Experiences with Deploying Legacy Code Applications as Grid Services using GEMLCA [1] [2]

A.Goyeneche[1], T.Kiss[1], G.Terstyanszky[1], G.Kecskemeti[1], T.Delaitre[1], P.Kacsuk[2], S.C. Winter[1]

[1] Centre for Parallel Computing, Cavendish School of Computer Science
University of Westminster,115 New Cavendish Street, London, W1W 6UW
[2] MTA SZTAKI Lab. of Parallel and Distributed Systems,
H-1518 Budapest, P.O. Box 63, Hungary
gemlca-discuss@cpc.wmin.ac.uk
www.cpc.wmin.ac.uk/gemlca/

Abstract. One of the biggest obstacles in the wide-spread industrial take-up of Grid technology is the existence of a large amount of legacy code programs that is not accessible as Grid Services. On top of that, Grid technology challenges the user in order to intuitively interconnect and utilize resources in a friendly environment. This paper describes how legacy code applications were transformed into Grid Services using GEMLCA providing a user-friendly high-level Grid environment for deployment, and running them through the P-GRADE Grid portal. GEMLCA enables the use of legacy code programs as Grid services without modifying the original code. Using the P-GRADE Grid portal with GEMLCA it is possible to deploy legacy code applications as Grid services and use them in the creation and execution of complex workflows. This environment is tested by deploying and executing several legacy code applications on different sites of the UK e-Science OGSA testbed.

1 Introduction

There are many efforts all over the world to provide new Grid middleware concepts for constructing large production Grids. As a result, the Grid community is in the phase of producing third generation Grid systems that are represented by the OGSA (Open Grid Services Architecture) [1] and WSRF (Web Services Resource Framework) [2] standards. On the other hand relatively little attention has been paid to how end-users can survive in the rapidly changing world of Grid generations. The primary

[1] The work presented in this paper was initially supported by EPSRC funded project (Grant No: GR/S77509/01): A proposal to evaluate OGSA/GT3 on a UK multisided testbed.

[2] The integration of GEMLCA, P+Grade and other Grid tools and projects is currently supported by "*CoreGrid*", network of excellence in "*Foundations, Software Infrastructures and Applications for large scale distributed, Grid and Peer-to-Peer Technologies*".

goal of our research is to construct a high-level Grid application environment where the end-users can:

- Deploy and use any legacy code as Grid Services.
- Use these Grid Services in order to easily and conveniently create complex Grid applications.

In an ideal Grid environment users would be able to access Grid Services through a high-level user-friendly Grid portal. More than that, users would not only be capable of using such services, they could dynamically create and deploy new services, and also construct complex Grid workflows in a convenient and efficient way. All these services can be either specifically designed Grid Services, or legacy code programs deployed as Grid Services.

Some of these objectives are achieved by an integrated high-level Grid execution environment [3] consisting of Grid Execution Management for Legacy Code Architecture (GEMLCA) and the P-GRADE Grid portal. GEMLCA enables legacy code programs written in any language (Fortran, C, Java, etc.) to be easily deployed as a Grid Service without any user effort. The integration of GEMLCA with the P-GRADE Grid portal results in an OGSA-based Grid portal. Using this integrated solution legacy codes can be deployed as Grid Services and accessed by authorized users from the portal. As a consequence, legacy codes (either sequential or parallel ones) can be used as workflow components to build complex Grid applications.

In the current paper the integrated GEMLCA & P-GRADE Grid portal is presented describing experiences deploying and running different legacy applications as Grid Services.

2 Deploying Legacy Code applications on the Grid

Many currently available industrial and scientific applications were written well before Grid computing or service-oriented approaches appeared. The incorporation of these legacy code programs into service-oriented Grid architectures with the smallest possible effort is a crucial point in the widespread industrial take-up of Grid technology.

