site name is mapped into the Site Catalog (SC) file containing information about the gatekeeper, job manager, and remote working directory, as shown in Figure 2. These two files together provide all the required information for a planner software to create the workflow.

Figures 1 and 2 provide a sample of the TC and SC files respectively showing information for the Jazz cluster at Argonne National Laboratory.

3.2. Executing the Workflows

The Pegasus planner in VDS maps the transformations of an abstract workflow to an execution resource based on the information derived from TC and SC files. Pegasus uses dynamic sources such as Monitoring and Discovery Service (MDS) [20] and Replica Location Service (RLS) [21] to map the data files used by the transformations during execution. If the transformations of a workflow are mapped on different resources and don't share the data, Pegasus adds new nodes in the workflow to transfer the data between the tasks. The resulting output in the form of a Condor DAG is submitted to a remote computing resource using the Condor-G's DAGMan. Condor-G and DAGMan form a powerful combination to execute a workflow on any Grid resource that provides a Globus Grid Resource Allocation and Management (GRAM) interface.

3.3. Using GADU on the Jazz Cluster

The GADU analysis server has automated all the steps involved in expressing and executing the bioinformatics application workflows using VDS as explained in the previous sections. GADU used VDS to execute the workflows involving bioinformatic applications such as Blast, Blocks, Pfam, Chisel, and InterPro for the first time on Jazz, a 350-node computing cluster that is part of the DOE Science Grid resources. The Jazz cluster is located at Argonne National Laboratory.

In order to execute a workflow on Jazz, the following steps were performed:

- Automate the steps involved in expressing, generating, and executing the application workflows inside GADU's analysis server.
- Install all the standard Grid middleware components such as Condor, Globus, and GridFTP on the submit machine.
- Install the Pegasus-based Virtual Data System and a RLS server for data items.
- Create a Globus GRAM interface on the Jazz cluster by coordinating with the site administrator.
- Install all the tools involved in the workflows on the remote cluster.
- Update the TC and SC file representing the transformations and the site specific information (shown in Fig 1 and Fig 2).

One of the main functions of GADU is to automatically perform most of these tasks in setting up the infrastructure. GADU then executes and monitors the workflows involving bioinformatics applications. Figure 3 represents GADU's configuration and the steps required to generate and execute a workflow on the Jazz cluster. All the steps performed on the submit host are automated by GADU.

3.4. Resource-Independent Configuration

GADU provides a resource-independent configuration to execute the workflows on the Grid. It can submit jobs remotely to a resource, as long as the resource provides a Globus GRAM interface (for example, the Jazz cluster). All the transformations of a workflow are expressed as Condor submit files and a DAG using Pegasus. The Condor-G submits the workflow to a remote resource using the GRAM interface and also monitors the workflow.

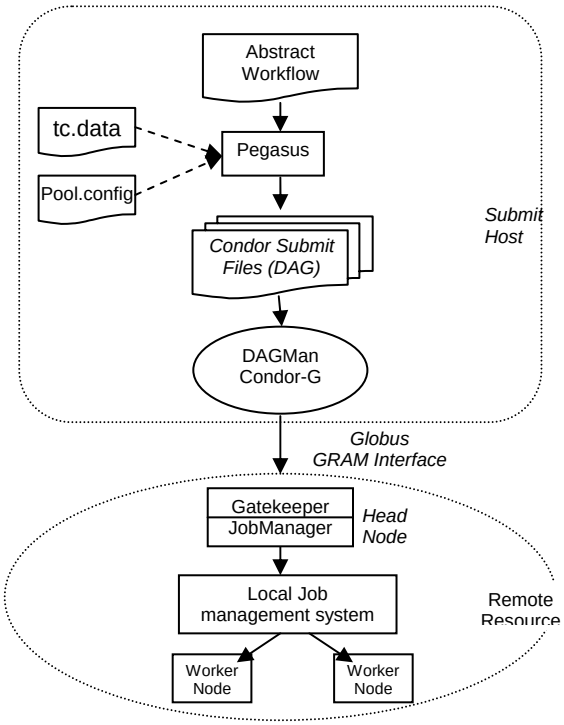

Figure 3: Workflow generation and execution on Jazz.

