High-level grid application environment to use legacy codes as OGSA grid services.

Peter Kacsuk[1,2]
Ariel Goyeneche[1]
Thierry Delaitre[1]
Tamas Kiss[1]
Zoltan Farkas[2]
T. Boczko[2]

[1]Cavendish School of Computer Science, University of Westminster
[2]MTA SZTAKI Lab. Of Parallel & Distributed Systems, Budapest, Hungary

High-level Grid Application Environment to Use Legacy Codes as OGSA Grid Services*

P. Kacsuk[1,2], A. Goyeneche[1], T. Delaitre[1], T. Kiss[1], Z. Farkas[2], T. Boczko[2]
[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
Web sites: http://www.cpc.wmin.ac.uk/gemlca & http://www.lpds.sztaki.hu/pgportal
Email: gemlca-discuss@cpc.wmin.ac.uk

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 that is not accessible as Grid services. The paper describes a new approach (GEMLCA: Grid Execution Management for Legacy Code Architecture) to deploy legacy codes as Grid services without modifying the original code. Moreover, we show a workflow execution oriented Grid portal technology (P-GRADE portal) by which such legacy code based Grid services can be applied in complex business processes. GEMLCA has been implemented as GT-3 services but can be easily ported into the new WSRF Grid standards.

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 [1] and WSRF [2] standards. On the other hand relatively little attention has been paid to how the end-users can survive in the rapidly changing world of Grid generations. The primary goal of our research is to construct a high-level Grid application environment where the end-users can:

- Easily and conveniently create complex Grid applications.
- Apply any legacy code as OGSA compliant Grid service when they create Grid applications.
- Migrate from GT2 Grids to OGSA-based Grids with minimal efforts.

In an ideal OGSA-based Grid environment users are able to access predefined Grid services through a high-level user-friendly Grid portal. More than that, users are not only capable of using such services but they can dynamically create and deploy new services in a convenient and efficient way. All these mentioned Grid services can be either specifically designed Grid services or legacy programs that automatically deployed as Grid services when desired. In order to achieve this ideal scenario and to provide the high-level Grid application environment mentioned above the following tasks should be solved:

- A method is necessary by which legacy code can automatically be deployed as a Grid service.
- Such Grid services should be accessible and usable in workflows supported by Grid portals.
- Such Grid portals should be derived from existing Globus Toolkit version 2 [3] (GT2) portals in order to minimize the users' learning curve when moving from GT2 Grids to OGSA Grids.

In the current paper, we show how these tasks were solved in the framework of the UK OGSA testbed project. First, we introduce the GEMLCA (Grid Execution Management for Legacy Code Architecture) concept by which legacy code written in any language (Fortran, C, Java, etc.) can easily be deployed as an OGSA Grid service without any user effort. Second, we show that the integration of GEMLCA with the P-GRADE Grid portal (a GT2 portal) results in an OGSA-based Grid portal that requires minor changes at the graphical user interface. In order to achieve

* The work presented in this paper is supported by a UK EPSRC funded project (Grant No.: GR/S77509/01) and by two Hungarian projects (Grant No.: IHM 40.0346/2/2004 and OTKA T042459).

this, GEMLCA should be installed, configured, and published on the Grid middleware server and the GEMLCA clients on the Grid portal servers. After that, any legacy code deployed in GEMLCA can be accessed by authorized users as OGSA Grid service through the Grid portal. As a consequence, legacy codes (either sequential or parallel ones) can be used as workflow components to build complex Grid applications.

Currently, one of the main obstacles of the industrial take-up of Grid technology is the existence of a large amount of legacy code that is not accessible as Grid services. The GEMLCA concept and its integration with the Grid portal technology eliminates this problem and can lead to a breakthrough in the establishment of business Grids. Of course, this approach will stretch even the usability of scientific Grids where most of the codes are written in Fortran and now all these applications will be accessible as Grid services.