There are several research efforts aiming at automating the transformation of legacy code into a Grid Service. Most of these solutions are based on transforming legacy code applications into Web Services outlined in [4] using Java wrapping in order to generate stubs automatically. One example is presented in [5], where the authors describe a semi-automatic conversion of programs written in C into Java using Java Native Interface (JNI). After wrapping the native C application with the Java-C Automatic Wrapper (JACAW), MEdiation of Data and Legacy Code Interface tool (MEDLI) is used for data mapping in order to make the code available as part of a Grid workflow.

Different non-wrapping approaches are presented in [6] and [7] but these solutions only define the principles of legacy code transformation, and do not specify an environment or tool to do the automatic conversion.

GEMLCA is based on a different principle. Instead of wrapping the application GEMLCA hides it behind a set of Grid Services leaving the legacy code untouched.

The Grid Services layer communicates with the client in order to submit service requests, manages input and output parameters, and contacts a local job manager through Globus MMJFS (Master Managed Job Factory Service) to submit the computational jobs. To deploy a legacy application as a Grid Service there is no need for the program source code and not even for the C header files as in case of JACAW. The user only has to provide the program attributes and parameters. The legacy code can be written in any programming language and can be not only a sequential but also a parallel MPI or PVM code that uses a job manager like Condor, and where wrapping can be difficult.

3 GEMLCA

GEMLCA is a Grid architecture with the main aim of exposing legacy code programs as Grid Services without reengineering the original code, and offering a user-friendly interface.

GEMLCA has been designed [8] as a three-layer architecture: the first (front-end) layer, offers a set of Grid Service interfaces that any authorized Grid client can use in order to contact, run, and get the status and any result back from the legacy code. This layer hides the second (core) layer, which deals with each legacy code environment and their instances as Grid legacy code processes and jobs. The final (back-end) layer is related to the Grid middleware where the architecture is being deployed. The present implementation is based on GT3 and is currently migrated to GT4.

GEMLCA conceptual architecture is shown in Figure 1 where we can identified a Grid client, a GEMLCA Resource which is composed of a set of Grid Services that provides a number of Grid interfaces in order to control the life-cycle of the legacy code execution and a Grid Host Environment provided by the deployment of Globus Toolkit 3 (GT3).

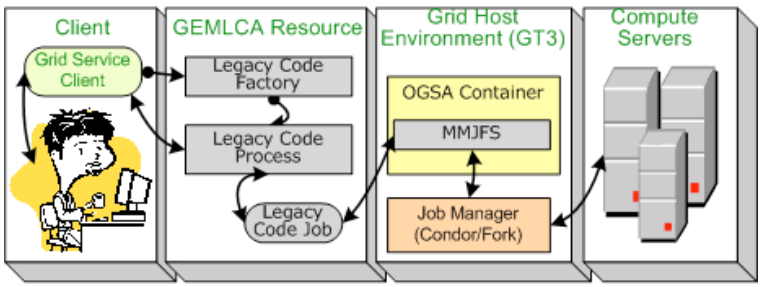

Fig. 1. GEMLCA functional architecture

In order to access a legacy code program, the user invokes the GEMLCA Grid Service client which creates a legacy code instance with the help of the legacy code factory. Following this, the GEMLCA Resource submits the job to the compute server through GT3 MMJFS using a particular job manager.

GEMLCA is been designed using three layers that encapsulate and divide the front end and core functionality with the Grid Host Environment dependant layer. At this time, a GEMLCA version using a GT3 Grid Host Environment is available where a GT4 dependant version is been under development and testing.

4 The integrated GEMLCA P-GRADE Portal

The P-GRADE Grid portal is a workflow-oriented portal which main goal is to enable users to manage the whole life-cycle of creating and executing complex applications in the Grid. This is achieved allowing users to edit, execute and monitor Grid workflows composed of various types of Workflow components (sequential programs, MPI, PVM).

Our goal, as it was mentioned in the Introduction, was to enable end-users to deploy legacy codes as Grid Services, and use them as components of workflows with the least possible effort. The integration of GEMLCA with the P-GRADE portal produces a high-level Grid toolkit environment that achieves this goal. The integration has been done through the creation and use of a number of Grid clients.

