GRIDBLOCKS – WEB PORTAL AND CLIENT FOR DISTRIBUTED COMPUTING

Juho Karppinen, Marko Niinimäki, John White
Helsinki Institute of Physics
P.O. Box 64, 00014 University of Helsinki, Finland

Tapio Niemi
University of Tampere
Kalevantie 4, 33014 Finland

Keywords: Grid, distributed computing

Abstract: GridBlocks is an architecture and a reference implementation of a distributed computing platform for heterogenous computer clusters. It can be used when there is a need to analyze vast amounts of data that is stored in a distributed fashion. The computing and storage resources can be used both by a Web interface or by a standalone Java client. Grid Security Infrastructure (GSI) is used for secure authentication and communication.

1 INTRODUCTION

Foster and Kesselman describe the Grid concept as follows: "The Grid is a software infrastructure that enables flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources." (Foster et al., 2001)

The driving force behind the Grid is to use and share computing resources and to make it as ubiquitous as the World Wide Web is currently. However, a problem with current Grid implementations (e.g. Globus, (Foster and Kesselman, 1997)) is that they are difficult to install and maintain. This is mostly due to their large scope and generality, which implies complexity. Therefore, using the Grid is still far away from "surfing" on the Web.

Our partial solution is the GridBlocks platform, which consists of a web server and a Java client. It is very easy to install and use because of its architecture that conforms to Java standards. Java also ensures that the same program code can be executed in different computer platforms. The web server currently used by GridBlocks is the Tomcat (The Apache Project, 2002) servlet container. The installation of GridBlocks does not need any special privileges and an ordinary user can deploy it and the Tomcat server in their own home directory. The GridBlocks server needs to access only one TCP port that can be configured by the user to reduce security problems.

The most important advantages of GridBlocks are the following:

- Being purely Java-based, it supports truly heterogeneous environments (Solaris, Linux, Windows, ...).
- Using GSI (Foster and Kesselman, 1997) in user authentication, it increases security and enables intercommunication with other certificate based systems.
- Gains computing power by distributing the jobs to multiple computers.
- Saves bandwidth because of no need to transfer large amounts of data.
- Enables easy and user friendly access to Grid services.
- Utilizes Web interface and therefore can be used without any installations, too.

In this paper, we discuss the overall architecture in Section 2. In Section 3 we present the functionality of the server and in Section 4 the functionality of the Java agent. Section 5 discusses applications, and Section 6 a summary and items for future research.

2 GRIDBLOCKS ARCHITECTURE

GridBlocks is a centralized platform that consists of server(s), client(s) and analysis (computing) nodes. The basic architecture of the system is shown in Figure 1. The GridBlocks server serves the requests of

Karppinen J., Niinimäki M., White J. and Niemi T. (2004).
GRIDBLOCKS – WEB PORTAL AND CLIENT FOR DISTRIBUTED COMPUTING.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 36-40
DOI: 10.5220/0002654600360040
Copyright © SciTePress

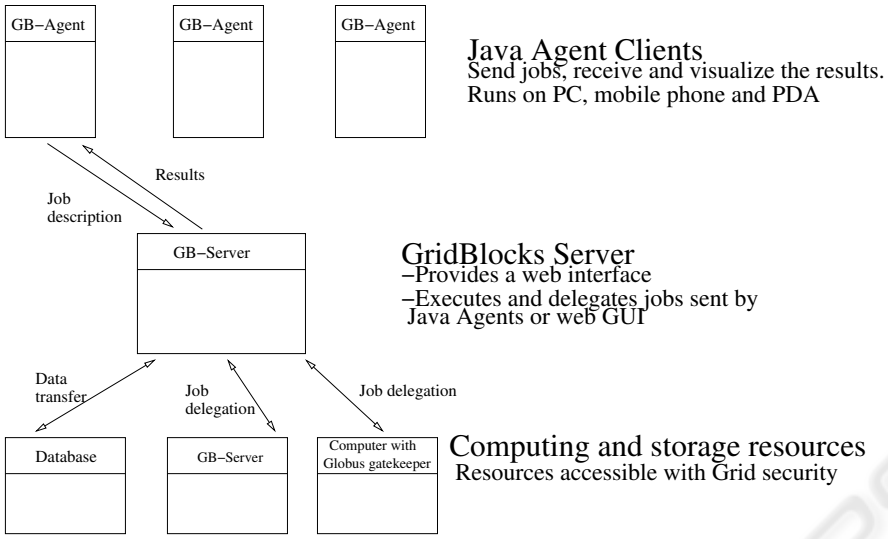

Figure 1: The overall system architecture.

clients and dispatches jobs to the correct computing node using the information stored inside the lookup server. The analysis servers are just the computing power and data storage for the jobs and they can be easily multiplied. The client computer is an access point to the GridBlocks network and it does not perform any computation-intensive tasks. This architecture allows the GridBlocks access point to be a web client or other thin clients such as a PDA or a mobile phone.