In the case of Jazz as well as most of the Grid resources containing multiple compute nodes, the gatekeeper acts as the single point of job submissions to

the resource. This gatekeeper provides the Globus GRAM interface and is responsible for submitting the job to the local job management system (PBS, LSF, Condor, etc.), which executes the jobs on the compute nodes.

GADU's ability to run the jobs on any high-throughput computational resource that provides a GRAM interface makes it scalable to accommodate more resources.

Figure 3 represents the configuration of GADU as a resource-independent system for generating, executing, and monitoring the workflows on a multinode cluster resource.

4. Accessing the Open Science Grid and TeraGrid

This section explains GADU's ability to easily accommodate new Grid sites into its pool of resources. It also explains how GADU automatically manages the dynamic changes in the state of the Grid resources using monitoring and information services along with the authentication and access models used at different Grids.

Open Science Grid and TeraGrid are large-scale Grid computing infrastructures that provide computational resources for scientific discovery. TeraGrid was completed in September 2004, bringing over 40 teraflops of computing power and nearly 2 petabytes of rotating storage. It is a combined resource from eight different partner sites. The Open Science Grid Consortium was formed in 2004 to enable diverse scientific communities (physics, chemistry, astrology, etc.) to access the shared resources from a common Grid infrastructure. Currently it has a collective number of more than 18000 CPUs contributed by more than 60 institutions.

Each site in the OSG provides a set of fundamental services for the users to run their jobs. The OSG software stack consists of middleware like Globus and Condor technologies packaged into a toolkit called Virtual Data Toolkit (VDT) [22]. The VDT package installed on the Grid resources provides a Globus GRAM interface for the users to submit and run their jobs remotely. Thus, it fits easily into GADU's framework. GADU treats each of the OSG site as an individual resource and installs its scientific applications on all OSG resources that can be accessed. After each installation, the site-specific information such as applications installation directory on the remote resources, Globus location, gatekeeper name, and job-manager information. is collected and added in the TC and SC files of VDS.

Similar to OSG, TeraGrid resources at the individual sites are managed autonomously, and the architecture is built by deploying Grid service layers that contain software components such as GRAM, Condor-G, GridFTP, GASS, and MDS. TeraGrid uses the NSF Middleware Initiative's (NMI) software release as the Grid software base. Similar to OSG, TeraGrid also uses GRAM protocol for secure remote access to its resources. Thus, through the use of GRAM interface, GADU automatically adds TeraGrid sites as individual sites into the SC file. This addition gives GADU access to run its applications on the TeraGrid resources.

GADU was invited to use OSG resources to run bioinformatics applications and form a virtual organization (VO). OSG resources are used on an opportunistic availability basis, and TeraGrid resources are used through a proposal requesting allocations or CPU hours.

4.1. Automated Site Management in GADU

GADU automates all the steps involved in adding a new site as well as maintaining all the sites where it can submit its workflows for high-throughput computing.

I. Adding a New Site.

In order to add a new site, GADU obtains the required information about the remote resource (i.e., gatekeeper contact string, job-managers, application and data storage directories, Globus path environment) from monitoring and information services such as GridCat [23] and MDS and adds it to the TC and SC files. The architecture of each site is also collected from these information services. For example, most of the sites in OSG are IA32 based cluster, whereas TeraGrid has a variety of different clusters with IA32, IA64 and other architectures. Based on these hardware specifications, GADU installs the appropriate scientific applications by sending a packaged tar file onto the new remote resource. All these steps are automatically performed by GADU. It is explained in detail in our previous publication on GNARE.

II. Maintaining the Existing Sites

Currently GADU has access to more than 70 heterogeneous computational resources with different architectures and environments that are listed in its SC file. The SC file contains resources from OSG and TeraGrid, each having a different architecture and environment. Individual clusters such as Jazz are also used by GADU. With so many different resources, and all being managed autonomously by different institutions, the access information of these resources may change or update regularly, making it very difficult to track. GADU can handle these changes automatically and update the new access information periodically for each site.

III. Grid Monitoring and Information Services

In the cases of the OSG and TeraGrid, which contain multiple sites whose status may change without notice, it is important to have services that provide the information of any changes. These changes include addition of a new site or change of gatekeeper, job-managers, and installation directories. Additionally, it is necessary to

monitor the status of all the sites to show the number of nodes available, number of jobs running, load ,and other useful information about the remote site.

