Resource Selection and Application Execution in a Grid: A Migration Experience from GT2 to GT4

A. Clematis[1], A. Corana[2], D. D'Agostino[1,3], V. Gianuzzi[3], and A. Merlo[2,3]

[1] IMATI-CNR, Via De Marini 6, 16149 Genova, Italy
[2] IEIIT-CNR, Via De Marini 6, 16149 Genova, Italy
[3] DISI, Universitá Genova, Via Dodecaneso 35, 16146 Genova, Italy

Abstract. The latest Globus Toolkit 4 (GT4) version, fully OGSA compliant and almost completely based on Grid Services, is expected to become the reference version of the Globus Toolkit. Therefore, the necessity to migrate from previous versions, especially the widespread GT2, to GT4 is becoming a relevant issue for many Grid systems.

We present a migration experience from GT2 to GT4 of a simple tool for resource discovery and selection, and execution of parallel applications in a computational Grid environment.

The proposed tool provides a simple and intuitive user interface, a structured view of Grid resources, a simple mechanism for resource selection, and support for application staging and execution.

We discuss some relevant differences between GT2 and GT4, particularly for the Monitoring and Discovering subsystem. GT4 turns out to be more powerful and flexible.

Just because we address the implementation of basic functionalities, the proposed discussion may be of general interest.

1 Introduction

Grid platforms are increasingly used to share computation resources, data, instrumentation and knowledge. Grid systems and virtual organizations are no longer used only in scientific advanced projects, but are now of interest for a wider potential user's community, including business, industry and medium/small research groups. Such users need powerful but simple tools to set-up, administrate and exploit their own Grid platforms, often obtained by middleware customization. In the last few years standards for Grid systems evolved, and particularly there was a convergence between Grid and web services; indeed the Open Grid Services Architecture (OGSA) specifies the requirements for a Grid architecture based on web services [4]. This approach is general and powerful since it is based on existing standards, that are extended only when needed. In particular, Grid Services are improved web services, with some generalizations, e.g. the introduction of the state [7].

The Globus Toolkit is the most used middleware for Grid, since its introduction in the late '90. In particular GT2 [3] has been the most widespread version and still represents the basic middleware in many Grid platforms. Also

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4276, pp. 1132–1142, 2006.

the Globus Toolkit evolved following the OGSA specifications; however, the first service oriented GT version (GT3) experienced a diffusion lower than expected, also owing to the cost of migration of existing Grid applications.

The latest GT4 version (released in 2005) [8] is fully OGSA compliant and almost completely based on Grid Services and it is expected to became the reference version of the Globus Toolkit, also because the GT2 version is no longer supported. Therefore, the necessity to migrate from previous versions to GT4 is becoming a relevant issue for many Grid systems [5,9].

In the present work we are interested in computational Grids. In particular, we propose a tool that keeps in consideration the basic requirements for Grid resource monitoring and application deployment and provides a simple and intuitive user interface. The tool permits a structured view of Grid resources, the monitoring and selection of resources, and the staging and execution of parallel applications (both simple parameter sweep and message passing applications).

We present a migration experience of such tool from Globus Toolkit 2 to Globus Toolkit 4, pointing out the basic differences between the two versions of the structures involved in the Monitoring and Discovery Service (MDS).

2 A Simple Application Deployment Tool for a Grid

Since computational Grids comprise various machines, as a rule heterogeneous and belonging to different administrative domains, a simple tool able to classify nodes following their actual affinity degree, and to organize them into subclasses on the basis of suitable common criteria like cpu speed, RAM memory, network connection, and so on, is very useful.

The proposed tool consists of two parts: the first, for the Grid administrator, allows to set up a logical Grid structure (GSC, Grid Structure Creator); the second one (called GRSAE, Grid Resource Selection & Application Execution) permits the view of the structure created by GSC, the selection of a suitable set of resources and the staging and execution of the parallel application on the selected machines. The user interacts with the Grid through an ad-hoc GUI.

The tool has been designed with the following requirements: 1) it uses the mechanisms available in Globus Toolkit but it must be open to integration with other tools; 2) it must be modular; 3) it must be able to execute both simple parameter sweep as well message passing applications (eg. MPI applications).