In order to create a GEMLCA component in a workflow, the portal user is able to select a GEMLCA Resource and the required legacy code program which has already been deployed on the resource. To achieve this, the Grid GEMLCA client, embedded in the portal, allows the selection of a GEMLCA Resource and a legacy code application from the list returned. This client also allows the change of input parameters and upload of input files.

Once the workflow is completed and saved, the workflow manager, Condor DAGman [9], is used in order to manage GEMLCA jobs. In case of GEMLCA, the DAGMan's PRE, POST, and job submission scripts were modified. The PRE-script has been changed in order to call a GEMLCA client that creates an instance of the legacy code process returning a Grid Service Handle (GSH). Such GSH is used by a further GEMLCA client for setting its parameters, uploading input files using GridFtp and finally submitting the GEMLCA job. The POST-script has also been changed in order to download and make output files available to the user, and destroy the GEMLCA jobs. Alternatively, the output files will be transferred into the next legacy code environment of the workflow if it is required.

Another GEMLCA client is used for checking the status of the legacy code process and jobs in order to let the user know the status of the workflow and each of its components.

5 Legacy Code deployment with GEMLCA

5.1 Legacy Code program deployment using the P-Grade Grid Portal

Most of the solutions to expose legacy code programs as Grid Services require access to the source code. In contrast, in GEMLCA the only significant effort to be done is to create a Legacy Code Interface Description (LCID) file in XML format. The LCID file consists of three sections: the first section – "*environment*" - contains the name of the legacy code and its binary file, job manager to be used (Condor and Fork are supported in the current version of GEMLCA), maximum number of jobs allowed to be submitted from a single legacy code process, and minimum and maximum number of processors to be used; the second section – "*description*" - describes the legacy code in simple text format; the third section – "*parameters*" - exposes the list of parameters defining for each of them its name, friendly name, type (input or output), order, status (compulsory or optional), file, command line, and the regular expression to be used as input validation.

Some users, without basic GEMLCA and XML knowledge, may find difficult to learn how to manually create LCID files. To support them, the generation of LCIDs and the deployment of legacy code applications have been automated using a new Admin Grid Service in the GEMLCA Resource and a new Grid portal interface.

The Admin Grid Service can be accessed through the GEMLCA Administration Portlet which is integrated into the P-GRADE Grid portal, and offers a number of interfaces in order to create the legacy code program deployment environment and its LCID file.

The Portlet, based on the Document Type Definition file stored in the GEMLCA Resource provided by one of the Admin Grid Service interfaces, creates and presents a deployment Web form on the fly. If the legacy code description provided by the user is correct, the Admin Grid Service is used to create the legacy code deployment environment and its LCID file, and Grid FTP is used to upload the legacy code program and its input files. After this point the legacy code is published and available as a Grid Service.

The GEMLCA Administration Portlet hides the syntax and structure of the LCID file from users. As a result, users do not have to know LCID specific details. For example, they do not have to be familiar with possible modifications in legacy code description after new GEMLCA releases.

5.2 Internal legacy code deployment description

Besides the use of the P-Grade Grid portal, a legacy code program can be also deployed manually by the GEMLCA Resource administrator or any general user with access to the GEMLCA Recourse server.

The general user is allowed to deploy legacy code programs within the user's home folder. Any deployed program could only be used through GEMLCA by the Grid users which are locally mapped to the local general user. These legacy codes are considered "*private*" to a set of Grid users. On the other hand, the GEMLCA Resource administrator can also deploy a "*public*" legacy code program that is available to any Grid user mapped in that Grid host. This technique implements the GEMLCA authorization of legacy codes.

After deciding whether a legacy code will be available to either a set of users (*private*) or anyone (*public*) in a given GEMLCA Resource, a folder within the base deployment folder has to be created that gives a unique *private* or *public* identification. The legacy code can be uploaded into this folder from the portal, manually copied or linked from any place in the server without changing the program's internal folder structure. The input files can also be copied or linked into this folder, or made available using a full path folder. Finally, to make the legacy code available the LCID file has to be created.

