Experiences with Publishing and Executing Parallel Legacy Code using an OGSI Grid Service

T. Delaitre, A. Goyeneche, D. Igbe, P. Kacsuk, T. Kiss, P. Maselino, K. Sajadah,
G. Terstyanszky, N. Weingarten, S.C. Winter
Centre for Parallel Computing School of Computer Science,
University of Westminster,115 New Cavendish Street, London, W1W 6UW.
Email: testbed-discuss@cpc.wmin.ac.uk

Abstract

In order to deploy existing scientific or business applications on a service-based Grid, these legacy code programs have to be transformed into Grid services. GEMLCA (Grid Execution Management for Legacy Code Architecture) represents a solution to make available and execute legacy applications through an OGSI Grid service without re-engineering the original code. GEMLCA is a client front-end OGSI Grid service layer that offers a number of interfaces to submit and check the status of computational jobs, and get the results back. This paper describes the GEMLCA architecture and life-cycle, and outlines the current state of implementation.

1. Introduction

There is a need in both scientific and business applications to offer existing legacy programs as Grid services. However, legacy code applications have been developed and implemented to run on a single computer or on a computer cluster and do not offer a set of OGSA compliant interfaces in order to be published and made available as a Grid Service. One approach to achieve this, is to re-engineer the legacy code, which in many cases implies significant efforts. Another solution to make available existing legacy code programs as Grid services without having to re-engineer them is supported by GEMLCA [1].

Section 2 of this paper overviews previous efforts to use legacy code in Grid environments and makes clear the innovation of our approach. Section 3 describes the GEMLCA concept and its implementation in a GT3 Grid environment. Section 4 shows how GEMLCA was integrated with the P-GRADE portal in order to create a convenient, user-friendly, high-level Grid application environment in which end-users can easily access any legacy code as a Grid service and can include it into complex workflow applications. Finally, section 5 describes how existing legacy codes, the Manhattan Road Map Generator and the MadCity Urban Traffic Simulator, exposed as Grid services by GEMLCA, were used in order to create a workflow.

2. Related works to use legacy code in Grid systems

Many large industrial and scientific applications are available today that were written well before Grid computing or service-oriented architectures appeared. To integrate these legacy code programs into service-oriented Grid architectures with the smallest possible effort and best performance, is a crucial point in more 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 the general framework to transform legacy applications into Web services outlined in [2], and use Java wrapping in order to generate stubs automatically. One example for this is presented in [3], where the authors describe a semi-automatic conversion of legacy C code into Java using JNI (Java Native Interface). After wrapping the native C application with the JACAW (Java-C Automatic Wrapper) tool MEDLI (MEdiation of Data and Legacy Code Interface) is used for data mapping in order to make the code available as part of a Grid workflow.

Different approaches from wrapping are presented in [4] and [5] but unfortunately these solutions only describe the principles and do not give a generic tool to do the automatic conversion.

Compared to Java wrapping GEMLCA is based

on a different principle. It offers a front-end Grid service layer that communicates with the client in order to pass input and output parameters, and contacts a local job manager through Globus MMJFS (Master Managed Job Factory Service) to submit the legacy computational job. To deploy a legacy application as a Grid service there is no need for the source code and not even for the C header files as in case of JACAW. The user only has to describe the legacy parameters in a pre-defined XML format. The legacy code can be written in any programming languages and can be not only a sequential but also a parallel PVM or MPI code that uses a job manager like Condor and where wrapping can be difficult. Current implementation of GEMLCA is based on GT3 but the architecture itself is more generic and can be easily adapted to other service-oriented approaches like WSRF or a pure Web services based solution.

Besides substantial advantages offered by GEMLCA it is also important to note that, as most of the other solutions, it supports decomposable or semi-decomposable software systems where the business logic and data model components can be separated from the user interface. The former can then be transformed into a Grid service using GEMLCA, and the latter have to be re-implemented, for example as part of a Grid portal. An approach to deal with non-decomposable legacy programs is described in [4] using screen proxies and redirecting input/output calls. However, this solution is language dependant and requires modification of the original code.

3. GEMLCA

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

GEMLCA conceptual architecture is shown in Figure 1.

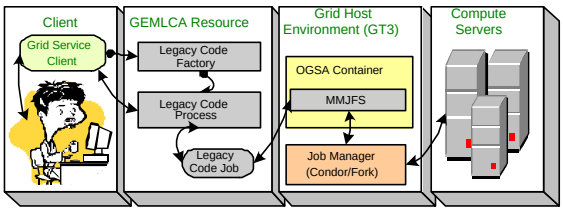

Figure 1. GEMLCA conceptual architecture.

In order to access a legacy code program, the user executes 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.

