BioGrid Application Toolkit: a Grid-based Problem Solving Environment Tool for Biomedical Data Analysis

R. Nóbrega, J. Barbosa*, A.P. Monteiro

Universidade do Porto, Faculdade de Engenharia,
Departamento de Engenharia Electrotécnica e de Computadores
Instituto de Engenharia Biomédica (INEB), Lab. Sinal e Imagem
Rua Dr. Roberto Frias, 4200-465 Porto (Portugal)
e-mail: {nnobrega, jbarbosa, apm}@fe.up.pt

Abstract. In this paper, we describe a Problem Solving Environment (PSE) tool called BioGrid Application Toolkit and its architecture over a grid environment. BioGrid Application Toolkit is a general purpose grid access tool that provides a parallel and distributed programming environment; it provides an efficient web-based user interface that allows users to develop, run and visualize parallel/distributed applications running on heterogeneous computing resources connected by networks. BioGrid Application Toolkit is easy to use and has two execution models: streaming and data parallel mode. The system transparently schedules the submitted jobs and gives user feedback on the job status. The relevant components for the success of this parallel processing system will focus in how effective the scheduler and interface is. We also discuss some of the reasons for our design, present the initial results and discuss issues that we consider relevant for future research.

Keywords: grid computing, problem solving environment, heterogeneous clusters

1 Introduction

The aim of grid computing is to provide an environment which allows easy, ubiquitous access to geographically distributed, heterogeneous computing resources. As the Grid provides integrated infrastructure for solving problems, PSEs (Problem Solving Environments) have been developed to improve the collaboration among Grid services and reduce significantly the time and effort required to develop, run, and experiment with large scale Grid applications.

Biomedical data analysis, or Biomedical Engineering, is a field that includes areas of research such as bioinformatics, biomechanics, biosignal processing, biotechnology, computational biology, medical imaging, among others [6]. As this

* This work has been partially funded by the Department of Electrical and Computer Engineering of the Faculty of Engineering of the University of Porto (FEUP), under contract DEEC-ID/X/2004

field evolve, the size of biological data to be managed and analyzed increases, requiring high-throughput computing. Grid computation matches well with this need but it has been applied only to several computational bioinformatic tools. In addition, the integrated solutions do not provide a high-level user interface in order to be used by non-expert computer users. One of the contributions of the work presented here is a high-level user interface that provides a problem solving environment (PSE) for biomedical data analysis.

Problem-solving environments (PSEs) [8] are problem-oriented computing environments that support the entire assortment of scientific computational problem-solving activities ranging from problem formulation, algorithm selection, execution simulation and solution visualization. Thus, this PSE tool can be viewed as the environment through which the users can exploit Grid resources. This tool needs to be transparent in order to minimize the impact of the inherent complexity on users. Transparency means that the distributed system hides its distributed nature from the users, appearing and functioning as a normal centralized system. The main advantage regarding to other available systems, consists in using heterogeneous clusters - nodes might have different processing capabilities.

- **Cluster** A cluster computer is a group of loosely coupled computers that work together closely so that in many respects it can be viewed as though it were a single computer. Clusters are commonly (but not always) connected through fast local area networks.
- **Grid** A grid computer is a collection of geographically distributed, heterogeneous resources with software running on them to facilitate communication between them, resource discovery, scheduling, load balancing, data delivery, quality of service, authentication, delegation and related issues.

BioGrid Application Toolkit was created for accessing a grid network that solves a computational problem with parallel and distributed computing algorithms. The Grid resources are available programmatically to algorithms that initiate a large number of computationally intensive jobs in streaming or data parallel mode:

- **streaming**: when the user submits a set of dependent inputs that must be processed by the submitted order. However, different streams might be executed in parallel;
- **data parallel**: when the user submits independent inputs to be applied the same algorithm.

The user models the processing requests (jobs) with Directed Acyclic Graphs (DAGs). A DAG is a graph consisting of nodes (tasks) and vertices (data passed between operators). It is directed, meaning that data flows in one direction only. Furthermore, the graph is acyclic, prohibiting the presence of cycles within the structure. The user is able to decide when to run a project and monitor its execution as well as to create, save, open, close, delete, import and export a project.