6 Deployment Examples

In this section the deployment of three legacy codes are described: MadCity traffic simulator [10], GAMESS-UK [11] and MultiBayes [12]. These programs were developed by the University of Westminster, the Daresbury Laboratory, and the University of Reading, respectively. The objective of these deployments is to publish these legacy codes as Grid Services in order to be available for external Grid users. In each example, the challenge faced is presented together with the solution implemented.

At the end of the chapter a list of constraints that have to be considered before deploying a program as a GEMLCA service is included. The constraints are based on the experience of deploying the previously mentioned legacy codes.

6.1 MadCity Traffic Simulator

In this example, a workflow for analysing urban car traffic on road was created that consists of three different components: a Manhattan road network generator, a traffic simulator, called MadCity, and a traffic analyser.

The Manhattan road network generator creates MadCity compatible car networks that are used as inputs for the simulator. MadCity is a discrete time-based traffic simulator that simulates car traffic on a road network, and shows how individual vehicles behave on roads and at junctions. Finally, the traffic analyser compares the traffic density of several simulations of a given city and presents a graphical analysis.

The main objective of this case study was to analyze and test the following points:

- Use of several GEMLCA Resources in a single workflow.
- Execute several legacy codes as Grid Services in parallel in a single and multiple GEMLCA Resources.
- Schedule the workflow and synchronize the Grid Services executions.

- Transfer files from one Grid Service to another in the same GEMLCA Resource and between different GEMLCA Resources.
- Display intermediate and final results in the portal.

In order to meet these objectives, the workflow is configured to use five GEMLCA Resources each one deployed on the UK OGSA testbed sites, and one server where the P-GRADE portal is installed. The first GEMLCA Resource is at the University of Westminster (UK), and runs the Manhattan road network generator (Job0), two traffic simulator instances (Job3 and Job6) and the final traffic density analyzer (Job7). Four additional GEMLCA Resources are installed at the following sites: SZTAKI (Hungary), University of Portsmouth (UK), the CCLRC Daresbury Laboratory (UK), and University of Reading (UK). One instance of the simulator is executed on each of these sites, respectively Job1, Job2, Job4 and Job5 (see Figure 2).

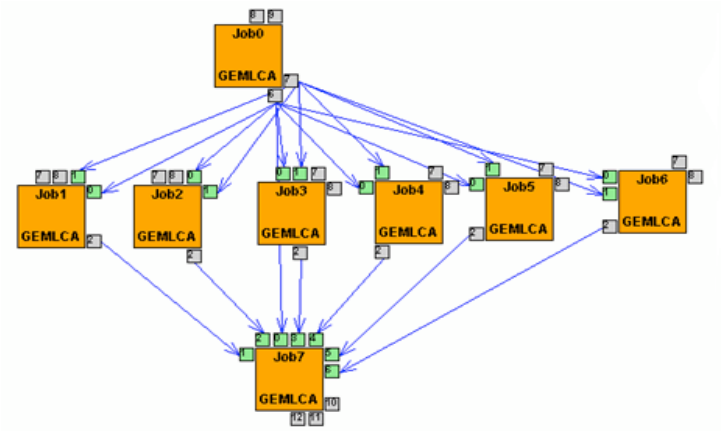

Fig. 2. Workflow graph for analysing road traffic

The MadCity network and turn files are used as input to each traffic simulator instance. In order to analyse the different behaviour of these instances, each one was set with different initial number of cars per street junction, one of the input parameters of the program. The output file of each traffic simulation is used as input file to the traffic analyser.