Section 2 overviews previous efforts to use legacy code in Grid environments and make clear the innovation of our approach. Section 3 describes the GEMLCA concept and its implementation in a GT3 Grid environment. Section 4 introduces a workflow-oriented Grid portal, called the P-GRADE Grid portal and Section 5 shows how these two systems, GEMLCA and P-GRADE portal were integrated in order to create a convenient, user-friendly, high-level Grid application environment in which end-users can easily access any legacy code as Grid service and can include it into complex workflow applications. Finally, section 6 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 complex 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 [4], and use Java wrapping in order to generate stubs automatically. One example for this is presented in [5], 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 [6] and [7] 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 [3] (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 that in the current GEMLCA version has to be done manually. However, the next release of the architecture will automate this process. 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 [8] and where wrapping can be difficult. The 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 [6] 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.

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.

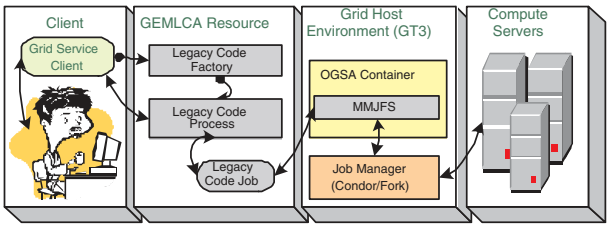

Figure 1. GEMLCA conceptual architecture.

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 [9] 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 [3] but this layer can be quickly updated to any new standard, such as WSRF [2]. Detailed explanation of the architecture can be found in [9].

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.

On the other hand 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 (Condor [8] 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. Work is currently undergoing to automate the process of creating the LCID file and this way making it even easier for the end user to deploy legacy applications as Grid services.

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.

3.2. GEMLCA security and multi-user environment

GEMLCA uses the Grid Security Infrastructure (GSI) [10] to enable user authentication and to support secure communication over a Grid network. A GEMLCA client needs to sign its credential and also to work in full delegation mode in order to allow the architecture to work on its behalf. GEMLCA has two levels of authorisation: the first level is given by the grid-map file mechanism [11]. If the user is correctly mapped, the second level comes into play which is given by a set of legacy codes that a Grid Client is allowed to use. This set is composed of a combination of a general list of legacy codes, available to anyone using a specific GEMLCA resource, and a user mapped list of legacy codes, only available to Grid clients mapped to a local user by the grid-map file mechanism.

GEMLCA administers the internal behaviour of legacy codes taking into account the requirements of input files and output files in a multi-user environment, and also complies with the security restrictions of the operating systems where the architecture is running. In order to do that, GEMLCA is using GEMLCA itself in a protected mode composed of a set of *system legacy codes* in order to create and destroy a unique process and job stateful environment only reachable by the local user mapped by the grid-map file mechanism.

3.3. Grid Client interaction with GEMLCA interfaces

Figure 3 shows the interaction between a Grid client and a GEMLCA recourse exposing Legacy code programs.

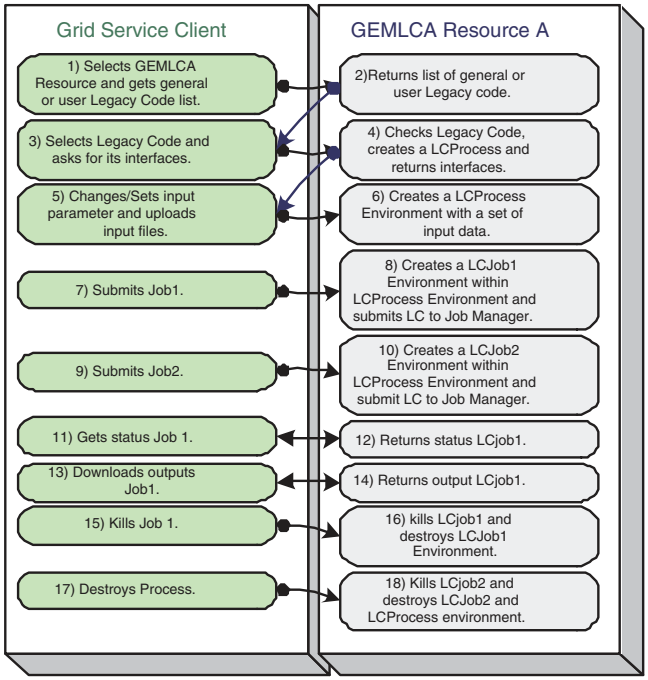

Figure 3. Grid Client and GEMLCA resource interaction.

