Legacy Code Support for Production Grids.

Tamas Kiss[1]
Gabor Terstyanszky[1]
Gabor Kecskemeti[1]
Szabolcs Illes[1]
Thierry Delaitre[1]
Stephen Winter[1]
Peter K. Kacsuk[2]
Gergely Sipos[2]

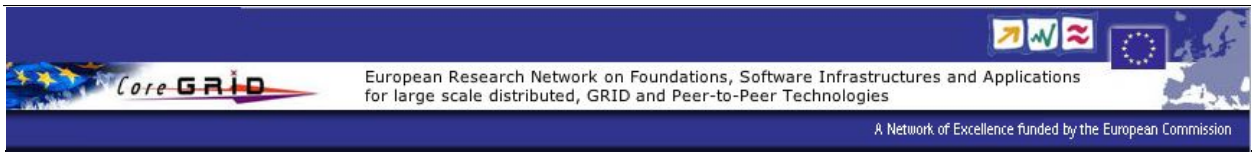
[1] School of Informatics, University of Westminster
[2] MTA SZTAKI, 1111 Kende u. 13, Budapest, Hungary

This is a reproduction of CoreGRID Technical Report Number TR-0011, June 6, 2005 and is reprinted here with permission.

The report is available on the CoreGRID website, at:

http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0011.pdf

---



Legacy Code Support for Production Grids

T. Kiss, G. Terstyanszky*, G. Kecskemeti*, Sz. Illes*, T. Delaittre*, S. Winter*,*
*P. Kacsuk**, G. Sipos***
**Centre of Parallel Computing, University of Westminster,*
115 New Cavendish Street,
London W1W 6UW United Kingdom
e-mail: gemlca-discuss@cpc.wmin.ac.uk
***MTA SZTAKI*
1111 Kende u. 13
Budapest, Hungary

CoreGRID Technical Report
Number TR-0011
6th June 2005

Institute on Problem Solving Environment, Tools and GRID Systems

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Legacy Code Support for Production Grids

T. Kiss*, G. Terstyanszky*, G. Kecskemeti*, Sz. Illes*, T. Delaittre*, S. Winter*,
P. Kacsuk**, G. Sipos**
*Centre of Parallel Computing, University of Westminster,
115 New Cavendish Street,
London W1W 6UW United Kingdom
e-mail: gemlca-discuss@cpc.wmin.ac.uk
**MTA SZTAKI
1111 Kende u. 13
Budapest, Hungary

CoreGRID TR-0011

6th June 2005

Abstract

In order to improve reliability and to deal with the high complexity of existing middleware solutions, todays production Grid systems restrict the services to be deployed on their resources. On the other hand end-users require a wide range of value added services to fully utilize these resources. This paper describes a solution how legacy code support is offered as third party service for production Grids. The introduced solution, based on the Grid Execution Management for Legacy Code Architecture (GEMLCA), do not require the deployment of additional applications on the Grid resources, or any extra effort from Grid system administrators. The implemented solution was successfully connected to and demonstrated on the UK National Grid Service.

1 Introduction

The vision of Grid computing is to enable anyone to offer resources to be utilised by others via the network. This original aim, however, has not been fulfilled so far. Todays production Grid systems, like the EGEE Grid, the NorduGrid or the UK National Grid Service (NGS) apply very strict rules towards service providers, hence restricting the number of sites and resources in the Grid. The reason for this is the very high complexity to install and maintain existing Grid middleware solutions. In a production Grid environment strong guarantees are needed that system administrators keep the resources up and running. In order to offer reliable service only a limited range of software is allowed to be deployed on the resources.

On the other hand these Grid systems aim to serve a large and diverse user community with different needs and goals. These users require a wide range of tools in order to make it easier to create and run Grid-enabled applications. As system administrators are reluctant to install any software on the production Grid that could compromise reliability, the only way to make these utilities available for users is to offer them as third-party services. These services are running on external resources, maintained by external organisations, and they are not integral part of the production Grid system. However, users can freely select and utilise these additional services based on their requirements and experience with the service.