A GEMLCA resource 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. This architecture can be deployed in several user containers or Tomcat application contexts.

GEMLCA has been designed [1] 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 section of the architecture that deals with each legacy code environment and their instances as Grid legacy code processes and jobs. The final layer, *backend* is related to the Grid middleware where the architecture is being deployed. Current implementation is based on GT3 but this layer can be updated to any new standard, such as WSRF. Detailed explanation of the architecture can be found in [1].

3.1 Legacy Code deployment

As it was described in Section 2, most of the current solutions to expose legacy code programs as Grid services require access to the source code.

Mkdir Legacy Code exposed as a Grid Service
Folder : /../.gemlca/legacycodes/mkdir
Content : i) mkdir binary or link ii) config.xml

Legacy Code Interface Description Fil **e**: config.xml

```
<?xml version="1.0"?>
<!DOCTYPE GLCEnvironment "gemlcaconfig.dtd">
<GLCEnvironment
    id="mkdir" executable="LINUX/mkdir" jobManager="Fork"
    maximumJob="11" minimumProcessors="1"
    maximumProcessors="1" universe="PVM"
>
<Description >Unix mkdir program</ Description >
 <GLCParameters >
   <Parameter name="-p" friendlyName="Folder to be created"
        fixed="No" inputOutput="Input" order="0"
        mandatory="No" fileCommandline="Commandline">
      <initialValue> </initialValue>
   </Parameter>
   </ GLCParameters >
</ GLCEnvironment >
```

Figure 2. Legacy Code Interface Description File.

Alternatively in GEMLCA, the only extra effort to be done is to create a Legacy Code Interface Description File (LCID) in XML. This LCID file shown in Figure 2 consists of three sections. The GLCEnvironment section contains the name of the legacy code and its main binary file, job manager

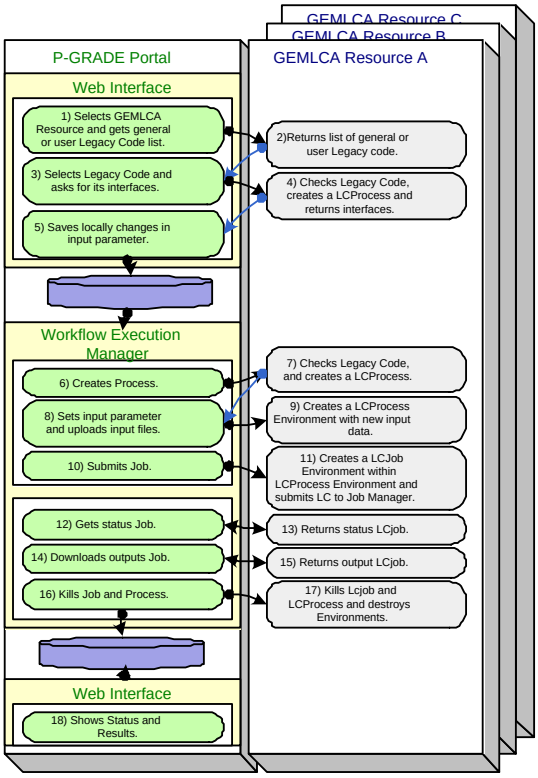

Figure 3. P-GRADE Portal and GEMLCA resource interaction.

(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 next section describes the legacy code in simple text format, and finally the parameter section exposes the list of parameters, each one describing its name, friendly name, input or output, order, mandatory, file or command line, fixed, and regular expression to be used as input validation.

4. The Integrated GEMLCA/P-GRADE Portal

Grid portals are essential facilities to make Grid applications available from a Web browser. By connecting GEMLCA to a Grid portal such as the P-Grade portal [7], legacy code applications are available as Grid services through the Web.

The P-GRADE portal integration with GEMLCA has been done through the creation and use of a number of Grid clients. The integration was divided into three parts, as shown in Figure 3:

1. In order to allow the portal user to create a workflow composed of GEMLCA Grid services, a new type of node, called GEMLCA, was added to the selection of available components. In order to create one of these nodes the Grid GEMLCA client, embedded in the portal, allows the selection of a GEMLCA resource and a legacy code from the list returned. A popup window is displayed in order to change or set any input parameter and upload any input file to the portal server. This information is temporally saved in the portal server until the workflow manager uses it.

2. Once the workflow is completed and saved, the workflow manager, Condor DAGman, is called in order to submit GEMLCA jobs. In GT2 Grids Condor DAGMan generates Condor-G job submissions. In case of GEMLCA, the DAGMan's PRE, POST and job submission scripts were modified in order to generate GEMLCA job submission. The PRE-script has been changed in order to call a GEMLCA client that contacts the GEMLCA resource. It creates an instance of the legacy code process returning a Grid service Handle (GSH). Such GSH is used by another GEMLCA client for setting any new parameters and also uploading input files using GridFtp. Finally, the GEMLCA client can submit the job.