In this communication, a unique set of stubs is used by the Grid client in order to interact with any exposed legacy code. When a client selects a legacy code, GEMLCA creates a LCProcess and its stateful environment using the default values, if any, for each input and output parameter. Each LCProcess can be customized to accept a maximum number of LCJobs to be submitted from its interfaces. GEMLCA also provides a set of interfaces for the Grid client in order to query and retrieve the LCProcess status, the list and number of LCJobs in each LCProcess, and the output results of each job. Finally, a particular LCJob can be killed or a LCProcess destroyed.

4. P-GRADE Portal

One of the most important applications of Grid systems is the solution of complex problems based on the workflow concept where the user can connect existing program components in a workflow graph. Nodes of the graph are typically executed in GT2 Grids as jobs and the arcs of the graph represent the necessary file transfer operations among the component jobs. Such workflow systems should be embedded into Grid portals in order to hide from the end-user the low level details of job execution and file transfers in Grid systems. Moreover such a Grid portal should be portable to various Grid systems in order to make the workflows portable. The ten years of Grid systems resulted in three generations of Grid architectures making it extremely difficult for the end-user to adapt applications to the rapidly changing Grid environments.

The P-GRADE portal was designed to fulfil all these requirements. It is a workflow-oriented Grid portal where the main goal is to enable users to manage the whole lifecycle of creating and executing a complex application in the Grid: graphically editing workflows from various types of existing components (sequential, MPI, etc.), submitting jobs relying on Grid-credentials and analysing the monitored trace-data by visualisation facilities. The P-GRADE portal currently supports the following functionalities (Figure 4):

- Grid certificate management,
- Setting the Grid environment
- Creation, modification and execution of workflow applications on grid resources
- Visualisation of workflow progress as well as each component job.

All these functionalities are highly portable among various Grid systems. Portability and fast development was also supported by the application of the GridSphere [12] Grid portal development framework as an implementation environment.

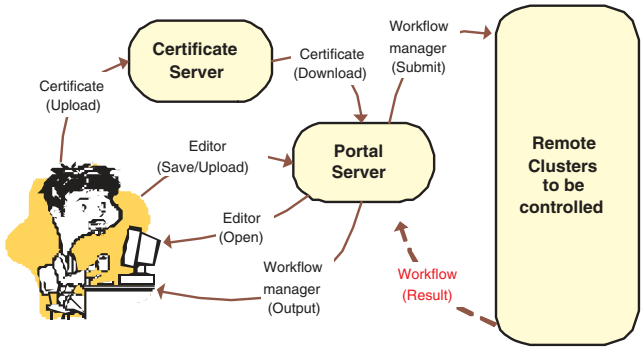

Figure 4. The P-GRADE portal functionalities.

Grid certificate management. The Credential Manager portlet allows the portal user to download Grid certificates (proxy credentials) from a MyProxy server [13] (MPS) as shown in Figure 4. A user may have more than one proxy from different MPSs and any proxy can be selected for usage. The user can view the details of a proxy, including its lifetime. This credential will

be passed to other portlets of the portal. The proxies, that are no longer needed, can be deleted. Renewing an expired proxy requires connecting the MPS again. If there is no MPS holding appropriate credential, the user may upload his private key and certificate to an MPS, and later this MPS can create a proxy by means of the portal setting the Grid environment. The other prerequisite to use the portal is the setting of the Grid resources the user has access and would like to use by the portal. At this stage the URL of the Grid resources and the applicable job managers should be given by the user. If Grid broker is available, like in the LCG-2 Grid, the identification of the Grid broker is enough at this point.

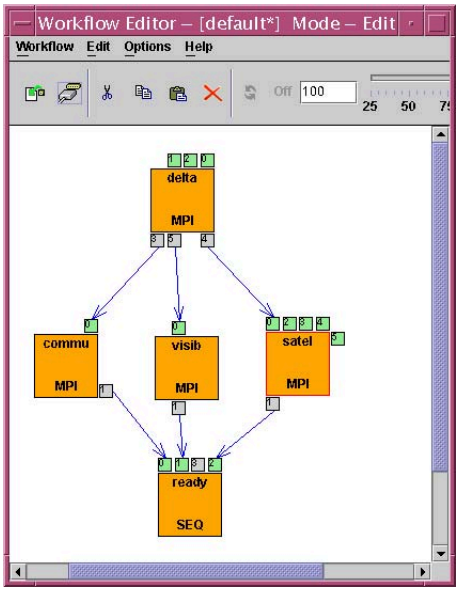

Figure 5. Workflow of a meteorology application.

