The GRB Library: Grid Computing with Globus in C

Giovanni Aloisio, Massimo Cafaro, Euro Blasi,
Lucio De Paolis, and Italo Epicoco

ISUFI / High Performance Computing Centre
University of Lecce, Italy
{giovanni.aloisio,massimo.cafaro}@unile.it
{euro.blasi,lucio.depaolis,italo.epicoco}@unile.it

Abstract. In this paper we describe a library layered on top of basic Globus services. The library provides high level services, can be used to develop both web-based and desktop grid applications, it is relatively small and very easy to use. We show its usefulness in the context of a web-based Grid Resource Broker developed using the library as a building block, and in the context of a metacomputing experiment demonstrated at the SuperComputing 2000 conference.

1 Introduction

In the last few years, a number of interesting projects like Globus [1], Legion [2] and UNICORE [3] developed the software infrastructure needed for grid computing [4]. Several grid applications have been developed since then, using the tools and libraries available. The "first generation" grid applications were highly successful in demonstrating grid computing effectiveness, fostering subsequent research in the field, but the lack of consensus about the kind of services a grid should have been providing, and about what programming models should have been used, proved to be a large barrier that hindered grid applications development.

To reach a wider community of grid applications developers and users we need to make easier the transition from desktop computing to the grid. One way to attract the users is to provide them with a comfortable access to the grid: indeed, current trends include the development of computing portals [5], to allow trusted users a seamless access to the grid through the Web.

With regard to the applications developers, an increasing number of tools developers is now working toward high-level tools and libraries layered on existing grid services. That way application development costs decrease because enhanced services are available and can be readily incorporated into existing legacy codes or new applications.

In this paper we shall concentrate on the use of the Globus Toolkit as grid middleware, and describe a library for grid programming layered on top of Globus. The paper is organized as follows. Section 2 introduces the library design and section 3 its implementation. Section 4 presents the Grid Resource Broker, a web-based grid environment built using the GRB library. We recall related work in the area and conclude the paper in section 5.

2 Library Design

In this section we present the main ideas that guided us through the process of designing our library. First of all, we selected as grid middleware the Globus Toolkit, because it is already deployed at many US academic and supercomputing sites, and is rapidly becoming the “de facto” standard for grid computing at a growing numbers of sites all over the world. Moreover, the Globus Toolkit is released under a public license that allows also commercial use and redistribution of the source code.

Using Globus we can transparently benefit from uniform access to distributed resources controlled by different schedulers and from both white and yellow pages information services whose aim is information publishing and discovery. A security infrastructure based on Public Key technologies protects network transactions using X.509 version three digital certificates.

In the design phase, we were faced with the problem of selecting a number of basic services among those provided by Globus, to be enhanced in the library. We decided to concentrate the attention on the most used services, because past experiences with several groups of Globus users revealed that only a small number of core services are effectively used by a vast majority of users. These services include:

1. Remote job submission for interactive, batch and parameter sweep jobs;
2. Job status control;
3. Resource discovery and selection through queries to GRIS and GIIS servers;
4. Remote file management using GSI FTP servers (Globus Security Infrastructure FTP).

Another requirement was to keep small the number of functions composing our library, in order to reduce the time needed to learn how to use them. Moreover, we also planned to make the library suitable for the development of web-based and desktop grid applications. Finally, the primary requirement was to provide users with enhanced Globus services. Using our library, the users can write code to:

1. Submit interactive jobs on a remote machine staging both the executable and the input file(s).
2. Submit batch or parameter sweep jobs on a remote machine. Again, we provide support for automatic staging of executable and input file(s). Moreover, the output file(s) can also be transferred automatically upon job completion to another machine. Information related to the job is stored in a request file, to be used when the users want to enquiry about the job status.
3. Check a batch or parameter sweep job status. A Globus job can be either PENDING, ACTIVE, SUSPENDED, FAILED or DONE. If PENDING, the job is waiting to be executed sitting in a queue; during execution the job is ACTIVE and can