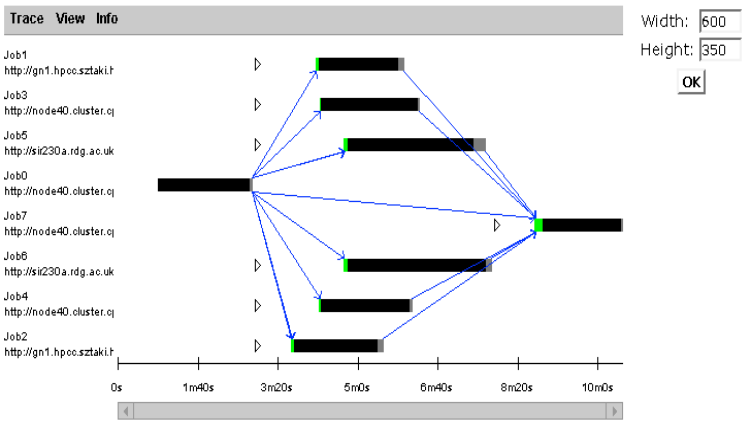

Fig. 3. Workflow execution graph

The described workflow was successfully created and executed from the P-GRADE portal installed at the University of Westminster. The workflow execution graph is shown in Figure 3.

6.2 GAMESS-UK

GAMESS-UK is an ab initio molecular electronic structure program for performing SCF-, DFT-, and MCSCF-gradient calculations using a variety of techniques for post Hartree-Fock calculations.

The sequential version of GAMESS-UK, which has a complex folder structure with multiple files, was deployed. The application is an executable binary file that has one command line input parameter: a string that specifies the full path and name of the input file containing the program execution configuration. The ".in" file extension is internally added to this input file before reading it. The application is launched by a set of Bourne shell scripts that uses a number of ASCII text files to configure the application. Besides the standard and error outputs the legacy code generates two files where GEMLCA sends the final results.

The challenge of this deployment was to test GEMLCA with a complex structured legacy code and with run-time constrains, such as the input file implicit name in the input command line parameter, and the requirement of a full path definition in it.

The GAMESS-UK Grid Service was published using the $GEMLCAJOBPATH dynamic variable in order to set the input command line parameter, i.e. $GEMLCAJOBPATH/c2001_a. This variable is replaced at the time of running each job by the location of the volatile job environment that the job uses. The final input parameter defined in the LCID was set as a non-command line input file with the extension "in", i.e. c2001_a.in. Therefore, this file is included in each job environment but not listed as command-line parameter, given that another restriction of the legacy code is that it only accepts one and only one command line parameter.

The legacy code was successfully deployed using two GEMLCA Resources loaded at the University of Westminster and Daresbury Laboratory and tested from the P-Grade Gird portal running at University of Westminster.

6.3 MultiBayes

MultiBayes is an application developed at the School of Animal and Microbial Sciences at University of Reading and used in the Phylogenetic Tree Construction. It generates a Monte Carlo Markov Chain sample of trees from DNA sequences of genes common to a group of species.

The serial version of MultiBayes, which consists of an executable dynamically linked binary that accepts an input file as a parameter and creates three output files with the results, was deployed as GEMLCA Grid Service.

This legacy code was not as restrictive as GAMESS-UK but it presented a challenge to GEMLCA concerning the synchronisation between the P-Grade Grid portal and the GEMLCA Resources at the time of getting results due to its large output files.

As a result, the P-Grade Grid portal workflow management post-script had to be tuned in order to cope with the Grid file transfer of results and their presentation.

The legacy code was successfully deployed at the University of Westminster as a GEMLCA Resource, and used from the P-Grade Grid portal also running at University of Westminster.

6.4 Legacy Code deployment restrictions

Testing GEMLCA by deploying legacy codes with different level of complexity and requirements led to a number of GEMLCA improvements that extended the list of programs to be deployed as GEMLCA Grid Services. The improvements were achieved modifying the GEMLCA core behaviour and adding new capabilities to the LCIDs.