2.1 A Structured View of Available Machines

In order to describe in a simple but effective way the complexity of the Grid, we choose to divide the computational resources potentially available in a Grid into three classes, namely the *single machine*, the *homogeneous cluster* and the *heterogeneous cluster*.

The single machine is the base unit, with a set of resources such as: node name, IP address, model and clock of the processor, number of available processors on the motherboard, total and free RAM, total and free disk space, type and release of O.S.

A homogeneous cluster is a set of machines having the same set of properties (i.e. a set of machines with the same hardware and software connected by a homogeneous network).

A heterogeneous cluster is a set of machines with different characteristics. It can be thought as a set of different machines that can be convenient to organize into a logical structure, for some kind of reason (e.g. all machines belong to the same department or to the same research group).

We suppose that a cluster (both homogeneous and heterogeneous) must have a distributed file system and is composed of nodes belonging to the same administrative domain. It needs three additional attributes: number of nodes, type of the distributed filesystem, and description of the network connection. In the current tool version, the homogenous cluster is seen as a unique computational resource, whereas in a heterogeneous cluster we can view each component node.

2.2 The Logical Grid Structure

The three models we have just described permit to create using GSC a logical Grid structure on which GRSAE works on. Such resulting topology has a tree structure [1] and is composed by a hierarchical disposition of the instances of classes we have presented.

The root of such tree structure is a single machine to which are logically connected by a path all the available machines. Each son of this root can be the root of a cluster (homogenous or heterogeneous) or a single machine. Each heterogeneous cluster is itself a tree with a root node and single machines as leaves.

The Grid middleware (GT in our case) is installed on each single machine, on the root of the homogeneous cluster, and on each node of heterogeneous clusters. We employ a test Grid composed of about ten machines (which can be grouped into heterogeneous clusters) plus a 16-nodes homogeneous cluster. All processors are Intel Pentium or Xeon.

2.3 Resource Selection, Node Allocation and Program Execution

The tool GRSAE helps the user in the execution of a parallel application. It is composed by three different modules, anyone with a well-defined role.

Connection Module. This module discovers and shows the entire structure of the Grid, the logical connection of the nodes and the state of the resources.

Selection Module. It interacts through a GUI with the user, and allows him to analyze the available resources and choose a subset of them (single machines or clusters) on which to run the parallel program. The user can filter resources on the basis of suitable criteria (e.g. processor type, frequency, RAM, disk space).

Execution Module. This module automatically creates the configuration files to launch the application on the selected machines of the Grid. The configuration files contain the executable path, the working directory for temporary files, the names of machines selected for the execution, the number of processes, the information for the staging of remote input and output files.

3 Deployment in Globus Toolkit 2.4

The tool, whose functionalities have been described in the previous section, has been originally deployed into Globus Toolkit 2.4 [2].

Such Grid middleware is composed by different independent subsystems that manage different aspects of the Grid. To set up our structured view of available resources, we are interested specifically at one of them, the MDS.

MDS is the information service that keeps track of all information about the Grid resources and their state. MDS has a hierarchical structure that consists of three main components. Every resource (e.g. CPU, RAM, storage) is controlled by a logical structure (GRIS, Grid Resource Information Service), that checks/measures the resource attributes through a suitable set of meters (IP, Information Provider). A set of resources (i.e. a set of GRIS) is then collected in another structure (GIIS, Grid Index Information Service): typically, there is a GIIS for every node of the Grid, collecting information about all resources of that machine. Various GIISs can be then organized hierarchically, creating groups of structures able to control many machines. A GIIS can be inquired about the set of resources it manages; information is supplied through a database using the standard LDAP (Lighweight Directory Access Protocol).

Deploying our structure on GT2 MDS is quite simple. In fact, we can map the root of the structure into a GIIS that manages a groups of other GIIS: one for each homogenous cluster, one for each heterogeneous cluster and one for each single machine, independent or contained in a heterogeneous cluster (Fig. 1).

The Grid structure is previously created by the Grid administrator using the tool GSC. Each cluster is registered in the root GIIS. Each machine is registered directly in the root GIIS, if it is a single machine, or in the GIIS of a cluster (if the machine belongs to a cluster).

