Legacy Code Repository with Broker-based Job Execution

Gabor Kecskemeti[1,3], Gabor Terstyanszky[2,4], Tamas Kiss[2,3], Peter Kacsuk[1,3],

[1]*MTA SZTAKI Computer and Automation Research Institute*
H-1518 Budapest, P. O. Box 63, Hungary
[2] *Centre for Parallel Computing, University of Westminster*
115 New Cavendish Street, London, W1W 6UW
[3]*CoreGrid Institute on Grid Systems, Tools and Environments*
[4]*CoreGrid Institute on Grid Information, Resource and Workflow Monitoring Services*

Abstract

As Grid technology matures more and more production Grids become available to run computationally intensive scientific applications. However, as these Grids are based on different middleware solutions interoperation between these platforms is one of the major challenges the Grid community is facing today. The P-GRADE/GEMLCA portal is a workflow oriented Grid portal that supports the execution of workflow components simultaneously on multiple Grids based on different underlying technology. Jobs can be submitted to Globus or LCG/gLite based Grids, or can be selected from the GEMLCA legacy code repository. However, as GEMLCA is implemented on top of GT4, it was capable to interact only with GT2 and GT4 based Grids until recently. Moreover, legacy codes selected from the GEMLCA repository were statically mapped to resources. This paper describes how GEMLCA has been ported to the LCG/g-Lite based EGEE infrastructure. Besides simply porting GEMLCA to another middleware it had to be made capable to interact with the EGEE broker solution. A legacy code can not only be selected from the repository, but using its legacy code interface description it is also defined which resources are capable to execute the given code. Based on this information the broker can find the most suitable resource at workflow execution. As a result of this integration the P-GRADE/GEMLCA portal is capable to interact with both Globus and LCG/g-Lite based Grids by the means of either direct or broker-based job submission, or by browsing and selecting the executable from the legacy code repository. The described work illustrates how two important components of production Grids, a legacy code solution and a Grid broker can be successfully integrated.

1. Introduction

The Grid computing environment requires special Grid enabled applications capable of utilising the underlying Grid middleware and infrastructure. As the Grid becomes stable and commonplace in both scientific and industrial settings, a demand is created for porting a vast legacy of applications onto the new platform. Companies and institutions can ill afford to throw such applications away for the sake of a new technology, and there is a clear business imperative for them to be migrated onto the Grid with the least possible effort and cost.

Grid computing is now progressing to a point where reliable Grid middleware and higher level tools will be offered to support the creation of production level Grids. A high-level Grid toolkit should definitely include components for turning legacy applications into Grid services. The Grid Execution Management for Legacy Code Applications (GEMLCA) [1] enables legacy code programs written in any source language (Fortran, C, Java, etc.) to be easily accessed through a Grid Service interface without significant user effort. GEMLCA does not require any modification of, or even access to, the original source code. A user-level understanding, describing the necessary input and output parameters and environmental values such as the number of processors or the job manager required, is all that is needed to port the legacy application binary onto the Grid. Once the interface is described GEMLCA legacy codes are also available for other authorised users to run them with custom parameters or to include them in their Grid workflows.

GEMLCA was originally implemented to support service oriented Grid middleware, namely Globus toolkit version 4 (GT4) [2]. GEMLCA was implemented as a set of GT4 Grid services that submit legacy code jobs to local job managers like Condor or PBS using the GT4 WS-GRAM service. However, it became apparent that different production Grids currently use different Grid middleware without providing interoperability between them. In order to offer a legacy code converter tool, like GEMLCA, that supports the large variety of existing Grid middleware solutions, a more flexible approach was required. GEMLCA was re-engineered to be able to utilise different back-end plug-ins submitting to different Grid middleware.

The original GEMLCA concept only supported direct mapping of legacy code jobs to resources. GEMLCA legacy codes were tightly bound to a particular resource. At job execution or workflow creation the user first selected the target resource hosting the executable legacy code and then defined custom parameters. As more and more production Grids are experimenting with resource brokers it was necessary to extend the GEMLCA concept in order to interact with these brokers.

This paper describes how GEMLCA is capable to support multiple Grid platforms by creating middleware specific back-ends. These back-ends can also interact with existing resource brokers, like the LCG/g-LITE broker of the EGEE Grid.