This previously described scenario was utilised to connect GEMLCA (Grid Execution Management for Legacy Code Architecture) [11] to the UK National Grid Service. GEMLCA enables legacy code programs written in any source language (Fortran, C, Java, etc.) to be easily deployed as a Grid Service without significant user effort. A

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

user-level understanding, describing the necessary input and output parameters and environmental values such as the number of processors or the job manager used, is all that is required to port the legacy application binary onto the Grid. GEMLCA does not require any modification of, or even access to, the original source code. The architecture is also integrated with the P-GRADE portal and workflow [13] solutions to offer a user friendly interface, and create complex applications including legacy and non-legacy components.

In order to connect GEMLCA to the NGS two main tasks have been completed:

- First, a portal server has been set up at University of Westminster running the P-GRADE Grid portal and offering access to the NGS resources for authenticated and authorised users. With the help of their Grid certificates and NGS accounts portal users can utilise NGS resources in a much more convenient and user-friendly way than previously.
- Second, the GEMLCA architecture has been redesigned in order to support the third-party service provider scenario. There is no need to install GEMLCA on any NGS resource. The architecture is deployed centrally on the portal server but still offers the same legacy code functionally as the original solution: users can easily deploy legacy applications as Grid services, can access these services from the portal interface, and can create, execute and visualise complex Grid workflows.

This paper describes two different scenarios how GEMLCA is redesigned in order to support a production Grid system. The first scenario supports the traditional job submission like task execution, and the second offers the legacy codes as pre-deployed services on the appropriate resources. In both cases GEMLCA runs on an external server, and neither compromise the reliability of the production Grid system, nor requires extra effort from the Grid system administrators. The service is transparent from the Grid operators point of view but offers essential functionality for the end-users.

2 The UK National Grid Service

The National Grid Service (NGS) is the UK production Grid operated by the Grid Operation Support Centre (GOSC). It offers a stable highly-available production quality Grid service to the UK research community providing compute and storage resources for users. The core NGS infrastructure consists of four cluster nodes at Cambridge, CCLRC-RAL, Leeds and Manchester, and two national High Performance Computing (HPC) services: HPCx and CSAR. NGS provides compute resources for compute Grid through compute clusters at Leeds and Oxford, and storage resources for data Grid through data clusters at CCLRC-RAL and Manchester. This core NGS infrastructure has recently been extended with two further Grid nodes at Bristol and Cardiff, and will be further extended by incorporating UK e-Science Centres through separate Service Level Agreements (SLA).

NGS is based on GT2 middleware. Its security is built on Globus Grid Security Infrastructure (GSI) [14], which supports authentication, authorization and single sign-on. NGS uses GridFTP to transfer input and output files to and from nodes, and Storage Resource Broker (RSB) [6] with OGSA-DAI [3] to provide access to data on NGS nodes. It uses the Globus Monitoring and Discovery Service (MDS) [7] to handle information of NGS nodes. Ganglia [12], Grid Integration Test Script (GITS) [4] and Nagios [2] are used to monitor both the NGS and its nodes. Nagios checks nodes and services while GITS monitors communication among NGS nodes. Ganglia collects and processes information provided by Nagios and GITS in order to generate NGS-level view.

NGS uses a centralised user registration model. Users have to obtain certificates and open accounts to be able to use any NGS service. The certificates are issued by the UK Core Programme Certification Authority (e-Science certificate) or by other CAs. NGS accounts are allocated from a central pool of generic user accounts to enable users to register with all NGS nodes at the same time. User management is based on Virtual Organisation Membership Service (VOMS) [1]. VOMS supports central management of user registration and authorisation taking into consideration local policies on resource access and usage.

3 Grid Execution Management for Legacy Code Architecture

The Grid computing environment requires special Grid enabled applications capable of utilising the underlying Grid middleware and infrastructure. Most Grid projects so far have either developed new applications from scratch, or

significantly re-engineered existing ones in order to be run on their platforms. However, as the Grid becomes commonplace in both scientific and industrial settings, the demand for porting a vast legacy of applications onto the new platform will emerge. 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. The Grid Execution Management for Legacy Code Architecture (GEMLCA) enables legacy code programs written in any source language (Fortran, C, Java, etc.) to be easily deployed as a Grid Service without significant user effort. In this chapter the original GEMLCA architecture is outlined. This architecture has been modified, as described in chapters 4 and 5, in order to create a centralised version for production Grids.

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 high-level GEMLCA conceptual architecture is represented on Figure 1.

