Towards On-Demand Ubiquitous Metascheduling on Computational Grids

J. M. Alonso, V. Hernández, G. Moltó
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
{*jmalonso,vhernand,gmolto*}*@dsic.upv.es*

Abstract

Grid Computing Technologies are mature enough to be successfully applied to computationally intensive scientific applications. However, the current process of applying Grid Computing to them is still hard and difficult for the least experienced users. In this paper we describe the adaptations made to the GMarte metascheduling framework in order to provide an ubiquitous access to its functionality as an efficient resource broker for the execution of tasks on computational Grids. A metascheduling component accessible by only means of a Java-enabled web browser has been developed, requiring almost zero configuration by the client. This approach has enabled to produce a generic multi-platform metascheduler which can be automatically deployed in the clients interested in resource brokering on computational Grids based on the Globus Toolkit.

1 Introduction

Grid Computing technologies [7] have emerged as a solution for the computational problems of Virtual Organisations (VOs), enabling the collaborative usage of remote resources to satisfy the execution of computationally expensive tasks. Among all the available Grid middlewares, the Globus Toolkit [6, 5] is broadly accepted to be the current de facto standard for deploying computational Grids. However, the Globus Toolkit only provides the basic services and capabilities to support Grid infrastructures. Performing complete executions of scientific applications on Grid environments typically requires the usage of *metascheduling* [10] technologies, that provide all the functionality required for efficient remote task execution.

We envisage Grid metascheduling technologies as a set of interoperable components which abstract all the underlying complexity of the Grid middleware, which in turn provides the execution support. Only if we manage to get the Grid closer to the end user, these technologies will have a real impact in many scientific fields.

In this paper we propose an on-demand metascheduler that is accessible via the network by every user interested in performing task allocation functionality on a Grid deployment. Access is provided by only means of a web browser, thus requiring almost zero configuration by the client. This way, the user just focuses on defining the computational tasks to be executed and delegates to this component their efficient execution on the available Grid infrastructure. As opposed to traditional metaschedulers, which typically require an installation and configuration on the client machine, and are typically bounded to a given platform (usually Unix-based), we propose a multiplatform metascheduling component that is accessible from different platforms and operating systems. This approach enables to simplify the usage of computational Grids for job executions, even for Windows-based desktop PCs, thus paving the way for the widespread adoption of Grid technologies by the least experienced users.

The remainder of the paper is structured as follows. First, section 2 introduces the metascheduling framework and abstraction environment provided by GMarte, which is the starting point of this work. Then, section 3 details the architecture of the developed GMarte-based component. Next, section 4 focuses on the related work. Finally, section 5 summarises the paper pointing to future research lines.

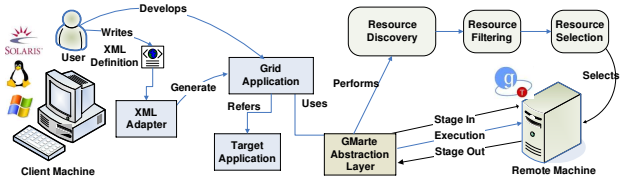

Figure 1. Overview of the principal phases involved in the GMarte framework

2 The GMarte Framework

GMarte [2, 4] is a software framework developed by our research group aimed at simplifying the process of executing batch parallel (MPI-based) applications on computational Grids based on the Globus Toolkit [6, 5]. It has been developed as a Java library which exposes a high level API (Application Programming Interface).

The GMarte framework is a layered software. First of all, a common interface is provided to access the information systems (via the Monitoring and Discovery Service) found on different versions of the Globus Toolkit (MDS2, MDS4) combined with several job managers such as Portable Batch System (PBS), Torque, Sun Grid Engine, etc. This enables to introduce high level objects that abstract the access to computational information from different resources. On top of it, GMarte introduces a new level that enables to perform the remote execution of a single task in a computational resource by combining the usage of the underlying Grid services provided by the Globus Toolkit such as GridFTP, for file transfer, or the Globus Resource Allocation Manager (GRAM), for job execution. Finally, above both layers, we abstract the process of multiple task allocation, enabling to perform resource discovery and fault-tolerant metascheduling.