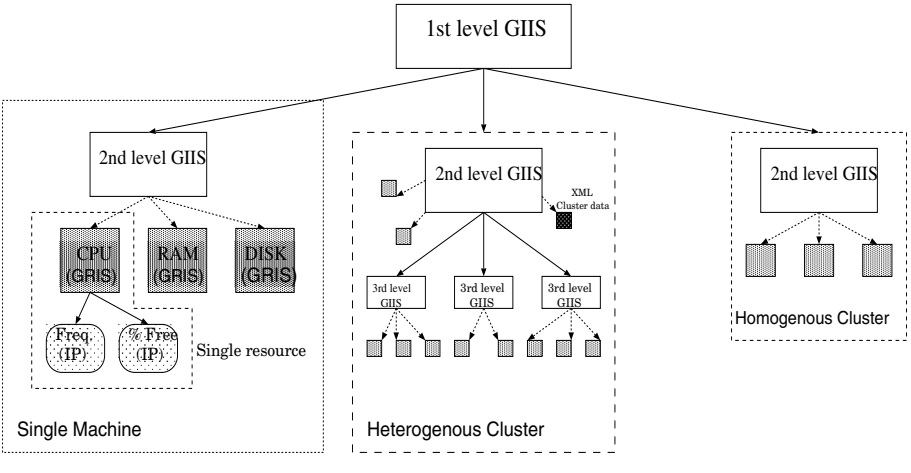

Fig. 1. The structure of the Grid under GT2; grey boxes denote GRIS

3.1 Modifying LDAP and MDS

In Globus Toolkit 2.4 data on resources of machines are kept on a distributed database (LDAP): a user that needs to know the state of a certain machine has to query the LDAP database to retrieve the attributes describing the resources of such machine. Unfortunately, in GT2 MDS and LDAP are able to keep track only of the resources on single machines: by default, there is no way to search and maintain collective information about a set of machines. For this reason, we modified the MDS and LDAP database in the following way:

- the new fields for the collective information needed for cluster are added to LDAP database;
- such information is kept in a XML file, one per cluster, on the cluster root; if characteristics of a cluster change we need only to modify the related XML file;
- in this way, a new IP for the cluster is created associated with the XML file: at every request of information about the cluster, it parses the XML file to find the relevant data that are stored in the new fields of LDAP database.

We denote by *'modified MDS'* the resulting MDS.

3.2 Interacting with the User

After the modification of MDS, the GRSAE tool is a Java application, implemented through Java CogKit, a Globus-oriented Java package, allowing modularity and easy interface with GT2. The modules of GT2 GRSAE behave as follows.

Connection Module. This module starts with enquiring the modified MDS to retrieve information about the structure; we query the root GIIS, we visit the tree and store all intermediate GIIS, until the leaves (single machines). Information about clusters is collected by a query of the cluster root GIIS and by reading the information stored in the corresponding XML file.

Selection Module. The structure and the resources are then shown to the user through the GUI (Fig. 2). From the whole set of resources found by the previous module, are returned the machines which fulfil the requisites given by the user. If the case, the user can further check the state of the resources, through a new query of the connection module.

Execution Module. After the choice has been made, GRSAE proceeds, through the Create RSL button, to create a Globus RSL executable file ($< file.rsl >$) with the parameters chosen by the user. Then the file is submitted to Globus Toolkit through the command '$globusrun < file.rsl >$', starting execution on the selected machines. Remote file transfer and management are performed by the module GASS (Global Access to Secondary Storage) of GT2.

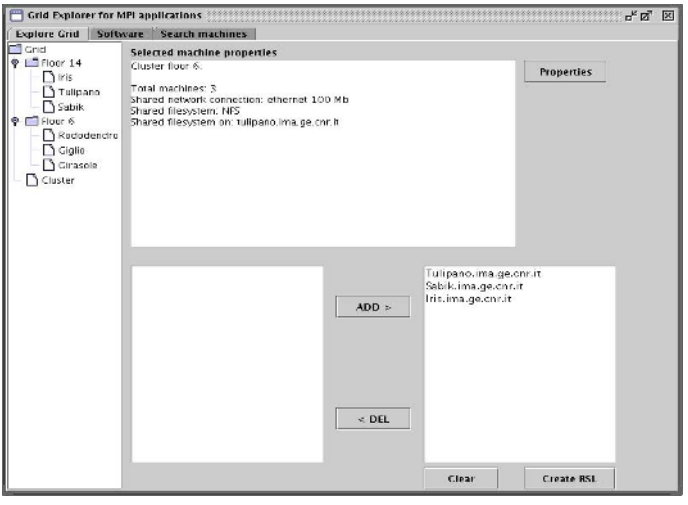

Fig. 2. An example of resource discovering and selection: on the left the whole tree, on the right the info about cluster Floor 6