2. Legacy Code Support for Multiple Grid Middleware

GEMLCA represents a general architecture for deploying legacy applications as Grid services without re-engineering the code or even requiring access to the source files. The deployment of a new legacy code service requires only a user-level understanding of the legacy application, i.e., to know what the parameters of the legacy code are and what kind of environment is needed to run the code (e.g. multiprocessor environment with ‘n’ processors). The execution environment and the parameter set for the legacy application is described in an XML-based Legacy Code Interface Description (LCID) file that should be stored in a pre-defined location. This file is used by the GEMLCA Resource layer to handle the legacy application as a Grid service.

GEMLCA is integrated with the workflow-oriented P-GRADE Grid portal [8]. The P-GRADE portal enables the graphical development of workflows consisting of various types of executable components (sequential, MPI or PVM programs), execution of these workflows in Globus-based Grids relying on user credentials, and finally the analysis of the correctness and performance of applications by the built-in visualization facilities. The P-Grade portal has also been extended with a GEMLCA administration portlet. This portlet hides the syntax and structure of the LCID file from users so that users do not have to know LCID specific details, and do not have to be familiar with possible modifications in legacy code description whenever a new GEMLCA release would require it. The user has to specify exactly the same parameters as in the XML file but this time using a simple Web form. The LCID file is created automatically and uploaded by the portal to the appropriate directory of the GEMLCA service.

GEMLCA has been internally designed in three layers. Each of these layers simulates an encapsulated black-box that is committed to deliver a well-defined functionality to the layer above that, independently of the underlying Grid middleware solution.

The first, *Front End Layer*, offers a set of functionalities as Grid Services. Any authorised Grid client can utilise the functionalities to deploy and use legacy code programs on a GEMLCA Resource. The GEMLCA functionalities are expressed with the help of the following GEMLCA Grid services:

- *GLCAdmin:* After authentication, *GLCAdmin* enables the client to modify the XML-based LCID file of an already deployed legacy code, or to create a new LCID file and upload it to the GEMLCA Resource.
- *GLCList:* Returns a list of already deployed legacy codes.
- *GLCProcess: S*ubmits the legacy application using GEMLCA to a compute server and gets job status and results back.

The GEMLCA architecture manages the concepts of legacy code processes (*LCProcess*). GEMLCA manages these processes, and they do not have to be confused with ordinary Grid processes. The *Core Layer* is in charge of the management of these internal concepts. When a GEMLCA Grid Service is created, automatically a *LCProcess* object is constructed within it. The *LCProcess* contains the memory structure to handle the legacy code input parameters and manage a unique hard disk environment to store legacy code input and output files. When a *LCProcess* is submitted, a *GLCEnvironment* object is created with new memory and hard disk structures. Using these concepts, several *GLCEnvironments* can be created, submitted and destroyed from a single *LCProcess*. Additionally, it allows multiple submissions in a multi-user environment, where user-specific information, input and output files and parameters, can be preserved separately from other instances running on the same node.

The final *Back End Layer,* is connected to the Grid middleware on the host where the