Workflow editing. The workflows can be graphically created at the client machine by the Workflow Editor written as a Java Web-Start application. A simple workflow used in a real-life meteorology application [14] is shown in Figure 5. Large squares represent jobs and small squares represent the input and output files. Definition of a job requires the identification of the job (e.g. delta), the location of the executable code and its type (SEQ, MPI, etc.). If there is no broker in the connected Grid system, the user should explicitly define the Grid resource where the job is to be executed. Files are identified by their logical name and location. It should be also defined if the file is a permanent one or used only temporarily. In the latter case when the job using the temporary file has been finished, the file is automatically removed from the Grid. After creating the workflow on the client machine it should be uploaded to the portal server machine.

Workflow management. The uploaded workflows are managed by the Workflow Manager portlet that provides the following functionalities:

a. Storing, updating and visualizing the status of the workflows and their component jobs
b. Submitting workflows when the user requests it.
c. Taking care of the necessary file transfers among the workflow jobs (executed on different Grid resources).
d. Detach and attach running workflows.
e. Delivering and storing the result files of the workflow execution.
f. Showing workflow execution visualization and individual job execution visualization if the job is a parallel program.

The Workflow Manager portlet collaborates with the Condor DAGMan [15] system that is responsible for selecting the next executable job of the workflow. Condor DAGMan provides a PRE-script facility to do job management before a job is actually started as a Condor-G [16] job. This PRE-script can be written according to the actual needs in different Grid environments. If there is no Grid broker in the connected Grid, the user specifies at workflow edit time the Grid site where the job should be run. If there is a Grid broker, the Workflow Manager portlet uses the PRE-script facility to contact the Grid broker and asks the selection of an appropriate Grid site. Once the Grid site is selected (in either way) the Workflow Manager portlet uses again the PRE-script facility to transfer the necessary input files by GridFTP into the selected Grid resource. Then the job is submitted as a Condor-G job to the Globus-2 gatekeeper of the selected Grid site. This is the way how the portal Workflow Manager portlet collaborates with a GT2 Grid environment. The P-GRADE portal was actually connected so far to three different GT2 based Grid systems: Hungarian Supercomputing Grid, LCG-2, and the EU GridLab testbed.

5. The Integrated GEMLCA/P-GRADE Portal

In Section 3 and 4 we have described two powerful tools by which the Grid can be made conveniently accessible by end-users. GEMLCA gives the possibility to apply any existing legacy code as a Grid service. The workflow editor of P-GRADE portal enables the connection of component jobs into complex workflows by an easy-to-use graphical environment. The Workflow Manager of P-GRADE portal takes care of executing such workflows on various GT2 Grid systems in a user-transparent way. As written in the Introduction, our goal was to enable end-users to apply legacy code