Figure 1 summarises the principal phases that covers the GMarte framework. First of all, the user develops a small GMarte API-based Java application that provides the description of the tasks to be executed on the Grid as well as an enumeration of the computational resources to be employed. Alternatively, the user can also rely on the resource discovery functionality implemented. The metascheduling functionality requires several phases. First of all, the *Resource Discovery* stage obtains a set of candidate execution machines from either a GIIS (Grid Index Information Service) or a BDII (Berkeley Database Information Index). Both components aggregate resource information and can be queried to retrieve a list of machines that satisfy some features specified by the user. Once a list of potential execution machines have been retrieved, the *Resource Filtering* phase is in charge of discarding those resources that either are not accessible with the user credentials or do not satisfy the computational requirements of the task to be executed. When both previous phases finish, a set of computational resources on which to perform the execution of the tasks is available. In order to decide which resource is going to execute each task, a *Resource Selection* stage must take place for each one, which chooses the current best machine on which to execute a given task. In GMarte, a performance model is employed for resource selection that considers the task requirements, the dynamic state of resources and the network bandwidth. Finally, the *Remote Execution* phase takes place which requires proper data staging. A key point in GMarte is that it implements a multi-threaded metascheduler that enables to concurrently perform the different phases involved in remote task execution for the different tasks (resource selection, stage in, execution, stage out).

The usage of such a system enables unattended reliable execution of applications on computational Grids based on the Globus Toolkit versions 2.X (GT2) and 4.X (GT4).

3 Towards Ubiquitous On-Demand MetaScheduling

With the GMarte framework providing metascheduling functionality via an API, we decided to move further and create a component fully accessible by clients interested in task allocation on computational Grids. This section covers how GMarte has been adapted so that users no longer need to write a single line of Java code to access its functionality, but they specify their computational tasks and Grid resources via XML (eXtensible Markup Language) documents. Then, the Java Web Start technology that enables to publish applications on the web is briefly described. Finally, this section discusses the overall system developed to achieve on-demand metascheduling.

3.1 Developing XML interfaces to access GMarte

If we abstract the semantics of metascheduling, the main information needed for this process can be summarised in several items. On the one hand, we require the set of tasks, that is, a definition of all the jobs that are going to be executed on the Grid. This means specifying, for each task, the executable file, the dependent input files, the output files that should be retrieved upon execution and the computational requirements of

each task in terms of available RAM, minimum number of processors to use for parallel applications, etc. On the other hand, we require the set of resources, that is, an enumeration of the machines that will be employed for the execution of the tasks. As mentioned before, the user may specify an Index Service (GIIS or BDII) so that resources are automatically discovered. Finally, different parameters related to the metascheduler must also be configurated. For example, in our case, we may decide the number of threads employed to perform the file stage in of the tasks, or to disable the automatic reallocation of tasks in the case of execution failure.

Thanks to the adaptations performed in GMarte, the user just needs to describe, in a declarative manner, all the above information in three XML documents, one per item. The XML language was chosen because it is both easily readable by human beings and it can be parsed to access the coded information using software tools in many programming languages, including Java. Also, XML documents can be constrained to a W3C XML Schema, so that the syntax specified by the Schema is followed by the XML document. This way, it is said that the XML document is an instance of the Schema. Therefore, we have developed three different W3C XML Schemas that specify the syntax of the three XML documents that the user has to specify.

All the information specified in the XML documents will be automatically mapped to Java objects without any additional user action. Thus, the developed gateway enables to simplify the access to the GMarte functionality as it is no longer required to write a GMarte-based Java application.

3.2 Automatic Execution on the Client: Java Web Start

In order to achieve the feature of an ubiquitous access to the application, we have relied on the Java Web Start technology, which enables to deploy a full application accessible from the network. The principal advantage is that the user, with the only requirement of a web browser and the Java Runtime Environment, can access a Java Web Start deployed application. In addition, the user needs no special privileged account on the computer in order to run the network application.

The usage of this technology requires deploying an application on a Web server. This involves a set of JAR (Java ARchive) files and a JNLP (Java Network Launch Protocol) file which mainly describes and references the application files. The client, with a web browser, accesses the JNLP file and this causes the Java Web Start support in the client machine to start downloading all the application files that specifies the

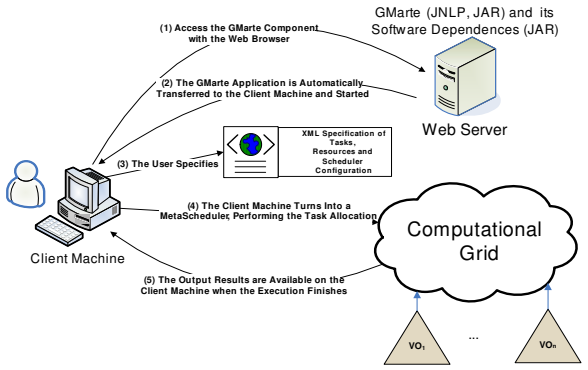

Figure 2. Scheme of the Grid Computing system developed

JNLP file. Once the application files are available in the user machine, the application is started outside the scope of the browser. Thus, it is considered that the application has been successfully deployed to the end user.

