A Model for Transparent Grid Interoperability

Daniel Chaffe Stone
UFGNet, Universidade Federal de Goiás
CEP 74690-815 – Goiânia-GO – Brazil
stone@ufgnet.ufg.br

Márcio Nunes de Miranda
Instituto de Computação – Universidade Federal de Alagoas
CEP 57072-500 – Maceió-Al – Brazil
marcio@tci.ufal.br

Fábio M. Costa
Instituto de Informática – Universidade Federal de Goiás
CEP 74690-815 – Goiânia-GO – Brazil
fmc@inf.ufg.br

Abstract

Grid Computing is a valuable contribution to the growing need for more computational power. However, grids are still not completely available and there are different solutions and implementations.
While we can certainly benefit from a single grid platform, many resources are still split across different grid middleware solutions. Some projects try to solve this problem by creating an additional layer to communicate the user requests through all participating grids. Our purpose is to present an alternative solution, where native requests are transparently translated through a simple protocol and forwarded to other grids without the need for learning a new interface.

Motivation

Today we have several different middlewaresolutions to build grids. One of them is the Globus toolkit, defined by the Globus Alliance [1]. Integrade [2] is another solution. These two implementations will be the focus of our work, although there are other equally important solutions, such as Legion [3] and OurGrid [4]. While diversity is a good thing, it brings back the original problem: computing power being wasted. It is possible that inside the same organization, in different departments, different grid solutions have been deployed for their particular needs. If one of these grids is not being actively used, the organization is certainly losing important resources. Furthermore, because of the use of different solutions, users of one grid are unable to automatically expand their available computing power through the addition of other available resources managed by different grid middleware.
In this work we present an approach to better use all the available resources based on the design of an underlying adaptation protocol. This protocol will enable one to build a bridge inside one grid in order to translate requests and information to a standard format and to forward such standardized information to others bridges running in different grids, where the reverse translation (from the standard format to the local protocols) would take place, all without the need for human intervention.

Background: Globus and InteGrade

Globus: The Globus Toolkit 4 [5] is a set of programs and libraries used to build grid systems. Among other features, it has resource management and discovery services, and a well-documented interface, which can be accessed via Web Services. We are particularly interested in two components: the Globus Resource Allocation Manager (GRAM) and the Monitoring and Discovering Service (MDS). GRAM, as the name suggests, manages resource allocation. Its interface allows clients to ask for the required resources and to submit programs, along with their parameters to execute. It also responsible for monitoring tasks and resources and to send notifications about the status of a task. It also should send the information about its resource status to MDS. MDS, in turn, acts as a central place (a registry) to keep information about all available resources in the grid system. Such resources can be queried by clients via Web Services and are inserted in the registry by GRAM using XML-based information, or by adapters.

InteGrade: Integrade [6] is a grid middleware architecture based on CORBA. It uses the available ORB services and has its own services exported as CORBA IDL interfaces. We are particularly interested in the Global Resource Manager (GRM), the Local Resource Manager (LRM) and in the ORB Trader service, used by GRM to store information about nodes. LRM runs in each node and is responsible for collecting the node status and available resources and reporting it back to the GRM. It also participates in resource reservation and job execution. GRM receives the information from the LRMs and stores it in the ORB Trader. It uses this information for selecting candidates for job executions when needed. Inter-GRM communication is organized in a hierarchical form. Each GRM consolidates the information about the nodes in its cluster and sends it to an upper level GRM. When a GRM is unable to fulfill a local request, it sends the request up in the hierarchy and so on.

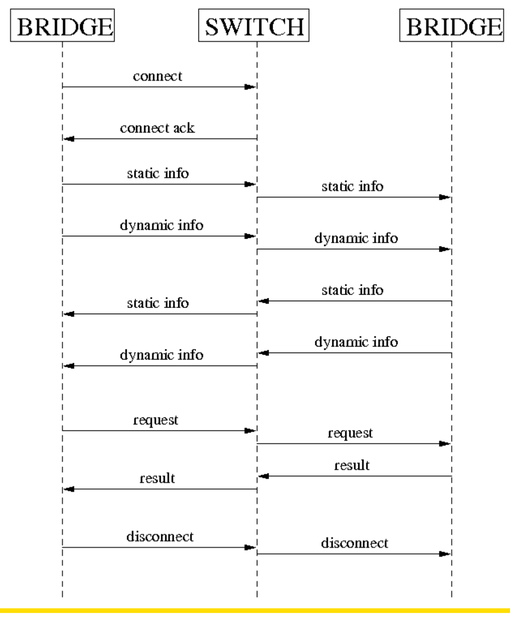

Figure 4:
Sequence Diagram of the Protocol

Architecture

Figure 1 illustrates the overall idea of our proposed architecture. As seen in the figure, a central switch interconnects the bridges located inside each grid. The switch is a very simple component and has two basic functions: to connect the bridges and generate a unique ID for each one, and to forward messages between the bridges. The ID will be used as the bridge address in the message forwarding function. It is also possible to generate a broadcast message to all bridges using a reserved ID. The broadcast messages will be used to advertise changes in the available resources, or by the switch to indicate the disconnection, by timeout, of one bridge.