2 Related Work

This section describes some common projects with our application.

GRAM - Basic Job Submission and Control Service [1] is a uniform service interface for remote job submission and control. It includes the ability to stage files (entire directory trees of files if necessary) into and out of a compute resource prior to and after job execution, remote I/O redirection, job status monitoring, and job signaling (stop, restart, kill, etc.). GRAM supports basic Grid security mechanisms and can map from Grid-wide identities to local accounts for accounting purposes. GRAM is not a scheduler and it is often used as a front-end to schedulers allowing Grid systems to submit jobs to the local schedulers.
Nimrod/G [1] is a "Grid aware" application - it is a tool for distributed parametric modeling. It exploits an understanding of its problem domain as well as the nature of the computational Grid to provide a high level interface to the user. Specifically, it provides transparent access to the computational resources, and implements user level scheduling.
Condor-G: A Computation Management Agent for Multi-Institutional Grids [7] is an extension to Grid via Globus Project [2]. It's a tool for managing a variety of parallel computations in Grid environments In brief, it combines the inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource management methods of Condor to allow the user to harness multi-domain resources as if they all belong to one personal domain. The user defines the tasks to be executed; Condor-G handles all aspects of discovering and acquiring appropriate resources, regardless of their location; initiating, monitoring, and managing execution on those resources; detecting and responding to failure; and notifying the user of termination.
NetSolve [10], is a client-agent-server system which provides a solution for access to remote resources and software. It includes an integration with a variety of client PSEs (Matlab, Mathematica, Octave). It uses server-proxies to leverage additional resource management and scheduling environments. NetSolves agent based scheduling eliminates the need for the user to know the location of resources capable of servicing the request and the fault tolerance mechanism allows selecting alternate resources without intervention from the user. The NetSolve server hides complexity from the user by invoking parallel programs or jobs on machines controlled by a batch queue.
DAGMan (Directed Acyclic Graph Manager) [3] is a meta-scheduler for Condor that allows users to specify a directed acyclic graph (DAG) of tasks to be executed with (potentially) complex relationships and dependencies. Condor finds machines for the execution of programs, but it does not schedule programs (jobs) based on dependencies. DAGMan submits jobs to Condor in an order

[1] http://www.globus.org/grid_software/computation/gram.php
[2] www.globus.org
[3] http://www.cs.wisc.edu/condor/dagman/

represented by a DAG and processes the results. An input file defined prior to submission describes the DAG, and a Condor submit description file for each program in the DAG is used by Condor. DAGMan is responsible for scheduling, recovery, and reporting for the set of programs submitted to Condor.
Proteus, a Grid based Problem Solving Environment for Bioinformatics [5] is a software architecture allowing to build and execute bioinformatics applications on Computational Grids by using an ontology-based methodology to describe bioinformatics applications as distributed workflows of software components. It can be used to assist users in:

– formulating problems, allowing to compare different available applications (and choosing among them) to solve a given problem, or to define a new application as composition of available software components;
– running an application on the Grid, using the resources available in a given moment thus leveraging the Grid scheduling and load balancing services;
– viewing and analyzing results, by using high level graphic libraries, steering interfaces (that allow to interactively change the way a computation is conducted), and accessing the past history of executions, i.e. the past results, that form a knowledge base.

As a PSE tool, BioGrid Application Toolkit provides a complete solution for transparent access to remote resources and software. The system allows users to create new projects and submit them while others are still running, this is the processing and management components of the PSE are independent of the user interface. The main advantage of the BioGrid application toolkit is to consider heterogeneity at the cluster level, this is, the scheduler can optimize job execution for homogeneous as well as for heterogeneous clusters. In the current state of development the scheduler assigns a entire job to a single cluster, although it is a project aim to consider a multi-cluster assignment. The available algorithms in the system are from the medical imaging area, however it is intended to consider the areas of biosignal processing and bioinformatics in a near future [4].

3 The System Architecture

3.1 Logical Architecture

Figure 1 shows the logical architecture of the system. The layers identified are:

1. **User Services Layer (USL).** This is the layer that provides a way for users to interact with the application. The user interface was developed in Java in order to remain independent of platform and operating system.
2. **Business Logic Layer (BLL).** Regardless of whether a business process consists of a single step or an orchestrated workflow, this application requires components that implement business rules and perform business tasks.