Both the OSG and TeraGrid have their own systems providing all the required information to the clients. The OSG uses GridCat, MonaLISA [24] and some other services whereas the TeraGrid uses Inca, GPIR and MDS based information services. GADU uses these services provided by OSG and TeraGrid at various levels. GADU regularly monitors these services for any change in the state of access and automatically updates the changes. It also triggers reinstallation of the application packages whenever required. For individual clusters like Jazz that don't provide any monitoring or information services, the information has to be manually identified and recorded.

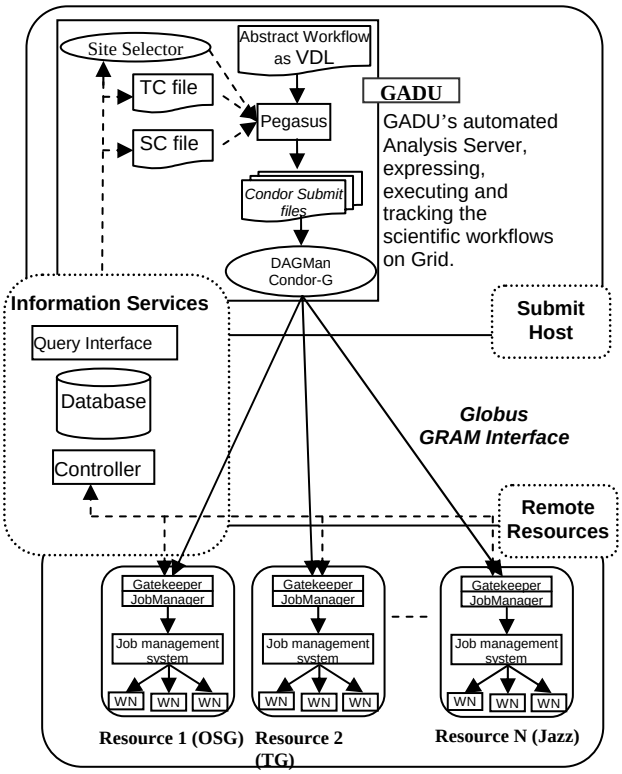

Figure 4: GADU using different Grid resources.

Figure 4 shows the various components of GADU as well as the information service and Grid sites from different Grid projects.

4.2. Authentication at Various Resources

GADU uses an X.509 certificate-based authentication scheme that utilizes the Grid Security Infrastructure (GSI) [25] protocol.

In the OSG, every user belongs to a virtual organization, which runs a Virtual Organization Membership Service (VOMS). GADU runs its own VOMS server, GADU VO. User certificates from the GADU users are added to the VOMS, which in turn gets updated by all the Grid sites of OSG providing access to the users. The TeraGrid, on the other hand, provides individual logins to all its users. A user installs his certificates in the grid-mapfile by logging into all the TeraGrid sites. For individual clusters such as Jazz, all the user's certificates are installed by the site administrator.

Once all the user's certificates are installed on the resources, GADU can submit the user's jobs securely through the GRAM interface, as explained in the previous sections.

4.3. Site Selection across Grids

GADU's access to a large pool of computational resources and their opportunistic availability makes it important to select a best available site that can execute the jobs with least amount of queuing times. Most of the high-throughput workflows executed through GADU involve running of bioinformatics tools on a large set of protein sequences. It is an embarrassingly parallel workload ideally suited for distributed computing, where a bunch of sequences are submitted to a Grid sites asking for a predefined number of CPUs.

When a large number of sequences are submitted to GADU to run through a computationally intensive tool, for example running a BLAST [26] tool for NCBI's non-redundant protein sequences containing over 3 million sequences, it usually takes a few days to get the results for all the sequences. GADU uses the embarrassingly parallel method, where it keeps sending a small set of sequences as a batch job (e.g., 1000 sequences per site) to a suitable site from its pool of sites listed in the SC file.