3 THE GRIDBLOCKS SERVER

The GridBlocks server fulfills following tasks:

1. It provides the GridBlocks users an access point to Grid;
2. It hosts the dispatcher component of GridBlocks;
3. It acts as computing node for agents.

The Grid software infrastructure is a middleware package that, ideally, functions in a heterogeneous computing environment. There are multiple middleware packages, among them Globus (Foster and Kesselman, 1997), Legion (Grimshaw and Wulf, 1997) and GridEngine (Grid Engine, 2002). These packages provide services such as resource management, authentication and authorization of users, secure (encrypted) file transfers and remote program execution. In the Globus middleware package: computing resources are managed by the Globus Resource Allocation Manager (GRAM) and Monitoring and Discovery Service (MDS) (Foster and Kesselman, 1997); the authentication and authorization of users are based on X.509 certificates (ITU X.509, 2000) and public key infrastructure; and for file transfer and remote program execution the Globus package is Global Access to Secondary Storage (GASS) (Foster and Kesselman, 1997).

The Grid Portal Development Kit (Novotny, 2002) (GPDK) provides a web interface to Grid services. The GPDK combines proven web technologies and the Commodity Globus (Cog) (The Globus project, 2002) Java API. The GridBlocks website relies mainly on the GPDK API based on CoG, as follows:

- CoG and GPDK classes communicate with GRAM, MDS and GASS services and enable user authentication and authorization, job submission and file transfer;
- Java Servlet Pages (JSP) call the COG and GPDK classes, provide them with user input and show their output in HTML form;
- A Java Servlet Container, in this case a Tomcat (The Apache Project, 2002) web server, runs the JSPs.

A demonstration of the default user interface of the server is shown in Figure 2.

In addition to hosting the user interface, the server is an access point to agents that work on a Grid. The server runs a dispatcher by which the Java Agents can submit jobs to other computing nodes and recover the results. The computing nodes run the same software as the GridBlocks servers which means that multi-level distributed chains can be performed. The GridBlocks agent is explained in Section 4.

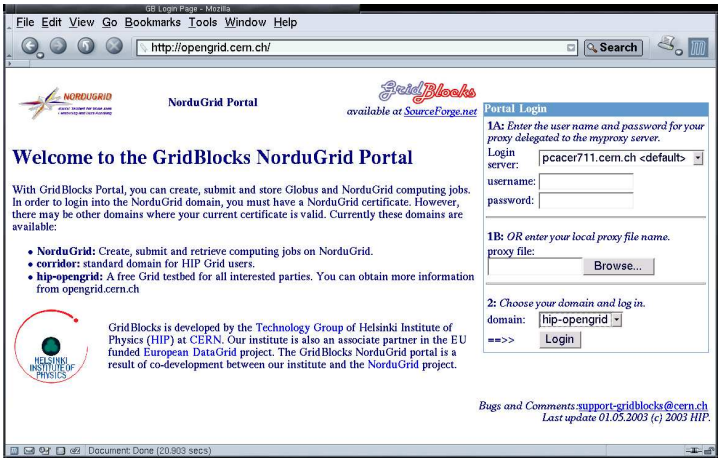

Figure 2: GridBlocks server (Opengrid.cern.ch demo site).

3.1 Authorization and authentication

A Grid user is identified by a certificate and private key, these may be combined to form a "proxy" certificate that is valid for a limited amount of time. Proxy certificates are widely used in the Grid context as they can be easily created and presented to Grid computing resources as an identity. The prerequisite for a proxy certificate is that the user has an identity certificate that has been signed by a certification authority. Moreover, he needs to have a proxy generation program that combines the user's identity certificate and password (pass phrase) into a proxy certificate.