[4] BioGrid is being developed mainly for biomedical engineering researchers (www.ibmc.up.pt/ lab_associado.php)

3. **Data Access Layer (DAL).** This is the layer that allows the application to access a data store at some point during a business process. It makes sense to abstract the logic necessary to access data in a separate layer of data access layer. Doing so centralizes data access functionality and makes it easier to configure and maintain.
4. **Databases.** The database contains all available functions and its parameters, data types, information about the users and their projects, the logical location (path) of inputs/outputs. Prior to execution the input data is uploaded to grid-level storage.
5. **Grid Services.** Provides transparent access to distributed processing. It provides a scheduler to optimize execution on heterogeneous clusters. In the current state of development, the scheduler assigns a user job (DAG) to a single cluster.
6. **Security.** The security layer is concerned with authentication, authorization, secure communication.
   - **Authentication** It is defined as secure identification, which basically means that we have a mechanism for securely identifying our users that is appropriate for the security requirements of the application. Authentication is implemented in the user services layer to provide authorization, auditing, and personalization capabilities. This involves requiring the user to enter credentials (user name and password) to prove his identity.
   - **Authorization** The authorization aspect of the security layer is concerned with identifying the permissible actions for each authenticated security principal. In simple terms, the authorization layer determines who can do what.
   - **Secure Communication** In addition to authenticating users and authorizing requests, you must ensure that communication between the tiers of the application are secure to avoid attacks in which data is sniffed or tampered with while it is being transmitted or is being stored in a queue. Secure communications involve securing data transfers between remote components and services. It is used the following options for secure communications:
     - Securing the whole channel: Secure Sockets Layer (SSL). This is the recommended option for HTTP channels, a widely accepted standard to open SSL ports on the firewalls. This option is recommended when exposing a service interface to the Web.
     - Securing the data: Implementation of SOAP encryption mechanisms. Encrypting a whole message makes the whole message unreadable if the network packets become compromised.
7. **Communication** The communication layer defines how the components in the application will communicate with each other. The communication layer covers issues such as communication format and protocol.
   - **Format** The communication between the BLL and the Grid is performed using sockets. The communication between the DAL and the database is performed with Java Database Connectivity (JDBC). JDBC technology

provides cross-DBMS connectivity to a wide range of SQL databases and access to other tabular data sources, such as spreadsheets or flat files.

- **Protocol** The DAL and BLL layers are implemented with web services. Communication to these layers is performed using SOAP which is used for maximum interoperability. SOAP is an XML-based messaging protocol that can be used to implement remote procedure calls and is supported over various transport protocols, including HTTP. The combination of HTTP and SOAP provides not only implementation independence but also platform and programming language independence. This is very important for deploying a general service in scientific computing environments where heterogeneous computing systems exist and preferred programming languages vary by scientific community.

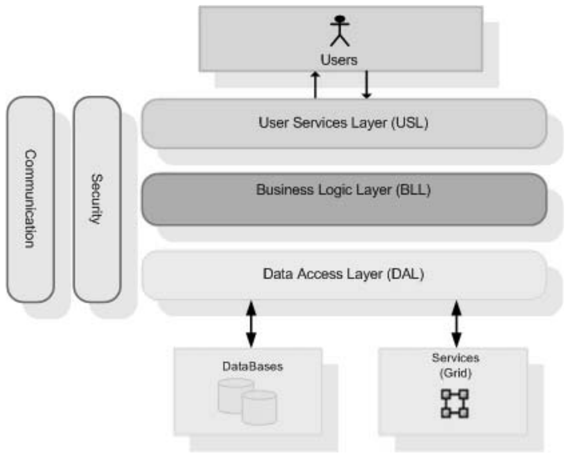


Fig. 1. The Logical Architecture

3.2 Physical Architecture

The hardware organization is shown in Figure 2.