As shown in the figure, there are four basic components in the architecture:

1. The *Compute Server* is a single or multiple processor computing system on which several legacy codes are already implemented and available. The goal of GEMLCA is to turn these legacy codes into Grid services that can be accessed by Grid users.
2. The *Grid Host Environment* implements a service-oriented OGSA-based Grid layer, such as GT3 or GT4. This layer is a pre-requisite for connecting the Compute Server into an OGSA-built Grid.
3. The *GEMLCA Resource* layer provides a set of Grid services which expose legacy codes as Grid services.
4. The fourth component is the *GEMLCA Client* that can be installed on any client machine through which a user would like to access the GEMLCA resources.

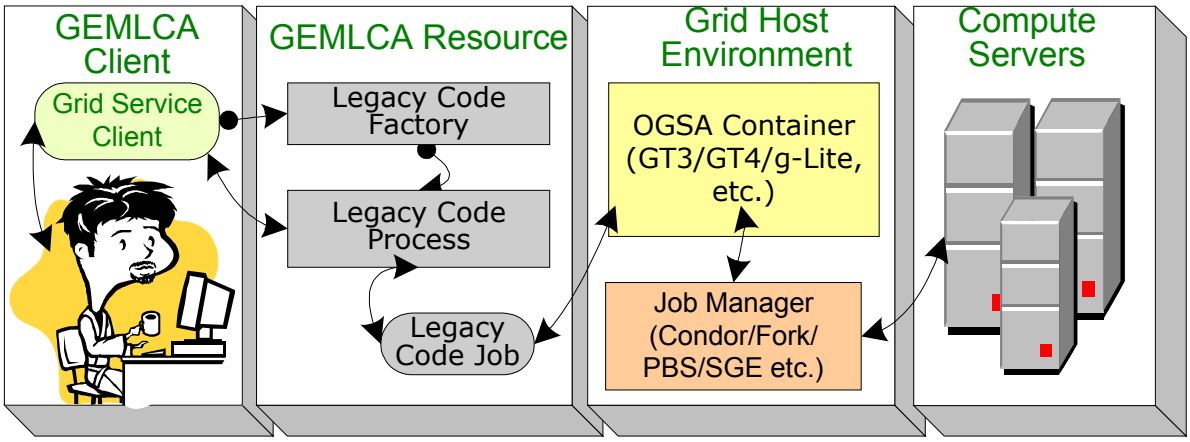


Figure 1: GEMLCA Conceptual Architecture

The novelty of the GEMLCA concept compared to other similar solutions like [10] or [5] is that it requires minimal effort from both Compute Server administrators and end-users of the Grid. The Compute Server administrator should install the GEMLCA Resource layer on top of an available OGSA layer (GT3/GT4). It is also their task to deploy existing legacy applications on the Compute Servers as Grid services, and to make them accessible for the whole Grid community. End-users do not have to do any installation or deployment work if a GEMLCA portal is available for the Grid and they only need those legacy code services that were previously deployed by the Compute Server administrators. In such a case end-users can immediately use all these legacy code services - provided they have access to the GEMLCA Grid resources. If they would like to deploy legacy code services on GEMLCA Grid resources they can do so, but these services cannot be accessed by other Grid users. As a last resort, if no GEMLCA portal is available for the Grid, a user must install the GEMLCA Client on their client machine. However, since it requires some IT skills to do this, it is recommended that a GEMLCA portal is installed on every Grid where GEMLCA Grid resources are deployed.

The deployment of a GEMLCA legacy code service assumes that the legacy application runs in its native environment on a Compute Server. It is the task of the GEMLCA Resource layer to present the legacy application as a Grid service to the user, to communicate with the Grid client and to hide the legacy nature of the application. The deployment process of a GEMLCA 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 deployment defines the execution environment and the parameter set for the legacy application 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 provides the capability to convert legacy codes into Grid services just by describing the legacy parameters and environment values in the XML-based LCID file. However, an end-user without specialist computing skills still requires a user-friendly Web interface (portal) to access the GEMLCA functionalities: to deploy, execute and retrieve results from legacy applications. Instead of developing a new custom Grid portal, GEMLCA was integrated with the workflow-oriented P-GRADE Grid portal extending its functionalities with new portlets.