In GPDK, two ways of authenticating a user to the portal have been implemented; logging in using an existing proxy certificate file and "proxy hosting", where a user's proxy certificate is retrieved from a server. For newcomers, the amount of work needed to install the programs and getting an identity certificate can be overwhelming. With GridBlocks, we use a different approach, as follows:

- A user can order a personal proxy file by email, simply by submitting his email address with a web form;
- The "certifier" program that processes the web form request has the capability to create new personal identification certificates and sign them (our certification authority ID is `/C=CH/O=HIP/OU=TECH/CN=112 Test CA`);
- The certifier creates and signs a certificate for a "user" `/C=CH/O=HIP/OU=TECH/CN=hillaX`, where 'X' is an iterator, but this certificate is not sent to the user;
- Instead a proxy file, valid for 7 days, is generated using the "user's" certificate. This proxy file is then sent to the user by email, as shown in Figure 3.

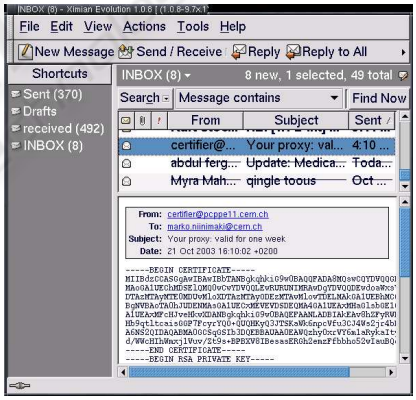

Figure 3: Obtaining a proxy certificate by mail.

Using this proxy file, an GridBlocks user can log into the Grid portal and access the basic services of some (limited) resources (computers). These computers are configured to accept Grid requests where the user certificate is `/C=CH/O=HIP/OU=TECH/CN=hillaX` and the certificate has been signed by an authority named `/C=CH/O=HIP/OU=TECH/CN=112 Test CA`.

An GridBlocks user can use the Globus MDS to view the attributes (CPU, RAM disk space etc.) of GridBlocks computers. They can also use the Globus GRAM and GASS functions to transfer files between, and run compiled jobs on, GridBlocks computers.

3.2 Shared resources

The GridBlocks website accesses a Globus MDS server that publishes the attributes of GridBlocks resources. This MDS server can also accept any publicly available service directory where any organization can promote their resources. If an GridBlocks user has a Globus installation on their computer, they can add their computing resources to GridBlocks by publishing attributes to the MDS server. This procedure is detailed on the GridBlocks website and, in short, includes:

- Configuring the user's MDS daemon (grid-info-resource-register) report the computer to the GridBlocksMDS server;
- Making the computer's Globus installation accept `/C=CH/O=HIP/OU=TECH/CN=112 Test CA` as a certificate authority;
- Modifying the local Globus authorization file (grid-mapfile) to map proxy certificates with subject /C=CH/O=HIP/OU=TECH/CN=hillaX to some user account.

4 THE GRIDBLOCKS AGENT

Here, we consider that the user has received or generated a personal proxy certificate, and starts their GridBlocks Agent (GB Agent) software, as in Figure 4. After that, the work flow proceeds as follows:

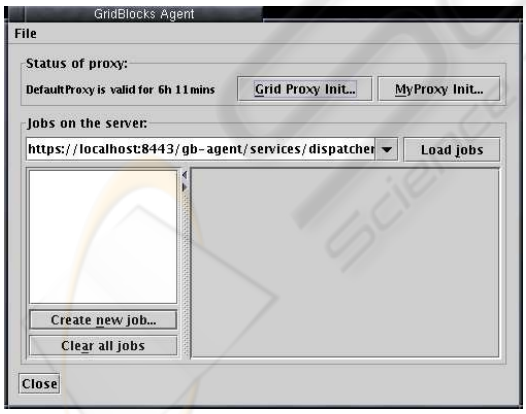

Figure 4: GridBlocks Agent.

1. The client sends a job request to the "primary" server with a JAR package of all analysis code. The client can disconnect right after this step, or stay connected and receive progress reports from the server.
2. The request can define which computing servers will be used. If this is not the case, primary server can contact a lookup server, locate the data, and dispatch the job to the target computers that perform actual computation.
3. The job is executed on the remote locations, and status messages are regularly sent to the primary server. When the analysis is completed, the results are sent back to the primary server where they are combined and stored.
4. The client can fetch the results from the primary server at any time.

The users code, the "agents", are divided into three parts. The first part is for actual execution which is done on remote servers. The second part is used for combining the results from multiple servers. This can be seen also as post-processing of results before they are sent back to the user. The last part of the agent is for presenting the results to the user in a visual format, if necessary. This means that results are not limited to certain formats and the visualization code can contain some intelligent features, like sending new jobs automatically based on the users input.