Job Manager (JM)
BioGrid Application Toolkit provides the ability for users to transparently execute jobs on remote resources. This is implemented by a job manager service that encapsulates all of the aspects of executing a job, or a set of jobs, from start to finish. There are two levels of JMs:

- **Global Job Manager (GJM)** This is the Grid JM. It's responsible for orchestrating the services used to start a job or set of jobs. It accepts jobs,

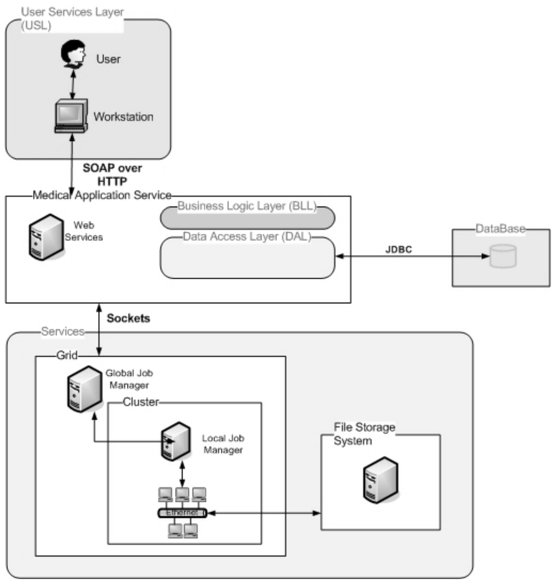


Fig. 2. The Physical Architecture

prioritizes them, and distributes them to different resources (clusters) for computation. In the actual development state the GJM assigns a entire job to one cluster.

– **Local Job Manager (LJM)** This is the JM of each cluster. It will schedule each job by selecting for each task the number of processors and the configuration that minimizes the computing time. It also manages "malleable tasks" [2] which allows more performance from the cluster in executing DAGs.

The Medical Application Service (MAS)
The Medical Application Service (MAS) contains the business and data components. These components offer interoperability, platform and programming language independence and transparency to the users in the sense that they are not aware about the Grid and database existences. MAS is implemented with JAX-RPC Web Services [5], which are java-based webservices. These web services are executed by Apache Tomcat, an application container. This service is used to exchange information among other layers in order to increase the interoperability among service components. MAS uses multithreading which allows two or more projects to be performed in parallel within the same application. The result is that MAS can continue to serve the user interface while processes projects to the Grid and MAS can receive requests from different users at the same time.

[5] http://java.sun.com/webservices/jaxrpc

The File Storage System
The physical location of the user inputs and outputs. Users are not aware of where these resources are physically located (location transparency). The input data referred in the user DAGs is, prior to execution and if not already available, uploaded to the File Storage System, which is shared by the participating clusters.

4 Implementation Case

4.1 Job Workflow

Figure 3 shows the job business process, from creation to completion. The entire process uses only one cluster.