temporarily become SUSPENDED if managed by a preemptive scheduler. If something goes wrong the job is marked FAILED, while in case of normal termination its status becomes DONE. Moreover, we signal to the user when the job was started and how, including the command line and the arguments given. It's worth noting here that Globus provides only a url which uniquely identifies a job.
4. Query a GRIS server to determine the features of a specific computational resource. Each machine running Globus can be queried about its features because a small LDAP server called GRIS (Grid Resource Information Service) is up and listens for connections on the IANA registered port 2135. The GRIS stores static and dynamic information related to hardware and system software, and can be thought of as a white pages service.
5. Query a GIIS server to find computational resources matching specified criteria. An institution or organization may decide to setup a GIIS service (Grid Index Information Service). Like the GRIS, the GIIS is an LDAP server that collects information maintained in each GRIS server available in the institution or organization. Thus, we can exploit the GIIS server like a yellow pages service to allow users to find, if available, computing resources with specific features, e.g. amount of mem-memory, number of CPUs, connectivity, operating system, load etc. Although Globus provides a command line tool called *grid-info-search*, the tool itself can only deal with strings, meaning that users can not compare numerical quantities using standard relational operators, because in this case the lexicographical order will be used in place of the numerical one. Since numerical quantities are stored in GRIS and GIIS servers as strings, it is meaningless to search for machines with at least 256 Mb of memory using the Globus tool. Our functions instead do automatically proper conversions.

3 Implementation

The GRB library is made of a small number of functions; it is based on Globus version 1.1.3 or 1.1.4 and requires also the Expect library. The implementation of some of the functions is a bit tricky, due to our commitment to support web-based grid computing in the form of CGIs. Some of the functions in our library take as one of their parameters a pointer to an external, user defined function. This is an output function that the user can specialize to output data to the console, properly formatted as HTML for web based applications or even dedicated to insert text in graphical widgets.

We begin the discussion about the implementation considering first how to authenticate to the grid. Computational grids based on the Globus Toolkit require that users authenticate once to the grid (single sign on) using a X.509v3 digital certificate before doing anything else. However, the user's certificate is only used to generate a *proxy*, which acts on behalf of the user (implementing a restricted form of delegation) and lasts for a specified number of hours. The two functions *grb_generate_proxy* and *grb_destroy_proxy* are used to create a valid proxy to start a grid session and to remove it later. Since Globus does not provide an API for proxy management, we call the *grid-proxy-init* and *grid-proxy-destroy* tools. However, to allow job submission from the web, two issues need to be solved. We must create a suitable user environment, switching from the traditional web-server nobody user to the real Globus user and adding some required Globus environmental variables to the user's environment. As a matter of fact, simply switching from nobody to the Globus user ID using the unix setuid function does not replace the nobody user's environment with the Globus user's one. The function *grb_create_env* that does the job is thus required for web based grid applications.

The other issue is related to the use of some Globus tools like *grid-proxy-init* and *Globusrun* from the web. We can not simply fork/exec these Globus commands from a CGI, it will not work because of the way Globus uses ttys. Instead, we need to drive it from a pseudo tty. This is why we use the Expect library.

Some of the functions in our library exploits the information stored in a file called the user's profile. This file contains information about the computational resources on the grid that the user can access. For each machine the user supplies the hostname (required to contact the Globus resource manager and GSI-FTP server), the pathname to her shell (required to access the user's environment on the remote machine), and a float number representing the node cost per hour in her currency (can be used for job scheduling). We plan to add more information in the future to allow for more complex scheduling algorithms. A number of functions in the GRB library deals with user's profile management (e.g. for adding, removing of updating a grid resource dynamically).

We now come to job submission. The GRB library supports interactive, batch and parameter sweep jobs. We decided to use the *Globusrun* tool instead of the GRAM APIs for interactive job submission in the *grb_job_submit* function because this tool supports automatic redirection of job's output. For batch submission we used the GRAM APIs to write both a low level blocking (the function returns only when the job to be submitted is terminated) and non blocking version (the function submits the job and returns immediately) called respectively *grb_batch_wait* and *grb_batch_nowait*.
The function *grb_job_submit_batch* provides additional support for automatic staging of executable and input file(s) using internally *grb_gsiftp_copy*. We recall here the possibility to start even a graphical client by setting properly the *display_ip_addr* parameter, so that the user just need to authorize the remote host to redirect the display on her machine using the "xhost" command on Unix machines.