Following this integration, end-users can easily construct workflow applications built from legacy code services running on different GEMLCA Grid resources. The workflow manager of the portal contacts the selected GEMLCA Resources, passes them the actual parameter values of the legacy code, and then it is the task of the GEMLCA Resource

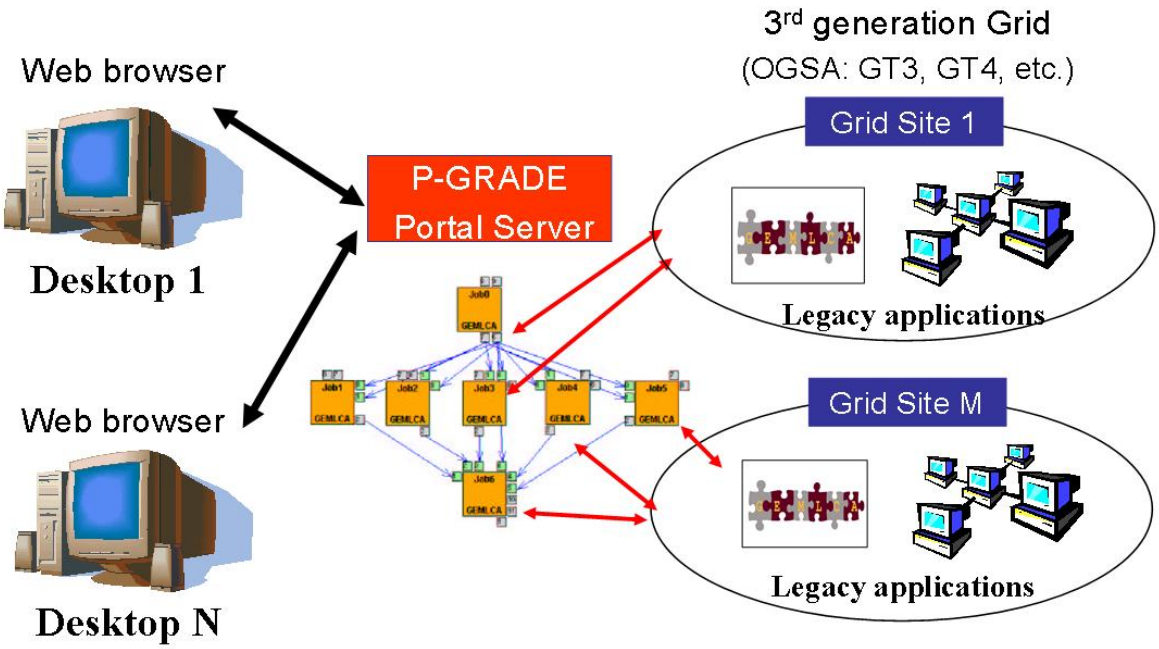


Figure 2: GEMLCA with Grid Portal

4 Connecting GEMLCA to the NGS

Two different scenarios were identified in order to execute legacy code applications on NGS sites. In each scenario both GEMLCA and the P-GRADE portal are installed on the Parsifal cluster of the University of Westminster. As a result, there is no need to deploy any GEMLCA or P-GRADE portal code on the NGS resources.

scenario 1: legacy codes are stored in a central repository and GEMLCA submits these codes as jobs to NGS sites,

scenario 2: legacy codes are installed on NGS sites and executed through GEMLCA.

The two scenarios are supporting different user needs, and each of them increases the usability of the NGS in different ways for end-users. The GEMLCA research team has implemented the first scenario in May 2005, and currently working on the implementation of the second scenario.

This chapter briefly describes these two different scenarios, and the next chapter explains in detail the design and implementation aspects of the first already implemented solution. As the design and implementation of the second scenario is currently work is progress, its detailed description will be the subject of a future publication.

4.1 Scenario 1: Legacy Code Repository for NGS

There are several legacy applications that would be useful for users within the NGS community. These applications were developed by different institutions and currently not available for other members of the community. According to this scenario legacy codes can be uploaded into a central repository and made available for authorised users through a Grid portal. The solution extends the usability of NGS as users can submit not only their own applications but can also utilise other legacy codes stored in the repository.

Users can access the central repository, managed by GEMLCA, through the P-GRADE portal and upload their applications into this repository. After uploading legacy applications users with valid certificates and existing NGS accounts can select and execute legacy codes through the P-GRADE portal on different NGS sites. In this scenario the binary codes of legacy applications are transferred from the GEMLCA server to the NGS sites, and executed as jobs.