Even after these improvements there are some constraints that have to be considered when selecting the legacy code to be deployed as GEMLCA Grid Service:

- The first and most important constraint is that the legacy code has to accept input parameters from the command line. Consequently, any legacy code with a user interface that accepts input data during the execution cannot be considered as a GEMLCA Grid Service.
- The legacy code has to write its results into a file or standard/error outputs in order to expose them to Grid users. Any other results stored in a different way, for example in external databases, could not be displayed. To avoid this problem a script should be attached to the legacy code program to write its output into local files.
- GEMLCA creates a volatile job environment where input files are uploaded before the legacy code is executed and output files are expected to be created. This environment is deleted when the Grid client destroys the job or when the Grid Service life-time expires. Another constraint to be considered is that the folder containing the input and output files have to be set dynamically. A legacy code with a predefined output file folder cannot be used because a Grid Service may be used by different users at the same time producing problems in the creation of output files.
- When an input parameter name is related to an output parameter, and a Grid client changes the input parameter name, it has to modify the output parameter name in order to let the portal know its new name.

7 Conclusions

In this paper the deployment of three legacy code applications as Grid Services using GEMLCA have been described. All deployments had different challenges that produced as a result, on top of the deployment of these programs, several improvements in GEMLCA. These changes enhanced the Grid architecture in order to guarantee its main objective, the deployment of legacy code without changing the program code, and also to increase the number of legacy codes programs to be accepted.

This exercise also produced a list of restriction that have to be considered when selecting a program to be deployed in GEMLCA. These restrictions are basically forced

by the multi-user environment that has to be considered in Grid computing, and also the Grid object-oriented approach that does not make a full interactive program the best option for deployment.

Finally, with these examples, we demonstrated that the Grid environment composed of the integration of GEMLCA with the P-GRADE Grid portal enables Grid users to deploy legacy code applications, and to use them as Grid Services through a high-level user-friendly environment.

References

1. I. Foster, C. Kesselman, J. M. Nick, S. Tuecke. The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration. 2002. http://www.globus.org/research/papers/ogsa.pdf
2. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke. From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring and Evolution Version 1.1 May, 2004, http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf
3. P. Kacsuk , A. Goyeneche , T. Delaitre , T. Kiss , Z. Farkas , T. Boczko: High-level Grid Application Environment to Use Legacy Codes as OGSA Grid Services, 5th IEEE/ACM International Workshop on Grid Computing, November 8, 2004, Pittsburgh, USA
4. D. Kuebler, W. Eibach: Adapting legacy applications as Web services, IBM Developer Works, http://www-106.ibm.com/developerworks/webservices/
5. Y. Huang *et al*., "Wrapping Legacy Codes for Grid-Based Applications", Proceedings of the 17th International Parallel and Distributed Processing Symposium, workshop on Java for HPC), 22-26 April 2003, Nice, France. ISBN 0-7695-1926-1
6. T. Bodhuin, and M. Tortorella, "Using Grid Technologies for Web-enabling Legacy Systems", Proceedings of the Software Technology and Engineering Practice (STEP), Software Analysis and Maintenance: Practices, Tools, Interoperability workshop September 19-21, 2003, Amsterdam, The Netherlands.
7. B. Balis, M. Bubak, and M. Wegiel, "A Framework for Migration from Legacy Software to Grid Services", Cracow Grid Workshop '03, Cracow, Poland, December 2003.
8. T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G.Z.Terstyanszky and S.C. Winter. GEMLCA: Grid Execution Management for Legacy Code Architecture Design., Conf. Proc. of the 30th EUROMICRO conference, Special Session on Advances in Web Computing, August, 2004, Rennes, France,
9. Condor DAGman, http://www.cs.wisc.edu/condor/dagman/
10. A Gougoulis, G. Terstyanszky, P Kacsuk, S C Winter: Creating Scalable Traffic Simulation on Clusters, PDP2004 Conf. Proceedings of the 12th EuroMicro Conference on Parallel and Distributed and Network-Based Processing, La Coruna, Spain 11-13th February, 2004.
11. GAMESS-UK, CCLRC, http://www.cse.clrc.ac.uk/qcg/gamess-uk/
12. MultiBayes, University of Reading, http://www.ams.rdg.ac.uk/