Finally, parameter sweep jobs can be submitted using *grb_job_submit_parameter*. A number of users need to run the same executable with different input to do parameter study. We assume that the user's input files are called *input-1*, *input-2*,...,*input-n*. Moreover, we assume that all of the input files are stored on the same machine and allow output files to be transferred on a machine possibly different from the machines hosting the executable and input files. GSI-FTP is used if needed for automatic staging of executable and input/output files.

Two functions are provided to check a batch or parameter sweep job status: *grb_batch_job_status* and *grb_parameter_job_status*. These functions read the information about submitted jobs that is stored in requests files at job submission time and contact remote Globus jobmanagers to inquiry about job status.

Information about grid resources can be obtained by querying Globus GRIS and GIIS servers. The functions *grb_search_gris* and *grb_search_giis* contact these servers using the LDAP protocol. Simple resource brokers can be built easily by searching Globus LDAP servers to acquire the features of grid resources available in the user's profile, comparing them to some user's specified criteria in order to dynamically select one or more computational resources suitable for a batch or parameter sweep job, and submitting to the resources with a best match. This is the approach we adopted for our web based Grid resource Broker.

4 The Grid Resource Broker

As an example of use of the GRB library, we developed a web-based Grid Resource Broker [6][7]. This is a computing portal that allows trusted users to create and handle computational grids on the fly exploiting a simple and friendly GUI. The Grid Resource Broker provides location-transparent secure access to Globus services. Users do not need to learn Globus commands, nor to rewrite their existing legacy codes.

The use of the library results in a dramatic decrease of the time needed to develop these functionalities; the main effort reduces to handling HTML forms, demanding to the library the issues related to grid programming.

Our portal architecture is a standard three-tier model. The first tier is a client browser that can securely communicate to a web server on the second tier over an HTTPS connection. The web server exploits Globus as grid middleware to make available to its clients a number of grid services on the third tier, the computational grid. The Globus toolkit provides the mechanisms for submitting jobs to remote resources independently of the resource Schedulers, querying for static and dynamic information about resources composing the computational grid using the LDAP API, and a secure PKI infrastructure that uses X.509v3 digital certificates.

There are no restrictions on what systems/sites could be served by GRB, because of the way user profiles are handled. As a matter of fact, GRB can be accessed regardless of system/geographical location and Grid Resources can be added/removed dynamically. In order to use GRB, a user must apply to the ISUFI/HPCC (University of Lecce) to get an account, to the Globus Certification Authority to get a certificate, and she must properly setup her Globus environment on the machine running the web server. We assume that Globus v1.1.3 or v1.1.4 and GSI-FTP is installed and listening on port 2811 on each one of the computing resources the user adds to her profile; moreover we assume that the GRIS server is up and running on the default port 2135 on each computing resource and that it can be queried starting from the distinguished name "o=Grid". The user's client browser must be configured to accept cookies.

To start using the GRB, a user authenticate herself to the system by means of her login name on the GRB web site and her PEM pass phrase (Privacy Enhanced Mail) that protects the globus X.509v3 certificate. The transaction exploits the HTTPS (SSL on top of HTTP) protocol to avoid transmitting the user's PEM pass phrase in clear over an insecure channel. Once authenticated, a user can start a GRB session that will last for at most twelve hours or until she invalidate her session using the logout tool.

A preliminary version of our library was also used in the metacomputing experiment (see Fig. 1) demonstrated at the Supercomputing 2000 conference held in Dallas. In particular, the functions related to resource discovery and selection in the context of the Egrid testbed [8] were used to allow Cactus [9] to query the testbed information service.