GADU has implemented a site selector which is described in detail in the paper on GNARE [12] and can be referred for a detailed understanding of the site selector implementation and various variables used as the criteria of selection. In brief, the site selector has a daemon that submits a small test job on a predefined interval of time that tests for availability of a site, data transfer, tools installed on a site, and the result of the test job submitted. These variables are very useful in selecting a site in the initial stages of big run such as BLAST for millions of sequences. As the jobs are being submitted to these various sites, the site selector dynamically tunes itself and gives priority to the sites based on the performance of the previous jobs at these sites, and also automatically eliminates any site that fails during runs. The site selector also looks at the local condor queue to get an estimate of the number of jobs running or in queue at the various sites. The site selector doesn't pick a site for the next job if more than a predefined number of jobs are sitting idle in the condor queue at that site, assuming that the site is perhaps busy currently. The fundamental

selection criterion is very simple, select a site that successfully completes previously submitted jobs and send more jobs to those sites that are completing the jobs faster, at the same time not overloading the remote queue. The specific times involved in each step of the workflow such as time taken by data transfer or tool execution are currently not taken into account, and the overall time from submission to the return of output file are accounted.

The data from information services such as GridCat from OSG are also used to eliminate any site that may have gone down for maintenance or if any of the services are not working. Figure 4 shows a site selector in GADU's configuration.

5. Results

GADU is extensively used by the bioinformatics group at Argonne National Laboratory for building applications for the high-throughput analysis of genomes. Applications such as PUMA2 [27], Pathos [28], Target [29], Chisel [30], and GNARE's prototype for user-submitted genomes regularly use GADU.

Figure 5 shows GADU's performance in three scenarios. GADU was first used to run BLAST on the Jazz cluster for 1.4 million sequences in the NCBI non-redundant database of protein sequences to compare against itself (In these BLAST runs, for N number of sequences, N^2 pair-wise sequence comparisons are performed, that is, each sequence in the database is compared against all other sequences as well as itself, though the comparison results against itself are not used often). It took 170 hours to process all the sequences using 200 reserved CPUs on the Jazz cluster. The projected amount of time required to run BLAST as the number of sequences increases from 1.4 to 1.5, 1.7 and 3.1 million is shown in Figure 5. Once the first stable

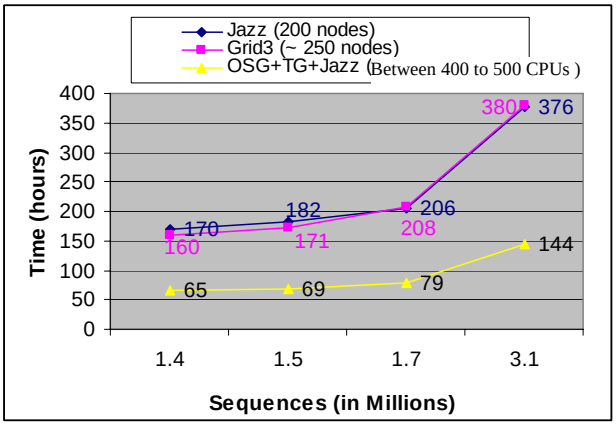

Figure 5: GADU's performance in using different Grid resources for running BLAST.

version of GADU was tested on the Jazz cluster, OSG (formerly known as GRID3) was added to the list of Grid resources in GADU. A BLAST job run on the OSG resources that involved 1.7 million sequences took about 208 hours to complete. GADU's site selector was used for this job to pick the appropriate site for each batch job. On average 250 nodes were used at any given time. GADU's access to the amount of compute resources has increased by the addition of OSG and TeraGrid resources together. But at the same time, the sequence data is growing exponentially. In the last BLAST run, 3.1 million sequences took only 144 hours using more than 30 OSG sites, 5 TeraGrid sites and 35 CPUs from Jazz, a significantly shorter time compared to the projected time using only OSG. These sites were used based on opportunistic availability and the number of CPUs used varied from 400 to 500 CPUs (500 was set as collective Maximum number of CPUs for this run – to control the load on the submit host used for the run). The site selector in GADU helped in picking the right resources among the various sites from different Grids. It was observed that the 64bit processor based sites in the TeraGrid gave better throughput compared to the 32bit processor based sites across all the grids, as BLAST performed faster on 64bit processors. When the number of jobs submitted to each Grid or the Jazz cluster is compared, more jobs were submitted to the OSG sites, as GADU was able to use more than 30 different sites in OSG compared to 5 sites in the TeraGrid.

Apart from BLAST, GADU has also run other bioinformatics tools such as BLOCKS, PFam, Chisel, and InterPro on various Grids to which it has access.