4 Grid Information Service: Differences Between MDS2 and MDS4

Globus Toolkit evolution from the widely-adopted GT2 to the new-come GT4 passed through significant structural changes [10]. The new way of conceiving every resource on the Grid as an "improved" Web Service (GRID Service) and the need to access such resources in a common and standard way through different organizations required radical changes in the Globus architecture, specially in the Information Service [8]. We show in this section the main differences between MDS2 and MDS4, pointing out the advantages of the new implementation.

GRIS & GIIS vs WS Index Service. MDS2 manages information about one or more resources through a hierarchical system: every resource is controlled by a logical structure (GRIS) that uses a group of meters (IP) to measure different properties of the resource. Such properties are represented in LDAP-standard form. A set of GRISs can be organized into a GIIS that can be inquired about the set of resources it manages; moreover, different GIISs can be organized hierarchically.

In the MDS4 approach, every single resource (e.g. CPU, RAM, storage) is mapped into a Grid Service: in such a situation, accessing the state of a resource means querying an appropriate interface of the Grid Service that controls the state of the resource. In this case, the Grid Service replaces exactly the GRIS in MDS2 approach. Information is now inquired through standard Web Services queries and is returned in a XML document.

The GIIS structure has been replaced with Index Service (that is "de facto" another Grid Service) on which all the "resource" Grid Services can be registered on. By default, there is an Index Default Service on every host, that manages

all the resources of the host, but there is the possibility to create personalized Index Services.

Index Services can be organized in a hierarchical structure like GIIS (Fig. 3). The main difference between the two approaches is that MDS4 is more general and highly configurable. The Index System offers also a quicker way to configure homogeneous and heterogeneous clusters, allowing to create hierarchy of Indexes and to add information about the state of the cluster in a simpler way.

LDAP Database vs. XML Document. To fulfil OGSA requirements, GT4 replaced the LDAP database with a XML based description. XML is more general and doesn't depend on a particular database architecture. Moreover, it is platform-free and can be read by any proprietary system.

In XML approach information about resources is presented through a well-formed document based on standard schemas, creating a common way to describe the same kind of resource through different Grids and virtual organizations. Such schemas can easily be modified to permit the addition of new information about a resource: any organization that adopts ad-hoc schemas to describe its own resources, has only to publish and share them to allow Information Service from other organizations to retrieve information.

LDAP Architecture vs. Plugin Architecture. Another big difference is in the way the two systems practically collect information at the lowest level. In MDS2 information about a certain property of a resource is obtained through queries to a LDAP database, distributed among the GIIS. On the other hand, MDS4 doesn't support a particular system to retrieve such information but has a plug-in based architecture that allows the Grid administrator to use the preferred mechanism to collect information. Such approach stimulates the production of new information collectors.

LDAP Browser vs. WebMDS. In MDS2, information about a resources could be graphically shown to the user through one of available LDAP browsers, without any default. In order to achieve a higher standardization, GT4 adopted a single standard browser (WebMDS, based on Apache Tomcat) to access information in a XML document.

Pulling vs. Pushing. In MDS2 the only way to be aware of the state of a resource is to perform an explicit query to the LDAP database, and to keep up-to-date the state of a resource we have to repeatedly query the LDAP database, without any possibility to be informed automatically when a particular situation occurs. MDS4 breaks such limitation through a particular component, called Trigger Service, that allows to inform the user when a certain situation happens through a particular signal called Notification.

SAML vs. HTTPS. The security protocol that manages the authorizations and the certificates changed from SAML used in GT2 to HTTPS in GT4, because of the wider use of HTTPS on the Web, even if the way to authenticate and manage the security, based on proxies and certificates, remains quite the same in GT2 and GT4.