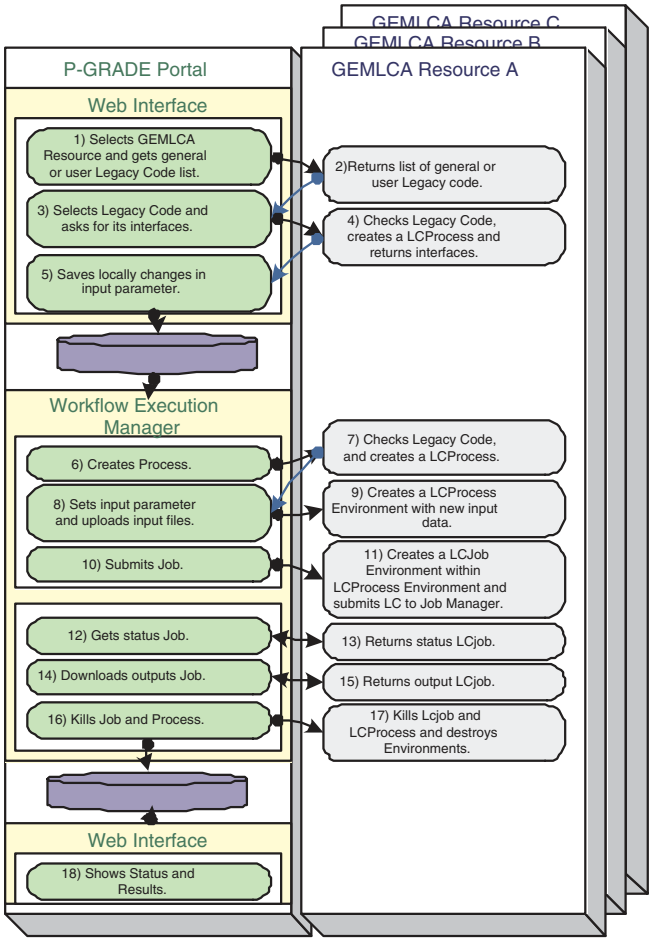

Figure 6. P-GRADE Portal and GEMLCA resource interaction.

as OGSA Grid services in workflows and to move from GT2 Grids to OGSA Grids with the least possible effort. In order to achieve this, we integrated the P-GRADE portal with GEMLCA where the user can construct workflows from legacy codes deployed as OGSA Grid services. The user interface as well as the internal structure of the P-GRADE portal required only minor changes that are subjects of this section.

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 6:

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.

6. Applications of the GEMLCA technology

Many legacy code programs are available exposing scientific, business and industry applications that need to be integrated into new service oriented Grid architectures. As GEMLCA offers this integration without any modification in the original code, it has huge potential in several application areas.

A new, Grid service based electronic marketplace architecture has already been introduced in [17] that uses GEMLCA in order to transform an existing marketplace and utilise the advantages offered by the Grid solution.

Also in [18] the concept of transforming a legacy traffic simulator into a Grid service and to visualise the results in the P-GRADE portal was described. Work is currently undergoing in both areas to fully implement the concepts. In this paper we explain how a complex workflow was created using existing legacy programs in order to demonstrate the capabilities of GEMLCA and its integration with the P-GRADE portal and workflow solutions.

6.1. Urban car traffic simulation

GEMLCA gives the possibility to deploy existing legacy code as a Grid service. The workflow editor of P-GRADE

portal enables the connection of components into complex workflows by an easy-to-use Web-based graphical environment and the workflow manager takes care of executing such workflows on various Grid systems in a user-transparent way.

A complex workflow has been created as a case study in order to test this High-level Grid Toolkit Environment.

6.2. Workflow description

The workflow consists of three types of legacy code components:

1. The Manhattan legacy code, is an application to generate MadCity compatible network and turn input-files. The MadCity network file is a sequence of numbers, representing a road topology, of a real road network. The number of columns, rows, unit width and unit height can be set as input parameters. The MadCity turn file, is a sequence of numbers representing the junction manouveres available in a given road network. Traffic light details are included in this input file.
2. MadCity [19] is a discrete time-based traffic simulator that was developed by the research team of the Centre for Parallel Computing at the University of Westminster. It simulates traffic on a road network and shows how individual vehicles behave on roads and at junctions. The simulator of MadCity models the movement of vehicles using the input road network file. After completing the simulation, the simulator creates a macroscopic trace file.
3. And a traffic density analyzer which compares the traffic congestion of several simulations of a given city and presents a graphical analysis.

In order to meet our test objective, the workflow shown in Figure 7 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 deployed. The first GEMLCA resource is installed at the University of Westminster (UK) and runs the Manhattan road network generator (Job0), one traffic simulator instance (Job3) and the final traffic density analyzer (Job6). 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) where the traffic simulator is deployed. One instance of the simulator is executed on each of these sites, respectively Job1, Job2, Job5 and Job4.

The MadCity network file and the turn file are used as input to each traffic simulator instance. In order to have a different behaviour in each of these instances, each one was set with different initial number of cars per street junction, one of the input parameter of the program. The output file