Various types of error checking are performed on the jobs to provide fault tolerance. The errors that are reported by the GRAM (e.g., a site is not reachable – goes down, tool not installed, data transfer failures) are automatically detected by Condor-G and a 'rescue' file is created in order to resubmit the jobs. For the errors where GRAM doesn't report failures; for example, if a node runs out of memory and fails to complete the job, or data transfer errors, GADU has application specific parsers that perform detailed sanity check on the output data. Due the diversity of the grid resources, we have found new unanticipated failures of the jobs that we identify only later during the actual use of the results by various applications. But with experience and repeated runs, we are able to identify most type of failures and use these heterogeneous grid resources more effectively.

GADU's ability to use multiple Grids has made it possible to analyze the biological data much faster and help the biologists in their scientific discoveries. GADU has been successfully used by many other groups and projects such as the NIH Midwest Center for Structural Genomics (MCSG) [32], the NIH Great Lakes RCE for Biodefense and Emerging Infections [33], MetaGenome

project from the DOE Microbial Genome program [34], and Shewanella Consortium for the analysis of Shewanella genomes [35].

6. Related Work

There are many Globus based execution management tools such as Condor-G, Grid(Lab) Resource Management System [36], Grid Service Broker [37], GridWay [38], VDS and others. Most of these systems are used for execution management and scheduling of the jobs on the various Grid resources. Though most of these allow easy access of Grid resources for applications, GridWay has many other features similar to GADU that help users for meta-scheduling and automated execution of jobs on heterogeneous grid resources. It provides fault recovery and allows users to migrate their jobs in case of failures.

GADU is also designed for Globus based Grids and utilizes some of these execution management systems such as Condor-G, VDS and many other softwares that are wrapped into Virtual Data Toolkit (VDT) to further automate the large workloads of running bioinformatics tools. One of the advantages of GADU is the use of VDS that allows expression of workflows in a higher level language that gets mapped onto a specified Grid. GADU automates the steps involved in utilizing the heterogeneous Grid resources. Automation of site-selection for new jobs as well as for migration of failed jobs is a unique feature of GADU. It utilized the features of VDS effectively to execution large workloads run for weeks and that requires dynamic site-selection and fault tolerance throughout the run. It can easily add new sites to the pool of sites even during the run. GADU demonstrates the advantages of using VDS to make its applications interoperable across heterogeneous resources.

7. Conclusions

In this paper we described the capabilities of GADU to use multiple Grid resources simultaneously for running bioinformatics tools. The exponential growth in genomic sequence data requires distributed Grid resources for its faster analysis using a variety of Bioinformatics tools and algorithms. Use of the Pegasus-based VDS system allows GADU to add more Grid sites easily into its pool of computational resources. It shows the interoperability achieved by GADU In this paper we are not concentrating on the other important features of VDS such as data provenance; we describe only those features that help in using multiple Grid resources through GADU.

Currently GADU is used to support various applications built by the Bioinformatics group at Argonne National Laboratory. In future, we plan to provide services to the bioinformatics community via a Web-based gateway, thus allowing users to submit and analyze their sequence data by a variety of tools and algorithms using Grid resources. A prototype of such a system has been built under the framework of GNARE's User Models. From the examples provides in the Results section, we can see that using multiple Grids has significantly reduced the total amount of time taken to execute BLAST.

The availability of GADU to the public usage is currently only via GNARE portal which is widely used for the analysis of user submitted genomes.

Acknowledgments

We extend special thanks to the following individuals who contributed valuable advice and support: Elizabeth Glass, Mark D'Souza, Jens Voeckler, Miron Livny, Alain Roy, Susan Coghlan, and the systems support groups of MCS, OSG, TeraGrid, Globus, Condor, and iVDGL VDT, Center for Grid Technologies at ISI, USC. This work was supported by the U.S. Dept. of Energy, under Contract DE-AC02-06CH11357, 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://www.genomesonline.org/
2. Pearson, W.R. (1994) Using the FASTA program to search protein and DNA sequence databases. Methods Mol. Biol., 24, 307-331.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.