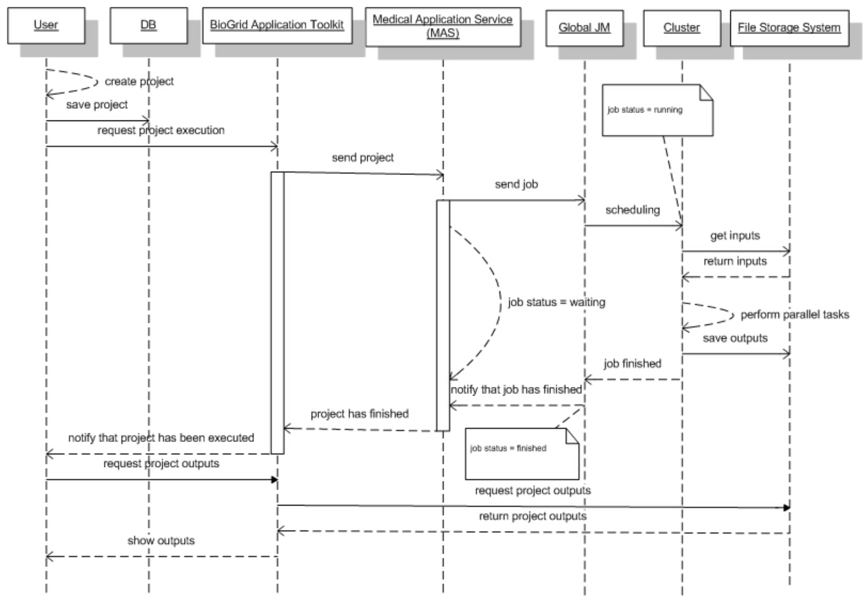


Fig. 3. The Job Workflow sequence diagram

1. The user creates a project, saves it in the database and requests its execution;
2. BioGrid Application Toolkit sends the project to the MAS;
3. MAS sends the project to the Grid Global JM (job status = waiting);
4. The Global JM performs cluster scheduling (job status = running);
5. The cluster gets the inputs from the File Storage System, performs tasks and saves the outputs in the File Storage System;
6. The Global JM notifies MAS that the project has been executed (job status = finished);

7. The MAS notifies BioGrid Application Toolkit about the project status;
8. If the project has been executed, the user visualizes its outputs.

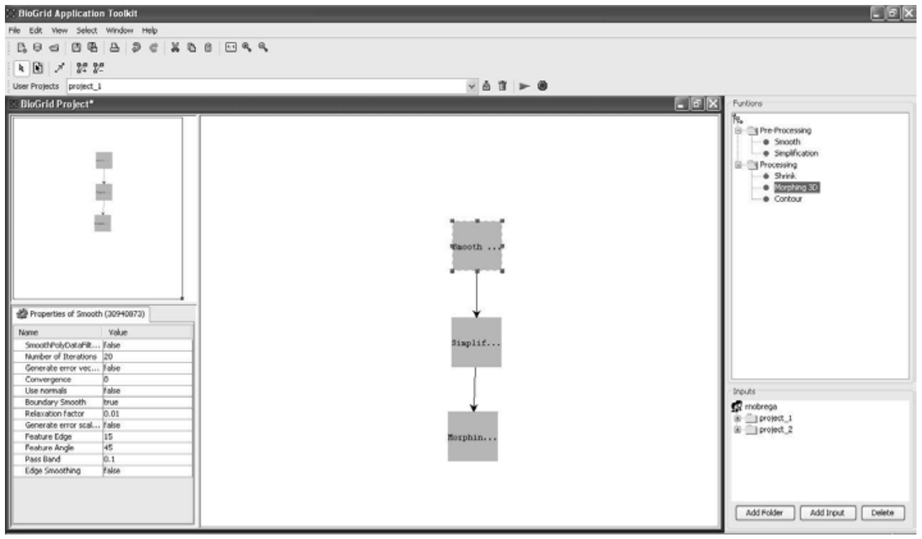
Job States
A job has 3 states:

– **running** When the job is currently in execution;
– **finished** When a job has finished its execution;
– **waiting** When the job is waiting to be schedule. A job may be in this state if it needs to access a temporary unavailable resource.

4.2 User Interface

Figure 4 shows a screenshot of the application. The user interface allows the user to treat the Grid as an entirely local resource, allowing the user to perform the following job management operations:

– Submit jobs, indicating the function name, input/output files and parameters;
– Query a jobs status or cancel the job;
– Be informed of job termination or problems;
– Obtain access to logs, providing a complete history of the job's execution.



Properties of Smooth (30940873)

Name	Value
SmoothPolyDataFilt...	false
Number of Iterations	20
Generate error vec...	false
Convergence	0
Use normals	false
Boundary Smooth	true
Relaxation factor	0.01
Generate error scal...	false
Feature Edge	15
Feature Angle	45
Pass Band	0.1
Edge Smoothing	false

Fig. 4. The BioGrid Application Toolkit

DAG creation
Figure 5 shows the DAG drawing process. A user wishing to execute a job must first create a project to provide information about it, where each DAG node represents a function from the Functions Panel. This information includes the:

- dependencies between nodes;
- parameters for each node;
- source of standard input for nodes.