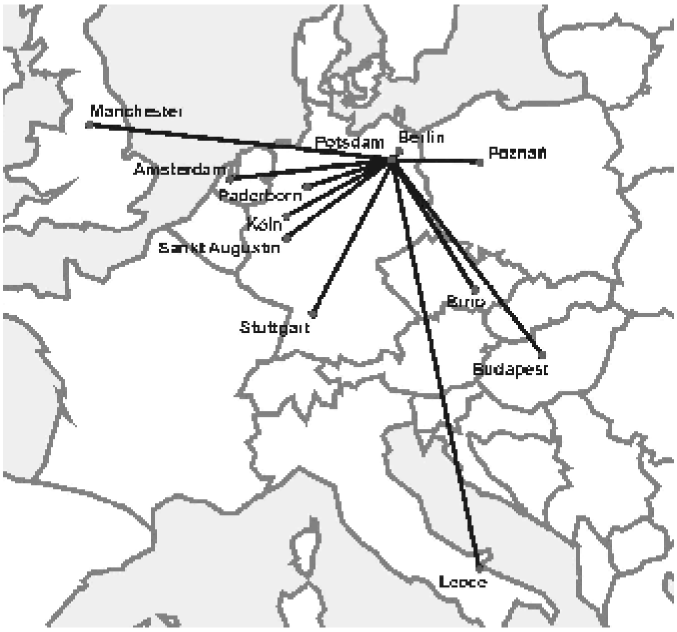

Fig. 1. The Egrid Testbed.

5 Related Work and Conclusions

The authors are not aware - as of this writing - of any projects aimed at releasing source code for enhanced Globus services. Anyway, the following two projects have provided useful software to help develop computing portals that provide access to the grid through basic Globus services:

1. SDSC GridPort Toolkit [10];
2. NLANR Grid Portal Development Kit [11].

The Grid Portal Toolkit (GridPort) provides information services that portals can access and incorporate; it leverages standard, portable technologies. GridPort makes use of advanced web, security and grid technologies such as Public Key Infrastracture and Globus to provide secure, interactive services.

Server-side Perl/CGI scripts build HTML pages, and a combination of HTML/JavaScript is used on the client side. Portals built on top of GridPort allow users job execution and data management through a comfortable web interface. Examples of portals built using GridPort are HotPage, LAPK, NBCR Heart and GAMESS.

The Grid Portal Development Kit provides access to Grid services exploiting Java Server Pages (JSP) and Java Beans. Java Server Pages invoke Bean methods to provide authentication services, management of user profiles, job submission, etc. The GPDK Java beans build on top of the Globus Java Commodity Grid (CoG) Toolkit [12].

GPDK Java beans present to web developers an high level interface to the CoG kit. Moreover, GPDK take advantage of the Myproxy package, so that users can gain access to remote resources from anywhere without requiring their certificate and private key to be located on the web browser machine. The Myproxy server is responsible for maintaing user's delegated credentials, proxies, that can be securely retrieved by a portal for later use.

A library layered on top of basic Globus services was presented in the paper. The library was designed to provide enhanced grid services and can be used to develop both web-based and desktop grid applications. Moreover, it is relatively small and very easy to use. We showed its usefulness in the context of a web-based Grid Resource Broker developed using the library as a building block, and in the context of a metacomputing experiment demonstrated at the SuperComputing 2000 conference.

References

1. I.Foster and K.Kesselman, "GLOBUS: a Metacomputing Infrastructure Toolkit", Int. J. Supercomputing Applications (1997), 115-28
2. A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver and P.F. Reynolds Jr., "The Legion Vision of a Worldwide Virtual Computer", CACM 40 (1997)
3. J. Almond, D.Snelling, "UNICORE: uniform access to supercomputing as an element of electronic commerce", FGCS Volume 15 (1999), Numbers 5-6, October 1999, pp. 539-548
4. I.Foster and C.Kesselman (eds.), The Grid: Blueprint for a new Computing Infrastructure, (Morgan Kaufmann Publishers, 1998) ISBN 1-55860-475-8
5. http://www.computingportals.org
6. http//sara.unile.it/grb
7. G.Aloisio, E.Blasi, M.Cafaro, I.Epicoco, The Grid Resource Broker, a ubiquitous grid computing framework, submitted to Journal of Scientific Programming
8. G.Aloisio, M.Cafaro, E.Seidel, J.Nabrzyski, A.Reinefeld, T.Kielmann, P.Kacsuk et al., Early experiences with the Egrid testbed, to appear in Proceedings of CCGRID2001, Brisbane, Australia
9. G.Allen,W.Benger,T.Goodale,H.Hege, G.Lanfermann, A.Merzky, T.Radke, and E.Seidel, The CactusCode: A Problem Solving Environment for the Grid. In Proc.High Performance Distributed Computing (HPDC-2000), pp 253–260, IEEE Computer Society, 2000
10. SDSC GridPort Toolkit, https://gridport.npaci.edu
11. NLANR Grid Portal Development Kit, http://dast.nlanr.net/Features/GridPortal
12. G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids", accepted to the ACM 2000 Java Grande Conference, 2000