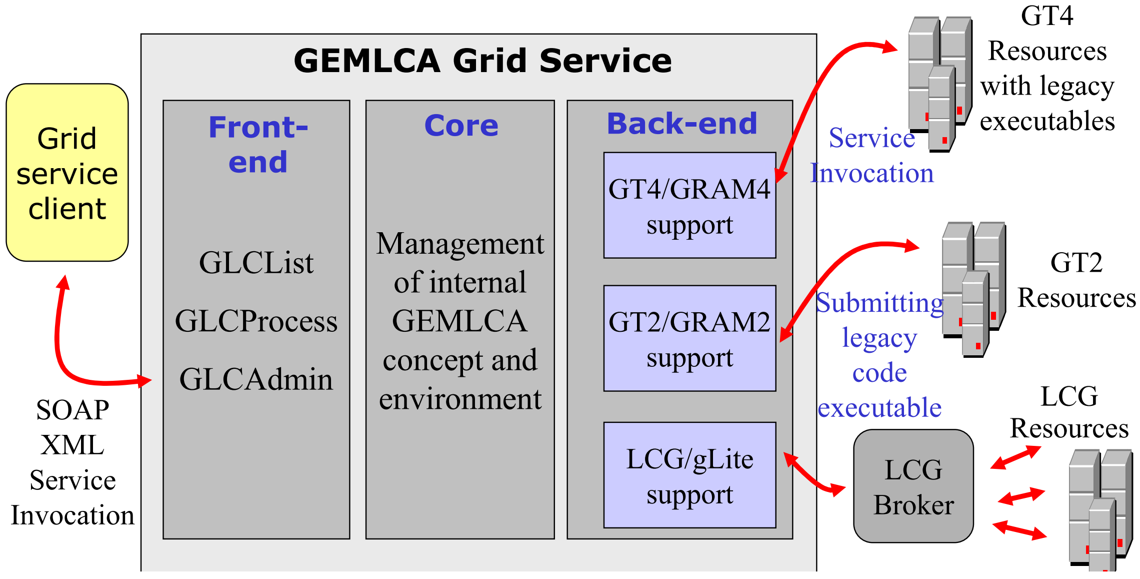

Figure 1 The three layered GEMLCA architecture with multiple back-ends

architecture is being deployed. It knows the different ways to contact, submit jobs, and get status back from the correlated middleware. This layer, as it was described in the previous section, was originally tightly coupled with the GT4 Grid middleware. However, this feature restricted the deployment of the GEMLCA architecture to GT4 based Grids only. As the largest production Grids currently use either GT2, like the UK National Grid

Service [3] or the Open Science Grid in the US [4], or LCG/g-Lite [5] like the EGEE Grid [6] it was inevitable to redesign the GEMLCA back-end layer to support multiple Grid middleware solution.

Figure 1 shows the three layer GEMLCA architecture with multiple back-ends. Besides the original GT4 support described in [1] GEMLCA has been extended to submit to GT2-based Grids. As detailed in [7], the GT2 GEMLCA back-end allows users to create legacy code repositories on the GEMLCA resource, and select and submit the codes to the remote GT2 gatekeeper utilising Condor-G. This paper concentrates on the design principles of the third back-end plug-in that interfaces with LCG/g-Lite based Grids through the EGEE broker.

3. GEMLCA in broker-based Grids

Production P-GRADE Grid portals, installed in the EGEE Grid, submit and execute jobs on EGEE resources using the LCG broker. GEMLCA has to be extended towards the EGEE platform to run legacy code applications on the EGEE Grid using the LCG broker. In order to run legacy code applications on EGEE resources with the LCG broker the following requirements should be satisfied:

1. GEMLCA should be able to submit jobs through the LCG broker using its JDL language instead of using the Globus's RSL.
2. GEMLCA should be able to send input files and retrieve output files using the LCG broker.
3. The LCG broker should submit legacy code applications only those resources where they can be executed taking into consideration their specific configuration and execution requirements.
4. The LCG broker based job submission for legacy code applications should be integrated seamlessly with the P-Grade Grid portal to minimise user efforts to learn and apply this solution.

The GEMLCA team created a legacy code repository, called GEMLCA-R, to run legacy code applications as jobs on the Grid. The repository contains the binary and may include some input files of the legacy code applications. Information about these applications can be retrieved through the repository list. Code owners first, should check whether their applications can run on a resource next, they should upload the applications into the GEMLCA-R, assigned to the resource. After uploading legacy applications users with valid certificates and the required authorisation can select and execute legacy codes using a list of available legacy code applications registered with a GEMLCA-R.

We identified two approaches to submit legacy code applications through the LCG broker:

1. One-to-one brokering
2. One-to-many brokering

3.1. One-to-one brokering

According to this approach a legacy code application, implemented as a GEMLCA Service, is assigned to one particular EGEE resource. To describe this assignment the service should extend the JDL file to restrict the submission of the legacy code to the allocated resource. As a result, when GEMLCA submits the legacy code application to the LCG broker, it forwards the code to the resource, which is specified in the JDL file. In this approach legacy codes could be assigned to resources using either the unbalanced or balanced job submission.

Unbalanced job submission. Each EGEE resource should have its own repository, which is maintained by GEMLCA services mapped to the resource. The mapping between the resource and the service is defined by the code owner and done by the P-GRADE portal. Submitting a legacy code as a GEMLCA service to an EGEE resource requires an extra input file, which defines the EGEE resource where the execution should take place. Using the unbalanced job submission, the user selects an EGEE resource specifying a GEMLCA-R. The portal forwards the job to the LCG broker, which is a single point of entry to the EGEE Grid. The broker sends the job to the resource defined by the user not performing any kind of brokering activities.