5 GSC and GRSAE Migration from GT2 to GT4

The differences between MDS2 and MDS4 we have shown before, make interesting the migration of GT2-based tools like GSC and GRSAE to GT4. It is clear that due to the significant architectural differences between MDS2 and MDS4, such operation could require a quite total rewriting of the application.

In particular, using GT4 we don't need to modify MDS to manage collective information about the cluster (FS Type, Connection Type, etc.), but we use an ad-hoc Grid Service on every node that is the root of a cluster.

The GT4 version of the tool is maintained in Java, adapting the GT2 one, and relying on the Java Core of the GT4 suite [11].

5.1 Organizing the Structure of the Grid

The creation of a Grid structure from a set of hosts and clusters can be performed on MDS4 through the Index Service. Every hosts with GT4 has a Default Index Service that keep track of the information about all the resources (as Grid Services) of the machine. Any of this Default Index Service can be placed in relation of hierarchical dependence with a second Default Index Service, simply modifying a XML configuration document (**hierarchy.xml**) on the machine that must have the lower level in the hierarchy.

For example, to make a machine A "father" of a machine B, we must only modify the hierarchy.xml file in B, inserting a tag labeled "upstream" with the address of the Default Index Service of A. After this modification has been done, the Default Index Service on machine A collects information about resources on B directly through Default Index Service of B.

In this way, the Grid Administrator can easily design a Grid structure (Fig. 3) using an ad hoc GSC tool for GT4. The hierarchy creation results simpler and more automatic than in GT2. Moreover, we can add in a simple way information about the Grid or the clusters (e.g. Total memory available on all the machines of a cluster) different from that offered by the Default Index Service of the single host. To obtain such result, GSC simply creates a new schema from the default ones, adding the new meters and mapping them to an appropriate collector of data.

5.2 Interacting with the User

With GT4 the three components of GRSAE perform the following tasks.

Connection Module. This module interacts only with the MDS4 components of GT4. It takes care of inquiry the Default Index Service of the host that is the root of the topology, in order to know the state of all the resources of the Grid. It receives a hierarchically-organized XML and parses it, taking out information about the Grid structure, and the state of every Grid Service on every nodes. Then, it starts a communication to the Selection module, passing to it the whole information organized in appropriate Java data structures.

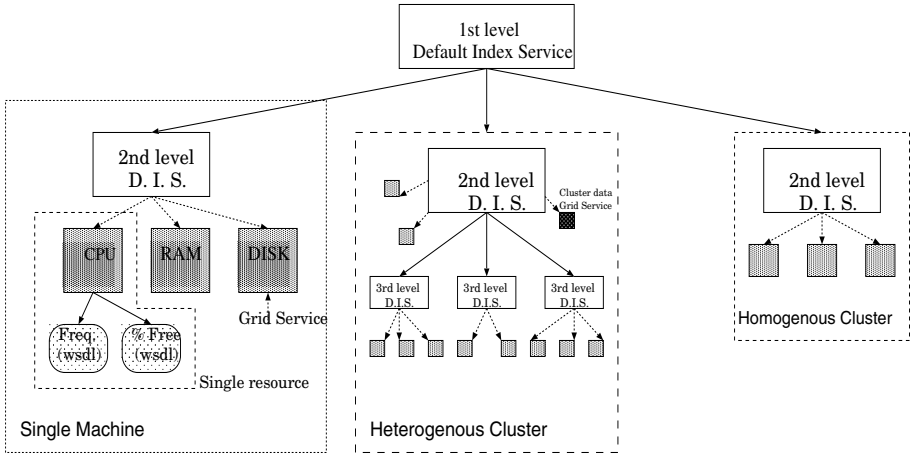

Fig. 3. The structure of the Grid under GT4; grey boxes denote Web Services

Selection Module. This module is the central one because interacts directly with the user, through a GUI quite similar to the one used with GT2. It firstly shows the state of the Grid and its resources (Grid Services of single host and clusters) to the user (from the data structures it receives from the Connection module). Then it gives the user the ability to choose the resources and the options to run correctly his parallel program.