Simple Object Access Protocol (SOAP) (Simple Object Access Protocol, 2002) is used as a transport media for consistency reasons. The communication protocol does not use any specific standard, such as the Foundation for Intelligent Agents (FIPA, 2003) standards for communication. This fact has been recognized and the possibility of adhering to a such a standard in the future is being investigated. The user authentication and all communication between clients and servers are secured by the Grid Security Infrastructure (GSI). This is achieved by using Java CoG Kit (The Globus project, 2002) and European DataGrid Java Security components (Security Coordination Group, 2003). Thus, GB Agent is fully compatible with existing Globus security platforms.

5 APPLICATIONS

Originally, GridBlocks was designed for the analysis of high energy physics data, as in Figure 5 (Niinimaki et al., 2003). Recently the software has been generalised for generic data analysis and distributed computing applications. We have, for example, implemented the distributed aggregation calculation for OLAP (On-Line Analytical Processing) using GB Agent and Spitfire (a web front end for relational databases (Hoschek and McCance, 2001)). We have also tested the use of neural network-based distributed optimization applications.

Implementing new applications is very easy because of the mobile Java code. The user must only

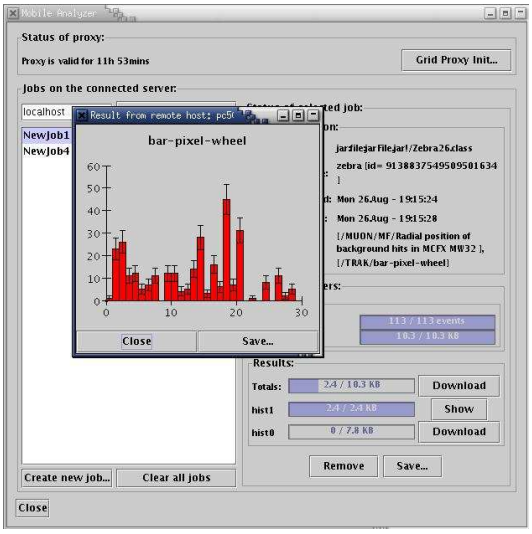

Figure 5: GridBlocks Agent visualising the results of physics simulations.

write the required functionalities inside the class implementing the remote job interface and send it within the job request. There is no need to update or install anything on the server side.

With only couple of lines of code, it is possible, for example, to use GB Agent as a remote shell console. The user's code simply executes given shell commands using Java's Runtime class and returns the textual output back to the user. This can be repeated automatically for multiple hosts and the job request can even give separate parameters for every host. Similarly, it is possible to implement distributed computing for any application.

6 CONCLUSIONS

The GridBlocks portal offers new Grid users an easy way to test Grid functions. The Grid functions are presented separately to encourage the GridBlocks user to see how they could be combined to form complete Grid applications. Service providers, interested in the Grid, can use GridBlocks to test and promote their computing resources by publishing to the GridBlocks MDS server. For academic organizations the GridBlocks is ideal for testing research software

GridBlocks is a new concept for distributed Grid computing based on the idea of mobile code. It has a simple and efficient Java implementation, which makes it easy both to install and use.

Both the client and the server are available at http://gridblocks.sourceforge.net.

REFERENCES

FIPA (2003). Foundation for intelligent physical agents. Available on: http://www.fipa.org.

Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2).

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3).

Grid Engine (2002). Welcome to grid engine. Available on: http://gridengine.sunsource.net/.

Grimshaw, A. and Wulf, W. (1997). The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1).

Hoschek, W. and McCance, G. (2001). Grid enabled relational database middleware. In *Global Grid Forum*.

ITU X.509 (2000). Information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks. Technical report, ITU.

Niinimaki, M., White, J., and Herrala, J. (2003). Executing and visualizing high energy physics simulations with grid technologies. In *proceedings of Second International Symposium on Parallel and Distributed Computing*, Ljubljana, Slovenia. IEEE.

Novotny, J. (2002). The grid portal development kit. *Concurrency and Computation: Practice and Experience*, 14(13-15).

Security Coordination Group (2003). Datagrid security design, deliverable 7.6. Technical report, European DataGrid Project. Available on https://edms.cern.ch/document/344562.

Simple Object Access Protocol (2002). SOAP - a XML based lightweight protocol for exchange of informations in a distributed environments. Available on: http://www.w3.org/TR/SOAP.

The Apache Project (2002). Apache tomcat. Available on: http://jakarta.apache.org/tomcat.

The Globus project (2002). Commodity Grid Kits. Available on: http://www-unix.globus.org/cog/.