It is important to point out that Java Web Start technology has a caching component that prevents from downloading any of the application files that have not been modified since the last access. This allows saving bandwidth from the client-side, while ensuring that the latest version of the application will always be updated on the client machine. In addition, although the application runs outside the browser, this does not mean that it has full access to the user machine. In fact, the application runs on the *sandbox* provided by the Java Virtual Machine, thus being restricted by the Java policies. If an application deployed via Java Web Start wants to have full access to the client machine, then it has to be *signed* by the certificate of application developer. This procedure also involves signing all the JAR files that our application depends on. Furthermore, the client is always warned that the remote application demands full access to the local machine.

3.3 Overview of the System Developed

Figure 2 describes the main architecture found in the system developed. First of all, the user must have available in the client machine all the components required for security based on Globus GSI (Grid Security Infrastructure). This information typically involves:

1. An *User Certificate* signed by a Certificate Authority (CA), who ensures that the *Public Key* included in the certificate belongs to the user. This way, the user can be recognised as such by the

computational resources in the Grid that trust in the CA that signed the user's certificate.

2. A *Private Key*, only readable by the user, which enables to decipher messages, addressed to the user, which were ciphered with his/her Public Key. In GSI security, the Private Key is also protected by a password, to introduce an additional level of security.

3. CA certificates which signed the computational resources certificates.

The GSI-based security configuration on the client machine is a one-time process, provided that any certificate involved does not expire. In order to ease this procedure, we have also deployed, under Java Web Start, the configuration setup of the Java Commodity Grid Kit [12], a step-by-step GUI that enables to configure the client machine.

Once the GSI configuration has been performed, the client will always interact the computational resources via a user *proxy*, which implements a single sign-on strategy that prevents the user from typing the password each time the Private Key is accessed. The proxy is just a small file that holds the user credential for a limited amount of time.

After the security configuration procedure, the user accesses the Web site that enables to launch the GMarte component via JNLP. This process automatically retrieves the GMarte software itself and all its dependent JAR libraries. All this information represents a total of 8 MBytes to be downloaded to the client machine (841 KBytes for GMarte). Considering the cache that implements Java Web Start and the fact that the only component which is expected to change in each new release is GMarte, we can affirm that a lightweight access to the metascheduler is provided.

Before the application is started in the client machine, the user is notified that the software demands full access to his/her machine (in order to read certificates, perform GridFTP data transfers, create directories, etc.) and shows the certificate information of the developer that signed the application. Upon acceptance, the GUI component is launched to enable the user specify the XML documents that define their usage of the computational Grid. Finally, the metascheduling procedure is started, thus turning the client machine into a full metascheduler.

To ensure multiplatform, successful tests have been performed in several architectures (Intel Xeon, AMD Opteron and UltraSPARC-III) and operating systems (Linux Fedora Core, Windows XP and Solaris 10). It is important to point out that, although the metascheduling is performed in the client machine, a commodity PC (Pentium IV, 512 MBytes of RAM) is able to handle the whole process, while enabling the user to work as usual. The data transfers during the stage-in and stage-out phases are the ones which require the maximum workload (30% of CPU usage). However, minimal workload is required to query for the status of the computational tasks.

Currently, the usage of the GMarte metascheduler has been successfully applied to different areas such as cardiac electrical activity simulation [3] and dynamic structural analysis of buildings [1], where it provides an efficient solution for the execution of case studies composed of multiple computational tasks.

4 Related Work

The work presented in this paper involves the usage of an accessible on-demand self-contained software combined with the metascheduling capabilities over the abstraction layer provided by the GMarte framework.

There are many research projects that aim at solving the problem of metascheduling over the Globus Toolkit. First of all, the GridLab project[1], funded by the European union, aims at developing application tools and middleware for Grid environments. Within this project, the Grid Application Toolkit (GAT) [11] also provides an object-oriented approach trying to provide an abstraction layer. From the GridLab documentation, it appears that the resource management is only available for the Globus Toolkit 2.4, thus being restricted to work with the Pre-Web Services components of Globus. Also, the Grid Application Framework for Java (GAF4J) [9], is a simple framework of classes that abstracts the essentials of interfacing with a Grid infrastructure, assumed to be the Globus Toolkit 2.0, but offers no possibility to access the computational services provided by the latest version of the Globus Toolkit.