An important improvement of GT4 is the new Trigger Service. The user can chose a set of constraints on one or more resources, and attend that such conditions are fulfilled before the launch of the parallel program. Such constraints are submitted through a particular form to the Trigger Service of the root node (through the Connection module). The connection module comprises an active daemon, able to interface with the Grid root node, that waits for the notifications from the Trigger Service of the root node and quickly transmits such notifications to the selection module that informs the user, by appropriate pop-up windows, that the constraints submitted has been satisfied.

When this happens, a new inquiry to the Default Index Service of the root node is performed and the state of the Grid is updated. Such query can also be performed manually at every moment by the user.

The user chooses the nodes that best fit his requirements and requests to launch the parallel job on that nodes: the Selection module passes such information to the Execution module.

Execution Module. It manages the creation of executables and starts their execution on the Grid, interacting directly only with the GRAM component of GT4. The Execution module receives data from the Selection module, and, for every host involved in the execution of the parallel program, creates an ad-hoc XML file with the running configuration. Such XML files follow an ad-hoc schema that allows to run the program on every GT4 GRAM Service. XML files are created by a sub-module that parses the structure of the schema and creates

the XMLs. Then, every XML file is staged to the corresponding host and the program is launched through the GRAM services of the selected hosts.

During and after the execution, all the output and error messages are reported to the Execution module and then to the user attention through the GUI. Such control of the status of the running job and the staging of all files (XMLs, inputs, outputs) has been obtained taking advantage of the new transfer protocol of GT4 RFT (Remote File Transfer), built on the old GridFTP one. After the execution is correctly finished, a detailed report is sent to the user, together with performance information (e.g. execution time on every node and other statistics that allows the user to understand potential bottlenecks in the set of nodes).

6 Conclusions

We presented a migration experience from Globus Toolkit 2 to Globus Toolkit 4 of a simple tool for the discovery and selection of computational resources and execution of parallel applications in a Grid environment.

We discuss the main differences between GT2 and GT4, particularly for the Monitoring and Discovering subsystem, more directly involved in our porting experience. We try to explain also the impact of some "philosophical" differences between the two Globus versions.

GT4 turns out to be more powerful and flexible, and it makes easier the design and implementation of our tool. In particular, GT4 facilitates the creation of the Grid structure, provides a more direct support to the view and selection of Grid resources, and makes easier the management of collective information about clusters. Moreover, the Trigger Service allows to enhance the capabilities of the Selection module.

The proposed work is a first step towards the implementation of a environment for the efficient execution of parallel structured applications on the computational Grid. There are worldwide several efforts in this direction: among the most important we mention the VGrADS Project [12], which is the prosecution of the earlier GrADS Project [6].

Acknowledgement

This work was supported in part by the Project PRAI-FESR Regione Liguria 2006, Action 28 'Software tools for the development of distributed high performance computing platforms for industrial research'.

References

1. Y. Chen, Y. Li, Z. Gong, Q. Zhu, A framework of a tree-based Grid Information Service, Proc. IEEE Int. Conf. on Services Computing (SCC'05), IEEE Computer Society, 2005.
2. F. Conti, Tools for execution and simulation of structured parallel programs in a GRID environment, Master Thesis, DISI, University of Genova (I), 2005.

3. L. Ferreira et al., Introduction to Grid Computing with Globus, Ibm Redbooks, 2003.
4. I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, Grid services for distributed system integration, Computer, 35, 6, 2002.
5. I. Foster, C. Kesselman, The GRID 2: Blueprint for a new computer infrastructure, Elsevier, 2005.
6. GrADS: http://www.hipersoft.rice.edu/grads/.
7. A. Grimshaw, Grid services extend Web services, SOAWebServices Journal, Jul. 24, 2003 (http://webservices.sys-con.com/read/39829.htm).
8. GT4: http://www.globus.org/toolkit/docs/4.0/.
9. B. Jacob, M. Brown, K. Fukui, N. Trivedi, Introduction to Grid Computing, IBM Redbooks, 2005.
10. Migrating from GT2 to GT 4.1.0: http://www.globus.org/toolkit/docs/development/4.1.0/toolkit-migrating-gt2.html.
11. B. Sotomayor, The Globus Toolkit 4 Programmer's Tutorial, 2005 (http://gdp.2 globus.org/gt4-tutorial/).
12. VGrADS: http://vgrads.rice.edu/.