Balanced job submission. It is the extension of the unbalanced job submission. The P-GRADE portal has its own built-in broker service, which can interface with GT2 and LCG Grids. The built-in broker selects the EGEE resources for the GEMLCA service using an aggregated list, which contains all legacy codes registered with GEMLCA services. The aggregated list leads to longer job setup times because of queries made to create the list and caching the legacy code list of GEMLCA services. Using the built-in broker, no resource selection UI is available. In this approach the user forwards the job to the built-in broker that selects an EGEE resource specifying a GEMLCA-R. The portal forwards the job to the LCG broker, which sends the job to the EGEE resource selected by the built-in broker.

3.2. One-to-many brokering

This approach assumes that the list of EGEE resources where a legacy code can be executed has been compiled and is available. The GEMLCA service, representing the legacy code, has to add this list to the JDL file. Having this list the LCG broker is able to select one of the resources, which is capable to run the legacy code. The user interface of the portal with broker is not different from the user interface without broker.

3.4 JDL file generation

The JDL file can be generated by either the Grid portal or GEMLCA. If the portal creates the JDL file, the GEMLCA does not even have to understand the contents of the JDL file. In this case GEMLCA acts as a "primitive" job submitter, e.g. GEMLCA argument integrity check is not used. To avoid this situation, the GEMLCA service generates the JDL file. This can be done in two ways: complete and partial JDL generation.

Complete JDL generation. The GEMLCA service has a full control over the contents of the JDL file and it provides all the information for the JDL file. To achieve it the legacy code descriptor has to be extended to support the JDL file generation. At each submission the GEMLCA service has to generate the JDL file for the broker. Therefore, there is no need for a JDL parser on the service side. The portal has a JDL writer engine, which could be used for the JDL generation. The drawback of this solution is the user has to be familiar with some EGEE specific details.

Partial JDL generation. GEMLCA creates only those JDL parts, which are different at submissions. This approach is based on a JDL file with a sample submission description. This file can be customised according to the actual state of the legacy code environment (e.g. arguments, inputs). The customization means that the GEMLCA service has to parse the JDL document and write a new one containing the details of the execution. It is quite important that the original JDL might hold some information on resource requirements, and also some job manager and execution specific data (like MPI execution preparation or queue setup), which are the same for all resources. If the generated and the sample JDL files are very similar, the job submission based on these files will be very similar. It happens if the GEMLCA service has to update only the resource restrictions but not the others.

The only question is where to get the sample JDL from. The administrator of the legacy code should pass it with the legacy code description. The P-GRADE portal may offer the generated and submitted JDLs for the GEMLCA or as an extra option the administration portlet should upload the JDL file as an attachment to the description.

4. Integration Within the STE Institute

GEMLCA is serving the users of the largest production Grids all around the world. Extending GEMLCA with multiple back-ends supporting different Grid middleware solutions opened the way to install GEMLCA on different Grid infrastructures serving a much larger user community. The achieved results thus represent a significant link between Coregrid and Grid end users representing research and industry. Within Coregrid task 7.4 researchers identified the challenges and solutions to deploy legacy applications in a Grid environment. Given the importance of legacy applications as described in section 1 of this paper, a component based Grid platform have to include solutions to wrap and present these applications on the Grid. GEMLCA is the most widely used solution to fulfil this task and is now capable to support multiple and different Grid platforms and Grid brokers, as described in this paper.

References

[1] T. Delaittre, T. Kiss, A. Goyeneche, G. Terstyanszky, S.Winter, P. Kacsuk: GEMLCA: Running Legacy Code Applications as Grid Services, “Journal of Grid Computing” Vol. 3. No. 1.

[2] Globus Team, Globus Toolkit 4.0 Release Manuals, http://www.globus.org/toolkit/docs/4.0/

[3] The UK National Grid Service Website, http://www.ngs.ac.uk/

[4] The Open Science Grid Website, http://www.opensciencegrid.org/

[5] The gLite website, http://glite.web.cern.ch/glite/

[6] The EGEE web page, http://public.eu-egee.org/

[7] T. Kiss, G. Terstyanszky, G. Kecskemeti, Sz. Illes, T. Delaittre, S. Winter, P. Kacsuk, G. Sipos: Legacy Code Support for Production Grids, Conf. Proc. of the Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing November 13-14, 2005, Seattle, Washington, USA

[8] P. Kacsuk and G. Sipos: Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal, Journal of Grid Computing Vol. 3. No. 3-4., 2005, Springer, 1570-7873, pp 221-238