Bridges in turn are responsible for translating requests and resource information from one grid's local format into a standard, grid-independent, format, so that they can be understood by other grids and, conversely, to translate information and requests from the standard format into the local format used in a particular grid. A bridge implements two interfaces. The former is well-defined and is common among all bridges. The latter depends on the particular architecture of the grid. In this work, although we present possible interfaces for Globus and InteGrade, the implementation and interface of a bridge for a particular grid architecture is entirely up to the bridge developer.

The common interface is used to implement the communication between the bridge and the switch. Both must listen to messages coming from each other. This communication uses the underlying protocol described below. The grid-specific interface is used to define the communication between the bridge and the internal components of the grid where it belongs to. This interface is used to represent particular nodes of the grid or even the entire grid. It should provide a means for transparent interoperability with different grids. It should also interface with a grid's elements and collect the necessary information about them and export such information to other grids.

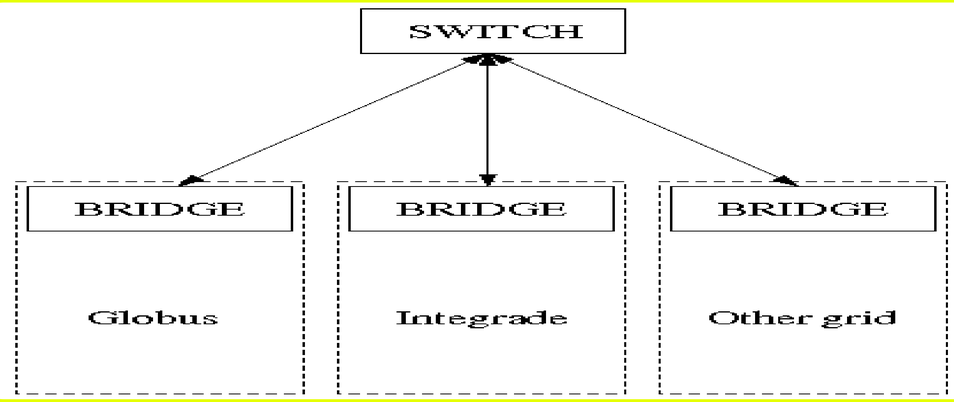

Figure 1:
Overal Architecture

The InteGrade Bridge

An InteGrade bridge can be built by inserting an element in the grid that simulates several LRMs to the GRM. Each LRM can be either a single node in another grid or consolidated information from another grid. A resource reader must be created to interact with the ORB Trader so that we can obtain the available GRM resources. Each LRM will be created from information received by the bridge from the switch. This information must be translated to the appropriate InteGrade service requests. The information obtained from the ORB Trader must be translated to the standard format and send to the switch. Although InteGrade has a hierarchical solution for Inter-GRM communication, a simulation of a GRM node would simply provide a consolidated view of the InteGrade grid. A finner granularity would be better for resource allocation by the scheduler. Because of this, the placement of the bridge is also important. We must insert it in the top GRM node to provide some granularity and also have the consolidated information form the nodes below it.

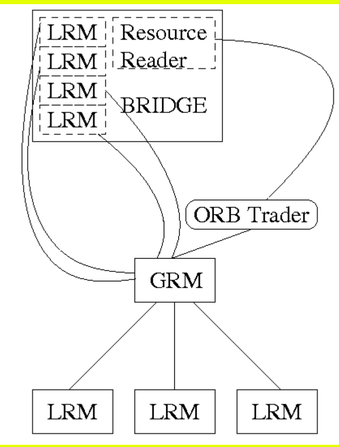

Figure 2:
The InteGrade Bridge

The Globus Bridge

A Globus bridge can be built by simulating GRAM. The bridge publishes the grid resources in the MDS and collects the other available resources from it. The bridge can simulate multiples GRAMs, each one for a different grid cluster. Upon receiving new information from the switch, the bridge must translate it to the appropriated requests to MDS. The bridge must also acquire information about the available resources from MDS and send it to the switch as well as other requests received.