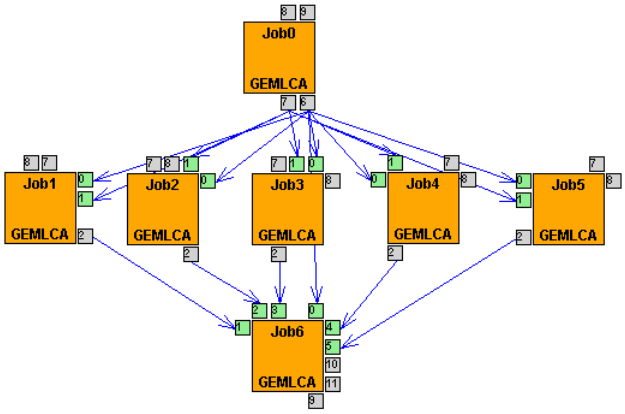

Figure 7. Complex workflow for analysing road traffic.

of each traffic simulation is used as input file to the Traffic density analyzer. The described workflow was successfully created and executed by the P-GRADE portal installed at the university of Westminster. The execution of the workflow can be shown in Figure 8.

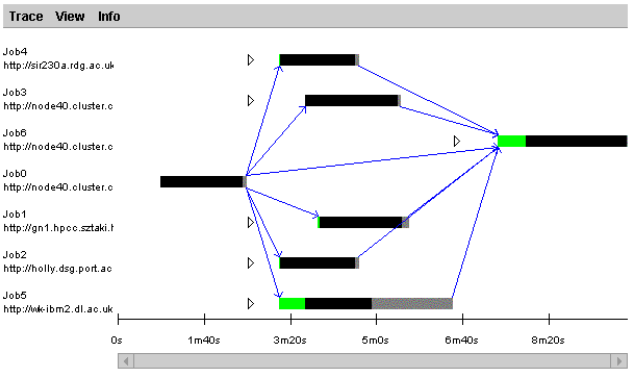

Figure 8. Visualisation of the workflow execution.

7. Conclusion

The more widespread take-up of Grid technology requires high-level Grid application environments where users can easily create complex Grid workflows including different Grid enabled applications and also legacy code programs. This paper described how the P-GRADE por-

tal and workflow solutions were integrated with GEMLCA in order to fulfil these requirements. 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 Manhattan road traffic generator, several instances of the legacy traffic simulator and a traffic density analyzer into Grid services. All these legacy codes were executed from a single workflow and the execution was visualised by the portal.

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

For more information, please visit the following web sites: `http://www.cpc.wmin.ac.uk/gemlca` and `http://www.lpds.sztaki.hu/pgportal`

8. Acknowledgements

The authors wish to acknowledge the support and contributions of Damian Igbe, Agathocles Gourgoulis and Noam Weingarten in the traffic simulation aspects.

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] Globus Team, Globus Toolkit, `http://www.globus.org`

[4] D. Kuebler, and W. Eibach, "Adapting legacy applications as Web services", IBM DeveloperWorks, `http://www-106.ibm.com/developerworks/webservices/library/ws-legacy/`

[5] Y. Huang, I. Taylor, D. Walker, and R. Davies, "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

[6] 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, `http://www.bauhaus-stuttgart.de/sam/bodhuin.pdf`

[7] 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, `http://www.icsr.agh.edu.pl/~balis/bib/legacy-cgw03.pdf`

[8] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003

[9] 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.

[10] J. Gawor, S. Meder, F. Siebenlist, V. Welch, GT3 Grid Security Infrastructure Overview, February 2004. `http://www-unix.globus.org/security/gt3-security-overview.doc`

[11] L. Ramakrishnan. Writing secure grid services using Globus Toolkit 3.0. September 2003, `http://www-106.ibm.com/developerworks/grid/library/gr-secserv.html`

[12] GridSphere homepage. http://www.gridsphere.org/

[13] J. Novotny, S. Tuecke, V. Welch: An Online Credential Repository for the Grid: MyProxy. Proceedings of the 10th IEEE Intl. Symp. on High Performance Distributed Computing, 2001.

[14] R. Lovas, et al.: Application of P-GRADE Development Environment in Meteorology. Proc. of DAPSYS'2002, Linz, pp. 30-37, 2002.

[15] Condor DAGman, `http://www.cs.wisc.edu/condor/dagman/`

[16] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids, Journal of Cluster Computing volume 5, pages 237-246, 2002

[17] L. Kacsukne Bruckner, T.Kiss. Grid Solution for e-Marketplaces Integrated with Logistics , To Appear in Conf. Proc. of the DAPSYS 2004 Conference, September 19-22, 2004, Budapest, Hungary.

[18] 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.

[19] 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.