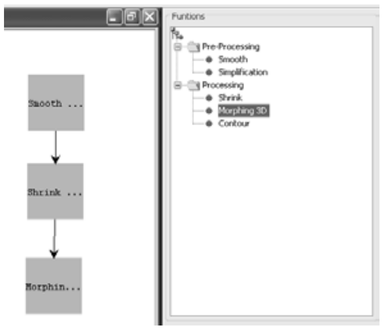
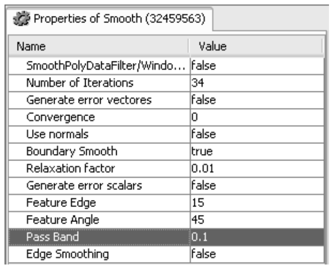


Fig. 5. The DAG drawing process

The DAG Properties Panel
This panel (Figure 6) contains the parameters for each function (node). Each parameter is editable.



Properties of Smooth (32459563)

Name	Value
SmoothPolyDataFilter/Windo...	false
Number of Iterations	34
Generate error vectores	false
Convergence	0
Use normals	false
Boundary Smooth	true
Relaxation factor	0.01
Generate error scalars	false
Feature Edge	15
Feature Angle	45
Pass Band	0.1
Edge Smoothing	false

Fig. 6. The DAG Properties Panel

The Functions and Inputs Panels
Figure 7 shows the Functions and Inputs Panels, that allow users to access files

(inputs) stored on a remote machine as if they were stored locally, through a file/folder hierarchy. The Functions Panel allows users to access database transparently to get functions. These functions will be used in the DAG creation. Graphically, a node represents a single function.

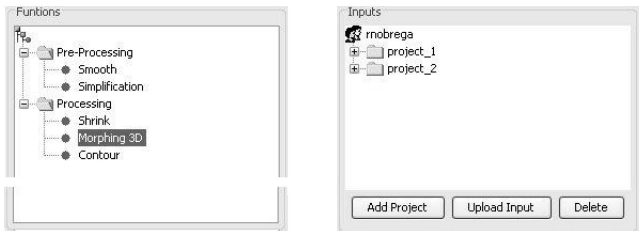


Fig. 7. The Functions and the Inputs Panels

5 Experimental Results

There are two types of results that matter to report: one is the user interface and the other is the execution efficiency. To obtain a more objective evaluation of the interface, it has to be tested according to the Interaction Design methodologies, which has not been done so far. Although the interface is very simple, showing the algorithms that are available, their parameters and the sequence of task execution (DAG representation), it is possible that such representation may not be so clear to a non-computer expert user. The execution efficiency depends on the scheduler (Job Manager) efficiency. The implemented scheduler is being developed for a few years ago, and in its first version it optimized the execution of a job (composed by several tasks) either in homogeneous and heterogeneous clusters. In [3] it was shown the scheduler efficiency for a set of linear algebra kernels, that are the base of many biomedical engineering algorithms. The second version [2] optimizes the execution of multiple jobs (user requests) on a cluster by allowing, the so called, parallel tasks. This is, each single task is executed by using several processors, as in the first version, but independent tasks are allowed to execute at the same time (in parallel) once there are available resources. It results two levels of parallelism: job and task parallelism [4]. The system efficiency is then demonstrated by the efficiency of the available scheduler. The algorithm chosen for the system evaluation is the object matching operation [9], widely used in biomedical imaging, that requires the computation of eigenvalues (tridiagonalization, Q matrix computation and QR iteration). It is considered problem sizes that generate input square matrices from 200^2 to 1600^2 elements.

Figure 8 shows the reply time for the object matching operation when submitted locally and remotely, for a 6 machine cluster with the peak performance

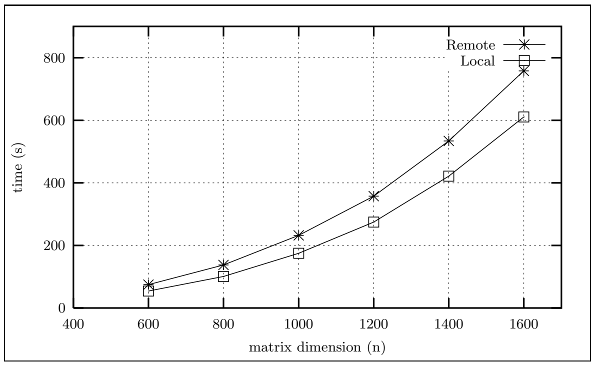


Fig. 8. Reply time for local and remote job submission

Matrix size (n)	600	800	1000	1200	1400	1600
Remote reply time (s)	74.8	137.7	232.3	357.2	533.9	757.7
Local reply time (s)	54.1	100.9	174.8	274.4	421.2	610.5
Absolute increase (s)	20.7	36.8	57.5	82.8	112.7	147.2
Relative increase (%)	38.3	36.5	32.9	30.2	26.8	24.1