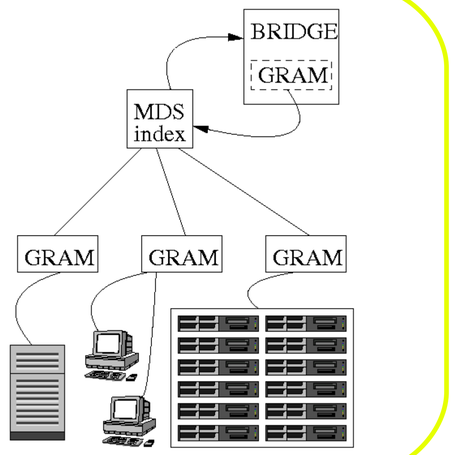

Figure 3:
The Globus Bridge

Protocol

The purpose of this protocol is to provide a single and generic standard format, allowing the exchange of resource information and requests among different grids. The protocol carries three kinds of messages: dynamic resources, static resources and requests. Dynamic resources are information that changes periodically with time, such as the amount of free memory and CPU usage. Static resources are fixed and do not change during operation of the grid, such operating system and the processor model. Finally, requests can be related to job execution or job cancellation, among others. The bridge at each grid environment translates messages from the native grid format to a standard format and sends these messages to the switch. The switch sends the received messages in standard format to all connected bridges. The bridges then translate from the standard message format back to the native format of their respective grids.
Figure 4 shows a possible interaction between one bridge, the switch and any other bridge. The first action bridge takes must be to send a connect request to the switch, which will assign a unique id to the bridge. After receiving its id, the bridge must send its static and dynamic information, which will be forwarded by the switch to all participating bridges.
Each bridge, upon receiving a message about a new bridge, along with its ID, should send back, only to that ID, its static and dynamic information. After this point, any connected bridge is able to send requests to other bridges. Finally, when one bridge leaves the system, it must send a final message to the switch. This message will be forwarded to all other bridges, indicating that the resources have changed.
XML is used to encode the messages, allowing the future addition of new elements in the messages without preventing bridges that use old versions of the protocol from working. The format of the messages is: a single tag "connect" for connection requests; an “acknowledge” tag with a numeric ID for the bridge connection request; an error tag with a string describing the error; and a message tag. The message tag is composed by a "src" numeric tag, with the ID of the sending bridge, a "dst" numeric tag with the destination bridge or zero for a broadcast message. There is also a "type" and "subtype" numeric tag, indicating the kind of message being sent and a "data" tag with the message itself.
The "type" and "subtype" are used to indicate a new bridge, a removal of a bridge, a ping and its response, requests for information, job execution and their responses and results. The "data" tag carries the payload for the message, which can be the dynamic or static information about a grid or the parameters of a job request.

Final Remarks

We have described a simple model for transparent grid interoperability. It allows the end user of one grid, which is comfortable with that grid's programming model, to submit job requests within his/her grid and have them propagated through to other grids, expanding the power of his/her grid without the need for human intervention.
While related work such as OurGrid [4], GAT [7], Nimrod/G [8] and Condor-G [9], tackle the same problem through an additional integration layer on top of which applications must be built, our approach prevents the need to learn an extra interface by letting programmers to develop applications based on the native grid interfaces they are used to. Another approach has been proposed by Nascimento in [10]. It provides transparent communication between Globus and InteGrade by creating alternative schedulers for Globus and InteGrade. While our solution uses the same grids as a case study, it is sufficiently generic to include other grids and does not influence nor change their original implementation.
We are currently working on the implementation of a functional prototype, and in future work we intend to design a bridge capable of translating inter-processes communication, such as from MPI to BSP an vice versa. We also aim to define an inter-switch protocol, allowing established grid clusters to share resources with other grid clusters.

References

[1] The Globus Alliance. http://www.globus.org/, December 2006.
[2] InteGrade: Object-oriented middleware for grid computing. http://integrade.incubadora.fapesp.br/portal, December 2006.
[3] Legion: A worldwide virtual computer. http://www.legion.virginia.edu/, December 2006.
[4] The Ourgrid Project. http://www.ourgrid.org/, Dec. 2006.
[5] I. Foster. Globus Toolkit version 4: Software for service-oriented systems. In IFIP International Conference on Network and Parallel Computing, pages 2–13, Springer-Verlag LNCS 3779, 2006.
[6] A. Goldchleger. Integrade: Um sistema de middleware para computação em grade oportunista (Integrade: A middleware system for opportunistic grid computing). Master's thesis, Department of Computer Science - University of São Paulo, December 2004.
[7] G. Allen, et al. The grid application toolkit: Toward generic and easy application programming interfaces for the grid. In Proceedings of the IEEE, volume 93, pages 534 – 550. IEEE, March 2005.
[8] R. Buyya, D. Abramson, and J. Giddy. Nimrod-G: An architecture for a resource management and scheduling system in a global computational grid, 2000.
[9] I. F. M. L. James Frey, Todd Tannenbaum and S. Tuecke, editors. Condor-G: A Computation Management Agent for Multi-Institutional Grids. IEEE, August 7-9 2001.
[10] A. C. G. do Nascimento. Integração de ambientes heterogêneos de grades computacionais: Globus e InteGrade. Master's thesis, UFG - EEEC, 2006.

Acknowledgments

This work has been partly funded by CNPq, The Brazilian Council for Scientific and Technological Development, grants 506.689/2004-2, 478.620/2004-7 and 550.094/2005-9.