As another example, GridWay [8] is an open source meta-scheduler that performs job execution management, enabling unattended, reliable and efficient execution of different kind of jobs. Currently, GridWay enables to perform execution to both GT2 and GT4 resources, offering additional functionality such as task migration depending on the current performance of the jobs being executed. However, GridWay only runs on UNIX-like operating systems, which prevents all the Windows client machines from using metascheduling on computational Grids.

The main advantage of GMarte over other metaschedulers can be summarised in the following

[1] GridLab - http://www.gridlab.org

items: First of all, an ubiquitous access is provided so that it is accessible by only means of a web browser with Java support. Second, we achieve multiplatform availability to be automatically run on all the platforms on which Java support is provided, thus covering a wide range of machines. Third, we provide an XML interface to access the functionality of GMarte so that users are not required to develop an API-based application. Finally, we support both GT2 and GT4 computational resources.

5 Conclusions and Future Works

In this paper we have described the adaptations performed to the GMarte framework in order to create an on-demand metascheduler that is accessible via a Java-enabled web browser. The usage of Java Web Start has enabled to deploy a web application accessible by all the clients interested in turning their PCs into metaschedulers with almost zero configuration. This approach simplifies the usage of Grid Technologies, thus avoiding the installation and configuration of the metascheduler itself, so the user just concentrates on defining *what* should be executed and where (or just rely on the resource discovery functionality).

The future works involves moving the metascheduling functionality from the client machine to a specialised host which offers a metascheduling Grid Service to all the lightweight clients. This would reduce the amount of work performed by each client, which would turn into simple agents that would interact with the brokering Grid Service to access the metascheduling functionality. With this approach, we would be able to deploy the Grid Service as a gateway for an already established Grid, so the users would just specify the computational tasks. As an additional benefits, the clients could submit their executions, turn off their PCs and reconnect at a later stage to investigate the progress of the metascheduling procedure.

Acknowledgements

The authors wish to thank the financial support received from the Spanish Ministry of Science and Technology to develop the project GRID-IT (TIC2003-01318). This work has been partially supported by the Structural Funds of the European Regional Development Fund (ERDF).

References

[1] J. M. Alonso, V. Hernández, R. López, and G. Moltó. A Service Oriented system for On Demand Dynamic Structural Analysis over Computational Grids. In *VECPAR'06: Seventh International Meeting on High Performance Computing for Computational Science*, 2006. To appear.

[2] J. M. Alonso, V. Hernández, and G. Moltó. An Object-Oriented View of Grid Computing Technologies to Abstract Remote Task Execution. In *Proceedings of the Euromicro 2005 International Conference*, pages 235–242, 2005.

[3] J. M. Alonso, V. Hernández, and G. Moltó. Experiences on a Large Scale Grid Deployment with a Computationally Intensive Biomedical Application. In I. C. Society, editor, *18th IEEE International Symposium on Computer-Based Medical Systems*, pages 567–569, 2005.

[4] J. M. Alonso, V. Hernández, and G. Moltó. GMarte: Grid Middleware to Abstract Remote Task Execution. *Concurrency and Computation: Practice and Experience*, 2006. [early view] http://www3.interscience.wiley.com/cgi-bin/fulltext/112606256/PDFSTART.

[5] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In LNCS, editor, *IFIP International Conference on Network and Parallel Computing*, volume 3779, pages 2–13, 2005.

[6] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Intl. J. Supercomputer Applications*, 11(2):115–128, 1997.

[7] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[8] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Software Practice and Experience*, 34:631–651, 2004.

[9] A. Jhoney, M. Kuchhal, and Ventakrishnan. Grid Application Framework for Java. 2003.

[10] J. M. Schopf. Ten Actions When Superscheduling. SchedWD 8.5, Scheduling Working Group, 2001.

[11] E. Seidel, G. Allen, A. Merzky, and J. Nabrzyski. Gridlab: A Grid Application Toolkit and Testbed. *Future Generation Computer Systems*, 18:1143–1153, 2002.

[12] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Commodity Grid Kit. *Concurrency and Computation-Practice & Experience*, 13(8-9):645–662, July 2001.