8. 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.
9. "Encyclopedia of Life" (http://eol.sdsc.edu/)
10. Open Science Grid, http://www.opensciencegrid.org
11. Catlett, C. The TeraGrid: A Primer, 2002. www.teragrid.org
12. Sulakhe, D., Rodriguez, A., D'Souza, M., Wilde, M., Nefedova, V., Foster, I., Maltsev, N. (2005) Gnare: automated system for high-throughput genome analysis with grid computational backend. J Clin Monit Comput., 19(4-5): 361-369.
13. DOE Science Grid, www.doesciencegrid.org
14. 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
15. 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. , Nucleic Acids Res., 30(1), 35-37.
16. Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., Kanehis, M. (1999) KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 27(1), 29-34..
17. Frey, J., Tannenbaum, T., Foster, I., Livny, M, and Tuecke, S. (2002) Condor-G: a computation management agent for multi-institutional Grids. *Cluster Computing*, 5, 237–246.
18. Foster, I., Voeckler, J., Wilde, M., Zhou, Y.(2002). Chimera: A virtual data system for representing, querying, and automating data derivation. In Proceedings of the 14th Conference on Scientific and Statistical Database
19. Deelman, D., Blythe, J., Gil, Y., Kesselman, C. (2002) Pegasus: planning for execution in Grids. , GriPhyN technical report 2002-20..
20. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C. (2001) Grid Information Services for distributed resource sharing, presented at 10th IEEE International Symposium on High Performance Distributed Computing.
21. Chervenak, A., Deelman, E., et al. (2002) Giggle: a framework for constructing scalable replica location services, in Proceedings of Supercomputing 2002 (SC2002).
22. The Virtual Data Toolkit, http://vdt.cs.wisc.edu
23. Grid Cataloging System, http://www.ivdgl.org/gridcat
24. Newman, H.B., Legrand, I C., Galvez, P., VoicuR., Cirstoiu, C. (2003). Monalisa: a distributed monitoring service architecture, in Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, Mar. 24-28.
25. Tuecke, S. (2000). Grid Security Infrastructure (GSI) roadmap. Internet Draft, October, http://www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.pdf.
26. 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
27. Maltsev, N., Glass, E. Sulakhe, D., Rodriguez, A., Syed, M. H. Bompada, T, Zhang, Y. D'Souza, M. (2006). PUMA2 -- Grid-based high-throughput analysis of genomes and metabolic pathways. Nucleic Acids Res. Jan 1; 34(Database issue):D369-372.
28. Pathos System, http://compbio.mcs.anl.gov/pathos
29. TarGet Environment, http://compbio.mcs.anl.gov/target
30. Chisel, http://compbio.mcs.anl.gov/CHISEL
31. Overbeek, R., Disz, T., Stevens, R. (2004). The SEED: a peer-to-peer environment for genome annotation, Comm. ACM, 47(11)46-51.
32. Midwest Center for Structural Genomics (MCSG) http://www.mcsg.anl.gov
33. Great Lakes Regional Center of Excellence for Biodefense & Emerging Infectious Diseases Research, http://www.glrce.org.
34. Maltsev, N., Bompada, T., Gopolan, B., Li, S., Zhang, W., Detter, J.C., Richardson, P., Romine, M. Brockman, F. (2005). Metagenome analysis of contaminated sediments at the DOE Hanford site. Internet Draft, January 5, http://doegenomestolife.org/pubs/2005abstracts/addendum.pdf
35. Fredrickson, J.,Romine, M., Giometti, C.S., Kolker, E., Nelson, K.N., Tiedje, J. M., Zhou, J. (2005). The *Shewanella* Federation: functional genomic investigations of dissimilatory metal-reducing *Shewanella*. Internet Draft, http://www.doegenomestolife.org/pubs/2005abstracts/shewanella.pdf
36. Ed Seidel, Gabrielle Allen, Andre Merzky and Jarek Nabrzyski, GridLab--a grid application toolkit and testbed, Future Generation Computer Systems, Volume 18, Issue 8, October 2002, Pages 1143-1153.
37. Herrera, J.; Huedo, E.; Montero, R.S.; Llorente, I.M. Execution of typical scientific applications on Globus-based grids. Parallel and Distributed Computing, 2004. Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, 2004. Third International Workshop on, Vol., Iss., 5-7 July 2004, Pages: 177- 183
38. Venugopal, S., Buyya, R., and Winton, L. 2004. A grid service broker for scheduling distributed data-oriented applications on global grids. In *Proceedings of the 2nd Workshop on Middleware For Grid Computing* (Toronto, Ontario, Canada, October 18 - 22, 2004). MGC '04, vol. 76. ACM Press, New York, NY, 75-80