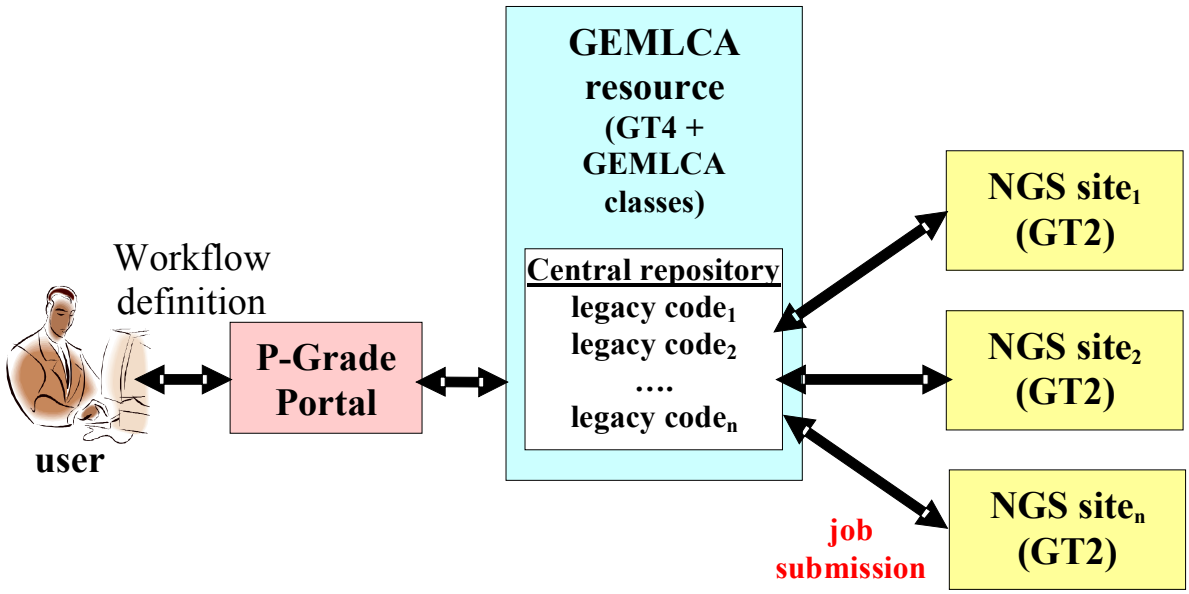


Figure 3: Scenario 1 - Legacy Code Repository for NGS

4.2 Scenario 2: Pre-deployed Legacy Code Services

This solution extends the NGS Grid towards the service-oriented Grid concept. Users cannot only submit and execute jobs on the resources but can also access legacy applications deployed on NGS and include these in their workflows. This scenario is the logical extension of the original GEMLCA concept in order to use it with NGS. In this scenario the legacy codes are already deployed on the NGS sites and only parameters (input or output) are submitted.

Users contact the central GEMLCA resource through the P-GRADE portal, and can access the legacy codes that are deployed on the NGS sites. In this scenario the NGS system administrators have full control of legacy codes that they deploy on their own resources.