3. The GEMLCA client is also used for checking the status of the legacy code process and jobs. The output files of legacy codes are downloaded by the portal server and made available to the user. Alternatively, the output file will be transfered into the next legacy code environment of the workflow if it is needed.

5. GEMLCA implementation and results

In order to demonstrate the integrated GEMLCA P-Grade portal solution, a workflow for analysing Urban car traffic on road was created as shown in Figure 4.

The workflow consists of three components: a Manhattan road network *generator*, a traffic *simulator*, called MadCity, and an analyser.

The Manhattan legacy code, is an application to generate MadCity compatible network and turn input-files. The output of the first component is used as an input to MadCity [8] traffic simulator. MadCity is a discrete time-based traffic simulator that simulates traffic on a road network. Each set of generators and simulators run in parallel on the UK OGSA testbed sites. After completing the simulations, the generated trace files are input to an

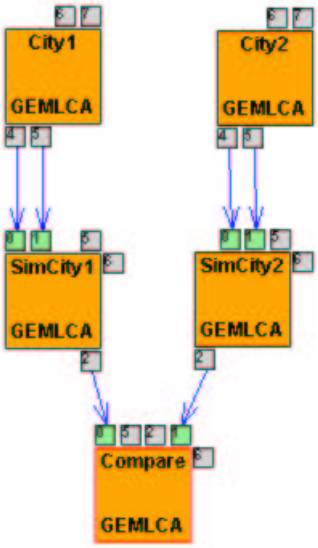

Figure 4. Workflow of Urban traffic simulations.

analyser that compares the traffic density of roads and illustrates the results on a graph.

The described workflow was successfully created by the portal on Westminster and executed on the UK OGSA testbed sites.

6. Conclusion

GEMLCA represents a new approach to deploy legacy code applications as Grid services without modifying the source code. The user only has to create an XML-based Legacy Code Interface Description File and GEMLCA enables the legacy code application to be run from a Grid service client. The enhanced P-GRADE Portal is now capable not only to connecting to GT2 resources but also enables to include legacy codes deployed as Grid services conforming to OGSA as part of complex Grid workflows.

The P-GRADE/GEMLCA integrated portal solution was demonstrated by deploying a legacy traffic simulator, a road traffic generator, and an analyser as Grid services. Each of these were connected by a worflow and executed on the UK OGSA testbed sites.

Future work includes the support of other service-oriented Grid architectures like WSRF and pure Web services. It is also envisaged to develop plug-ins for application specific visualisers to the P-GRADE portal.

References

[1] T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G.Z.Terstyanszky and S.C. Winter. GEMLCA: Grid Execution Management for Legacy Code Architecture Design. To appear in Conf. Proc. of the 30th EUROMICRO conference, Special Session on Advances in Web Computing, August, 2004, Rennes, France.

[2] D. Kuebler, and W. Eibach, "Adapting legacy applications as Web services", IBM DeveloperWorks.

[3] Y. Huang *et al.*, "Wrapping Legacy Codes for Grid-Based Applications", in 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

[4] T. Bodhuin, and M. Tortorella, "Using Grid Technologies for Web-enabling Legacy Systems", in Proceedings of the Software Technology and Engineering Practice (STEP), The workshop Software Analysis and Maintenance: Practices, Tools, Interoperability, September 19-21, 2003, Amsterdam, The Netherlands.

[5] B. Balis, M. Bubak, and M. Wegiel, "A Framework for Migration from Legacy Software to Grid Services", In Cracow Grid Workshop '03, Cracow, Poland, December 2003.

[6] T. Delaitre, A.Goyeneche, T.Kiss, G.Z. Terstyanszky, N. Weingarten, P. Maselino, A. Gourgoulis, S.C. Winter, Traffic Simulation in P-GRADE as a Grid Service , To Appear in Conf. Proc. of the DAPSYS 2004 Conference, September 19-22, 2004, Budapest, Hungary.

[7] Cs. Nemeth, G. Dozsa, R. Lovas and P. Kacsuk, The P-GRADE Grid Portal, Conference proceedings of the 2004 International Conference on Computational Science and its Applications. Editors: A. Lagana et al. LNCS 3044, pp. 10-19, S. Maria degli Angeli, Assisi(PG), Italy, 2004.

[8] A. Gourgoulis, G. Terstyansky, P. Kacsuk, S.C. Winter, Creating Scalable Traffic Simulation on Clusters. PDP2004. Conference Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network based Processing, La Coruna, Spain, 11-13th February, 2004.