Table 1. Reply time for local and remote job submission; absolute and relative reply time increase of remote over local execution

capacity of 970 $Mflops$. The local submission is made inside the local network where the cluster is installed and the remote submission is made from a broadband Internet provider access. Figure 8 shows that remote submission implies a higher reply time due to the communication that are made at a slower rate than in the local network, but the same execution efficiency can be guaranteed after the data is transmitted and the scheduler initiates the job execution. Although the delay imposed increases with the problem size, Table 1 shows that the relative reply time increase is less significant compared to the total reply time.

It is important to notice that if the user wants to use the same data in subsequent executions, the communications are reduced since only output data may need to be transmitted. Also the output data is transmitted if the user request that, because by default it is stored on the server user area.

6 Conclusions and Future Work

Here it was described the system architecture which has an interface over a Grid environment and provides a parallel and distributed programming environment; it provides an efficient web-based user interface that allows users to develop, run and visualize parallel/distributed applications running on heterogeneous computing resources connected by networks. The system have been tested with one cluster. It was used RPC-based Web Services (JAX-RPC) to exchange information among layers which increases the interoperability among service components. The adoption of a services-oriented architecture based on standard protocols will further reduce coupling, increase deployment options, and reduce costs. Resource/service discovery and scheduling in Grid environments are critical areas for further research. Future work includes the extension of the scheduler for a multicluster Grid and to develop an administrator version for BioGrid Application Toolkit. This will allow an easier management of the user profiles, functions and GUI configuration.

As a PSE-based tool, BioGrid Toolkit provides: an integrated computational environment for solving problems in a particular domain; it uses standards as part of the software infrastructure, which guarantees interoperability, easy development and maintenance, extensibility and platform independence; problem solving power without requiring user knowledge in the technologies used to effect the solution.

References

1. D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with nimrod/g: Killer application for the global grid? In *IPDPS*, pages 520–528, 2000.
2. J. Barbosa, C. Morais, R. Nobrega, and A.P. Monteiro. Static scheduling of dependent parallel tasks on heterogeneous clusters. In *HeteroPar'05: 4th Int. Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*. IEEE CS Press, 2005.
3. J. Barbosa, J. Tavares, and A.J. Padilha. Linear algebra algorithms in a heterogeneous cluster of personal computers. In *Proceedings of 9th Heterogeneous Computing Workshop*, pages 147–159. IEEE CS Press, 2000.
4. J. Blazewicz, M. Machowiak, J. Weglarz, M. Kovalyov, and D. Trystram. Scheduling malleable tasks on parallel processors to minimize the makespan. *Annals of Operations Research*, (129):65–80, 2004.
5. M. Cannataro, C. Comito, F. Lo Schiavo, and P. Veltri. Proteus, a grid based problem solving environment for bioinformatics: Architecture and experiments. *IEEE Computational Intelligence Bulletin*, 3(1):7–18, February 2004.
6. John Enderle, Susan Blanchard, and Joseph Bronzino. *Introduction to Biomedical Engineering*. Elsevier, Academic Press, second edition, 2005.
7. J. Frey, T. Tannenbaum, I. Foster, M. Livny, , and S. Tuecke. Condor-g: A computation management agent for multi-institutional grids. In *Tenth IEEE Symp. on High Performance Distributed Computing*, 2001.
8. J. R. Rice and R. F. Boisvert. From scientific software libraries to problem-solving environments. *IEEE Computational Science & Engineering*, pages 44–53, Fall 1996.
9. S. E. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:545–561, 1995.
10. K. Seymour, A. YarKhan, S. Agrawal, and J. Dongarra. Netsolve: Grid enabling scientific computing environments. In L. Grandinetti, editor, *Advances in Parallel Computing*, volume Grid Computing: The New Frontier of High Performance Computing of *14*, chapter 1. Elsevier, 2005.