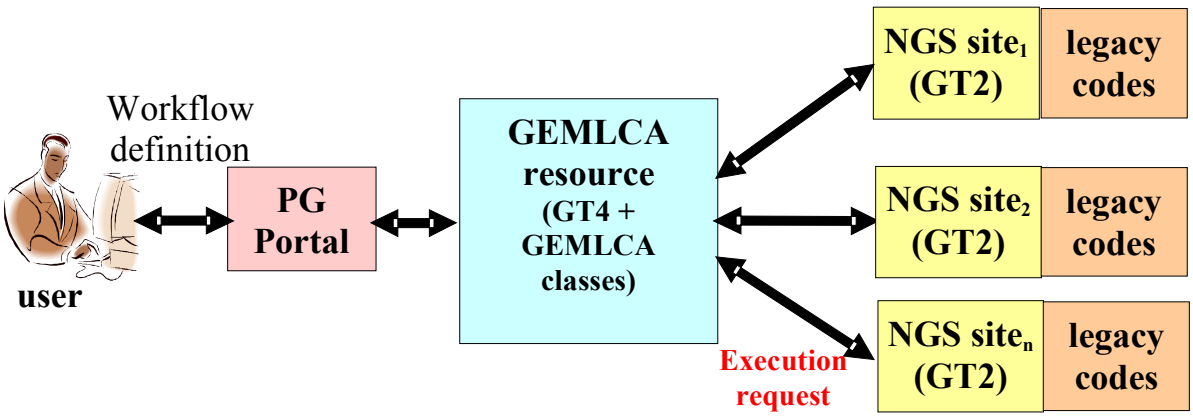


Figure 4: Scenario 2 Pre-Deployed Legacy Code on NGS Sites

5 Legacy Code Repository for the NGS

5.1 Design objectives

The currently implemented solution that enables users to deploy, browse and execute legacy code applications on the NGS sites is based on scenario 1, as described in the previous chapter. This solution utilises the original GEMLCA architecture with the necessary modifications in order to execute the tasks on the NGS resources.

The primary aims of the solution are the following:

- The owners of legacy applications can publish their codes in the central repository making it available for other authorised users within the UK e-Science community. The publication is not different from the original method used in GEMLCA, and it is supported by the administration Grid portlet of the P-GRADE portal, as described in [9]. After publication the code is available for other non-computer specialist end-users.
- Authorised users can browse the repository, select the necessary legacy codes, set their input parameters, and can even create workflows from compatible components. These workflows can then be mapped onto the NGS resources, submitted and the execution visualised.
- The deployment of a new legacy application requires some high level understanding of the code (like name and types input and output parameters) and its execution environment (e.g. supported job managers, maximum number of processors). However, once the code is deployed end-users with no Grid specific knowledge can easily execute it, and analyse the results using the portal interface.

As GEMLCA is integrated with the P-GRADE Grid portal, NGS users have two different options in order to execute their applications. They can submit their own code directly, without the described publication process, using the original facilities offered by the portal. This solution is suggested if the execution is only on an ad-hoc basis when the publication puts too much overhead on the process. However, if they would like to make their code available for a larger community, and would like make the execution simple enough for any end-user, they can publish the code with GEMLCA in the repository.

In order to execute a legacy code on an NGS site, users should have a valid user certificate, for example an e-Science certificate, an NGS account and also an account for the P-GRADE portal running at Westminster. After logging in the portal they download their user certificate from an appropriate myProxy server. The legacy code, submitted to the NGS site, utilise this certificate to authenticate users.

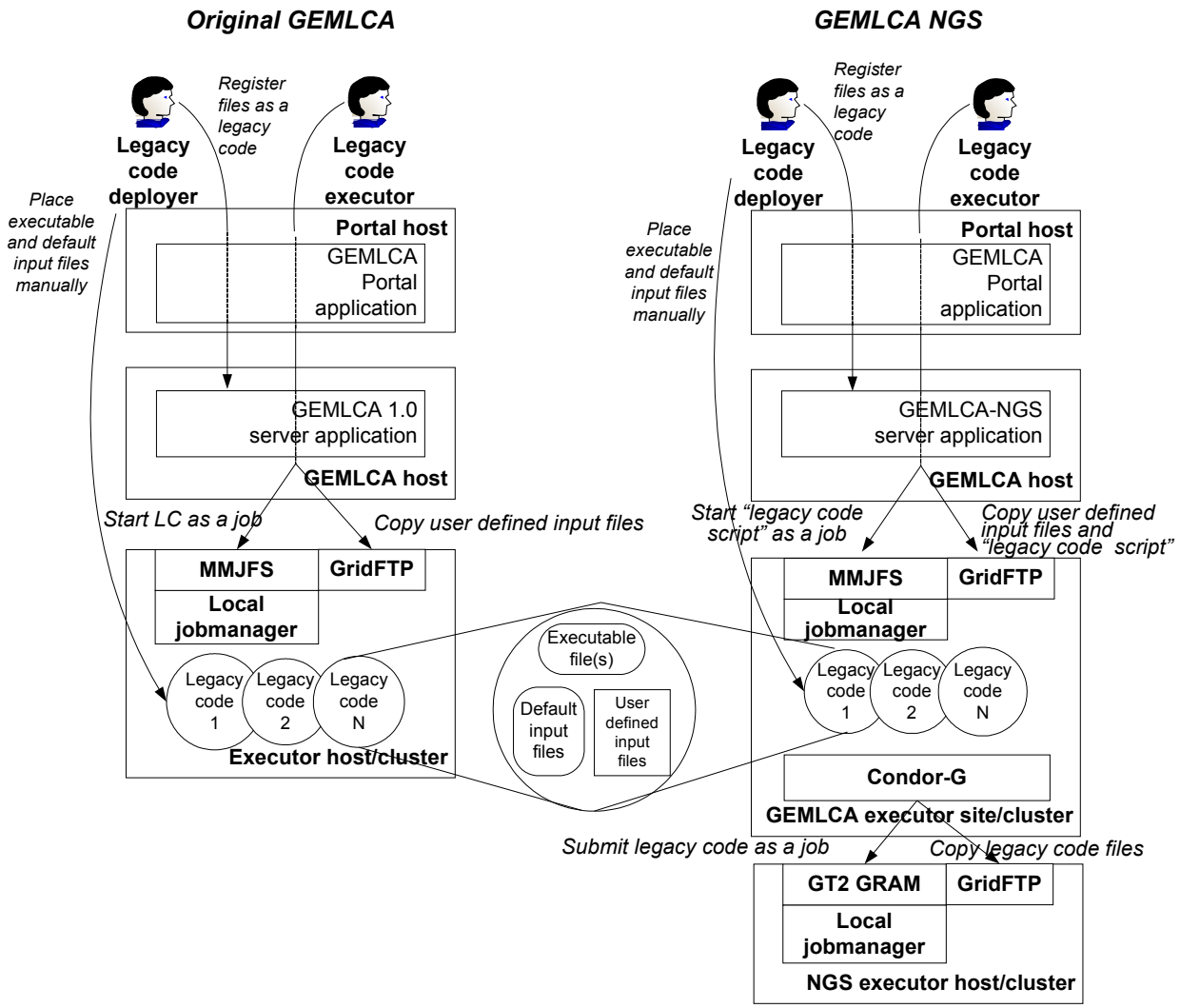


Figure 5: Comparison of the Original and the NGS GEMLCA Concept

5.2 Implementation of the Solution

To fulfil these objectives some modifications and extensions of the original GEMLCA architecture were necessary. Figure 5 compares the original and the extended GEMLCA architectures. As it is shown in the figure, an additional layer, representing the remote NGS resource where the code is executed, appears. The deployment of a legacy code is not different from the original GEMLCA concept; however, the execution has changed significantly in the NGS version. To transfer the executable and the input parameters to the NGS site, and to instruct the remote GT2 GRAM to execute the jobs, required the modification of the GEMLCA architecture, including the development of a special script that interfaces with Condor G.

The major challenge when connecting GEMLCA to the NGS was that NGS sites use Globus Toolkit version 2 (GT2), however the current GEMLCA implementations are based on service-oriented Grid middleware, namely GT3 and GT4. The interfacing between the different middleware platforms is supported by a script, called NGS script, that provides additional functionality required for executing legacy codes on NGS sites. Legacy codes and input files are stored in the central repository but executed on the remote NGS sites. To execute the code on a remote site first the NGS script, executed as a GEMLCA legacy code, instructs the portal to copy the binary and input files from the central repository to the NGS site. Next, the NGS script, using Condor-G, submits the legacy code as a job to the remote site.

The other major part of the architecture where modifications were required is the config.xml file and its related

Java classes. GEMLCA uses an XML-based description file, called config.xml, in order to describe the environmental parameters of the legacy code. This file had to be extended and modified in order to take into consideration a second-level job manager, namely the job manager used on the remote NGS site. The config.xml should also notify the GEMLCA resource that it has to submit the NGS script instead of a legacy code to GT4 MMJFS (Master Managed Job Factory Service) when the user wants to execute the code on an NGS site. The implementation of these changes also required the modification of the GEMLCA core layer.

In order to utilise the new GEMLCA NGS solution:

1. The owner of the legacy application deploys the code as a GEMLCA legacy code in the central repository.
2. The end-user selects and executes the appropriate legacy applications on the NGS sites.

As the deployment process is virtually identical to the one used by the original GEMLCA solution here we concentrate on the second step, the code execution. The following steps are performed by GEMLCA when executing a legacy code on the NGS sites (Fig. 6):

1. The user selects the appropriate legacy codes from the portal, defines input files and parameters, and submits an "execute a legacy code on an NGS site" request.
2. The GEMLCA portal transfers the input files to the NGS site.
3. The GEMLCA portal forwards the users request to a GEMLCA Resource.
4. The GEMLCA resource creates and submits the NGS script as a GEMLCA job to the MMJFS.
5. The MMJFS starts the NGS script.
6. Condor-G contacts the remote GT2 GRAM, sends the binary of the legacy code and its parameters to the NGS site, and submits the legacy code as a job to the NGS site job manager.

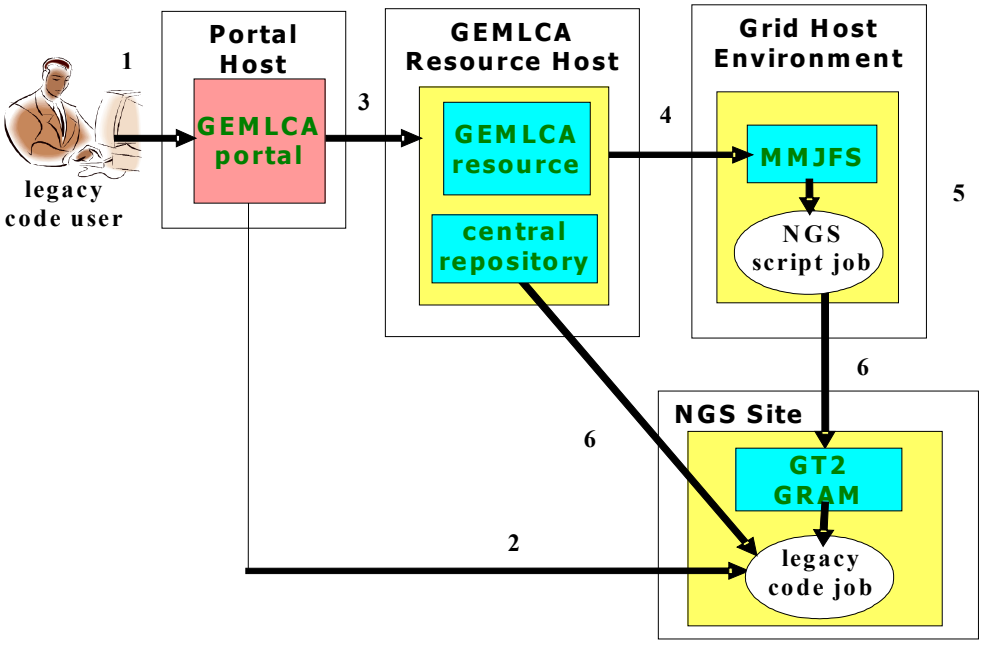


Figure 6: Execution of Legacy Codes on an NGS Site

When the job has been completed on the NGS site the results are transferred from the NGS site to the user in the same way.

6 Results - Traffic simulation on the NGS

A working prototype of the described solution has been implemented and tested creating and executing a traffic simulation workflow on the different NGS resources. The workflow consists of three types of components:

1. The Manhattan legacy code is an application to generate inputs for the MadCity simulator: a road network file and a turn file. The MadCity road network file is a sequence of numbers, representing a road topology of a road network. The MadCity turn file describes the junction manoeuvres available in a given road network. Traffic light details are also included in this file.

2. MadCity [8] is a discrete-time microscopic traffic simulator that simulates traffic on a road network at the level of individual vehicles behaviour on roads and at junctions. After completing the simulation, a macroscopic trace file, representing the total dynamic behaviour of vehicles throughout the simulation run, is created.

3. Finally a traffic density analyser compares the traffic congestion of several runs of the simulator on a given network, with different initial road traffic conditions specified as input parameters. The component presents the results of the analysis graphically.

Each of these applications was published in the central repository at Westminster as GEMLCA legacy code. The publication was done using the administration portlet of the GEMLCA P-GRADE portal. During this process the type of input and output parameters, and environmental values, like job managers and maximum number of processors used for parallel execution, were set. Once published the codes are ready to be used by end-users even with very limited computing knowledge.

Figure 7 shows the workflow graph and the execution of the different components on NGS resources:

- Job 0 is a road network generator mapped at Leeds,
- jobs 1 and 2 are traffic simulators running parallel at Leeds and Oxford, respectively,
- finally, job 3 is a traffic density analyser executed at Leeds.

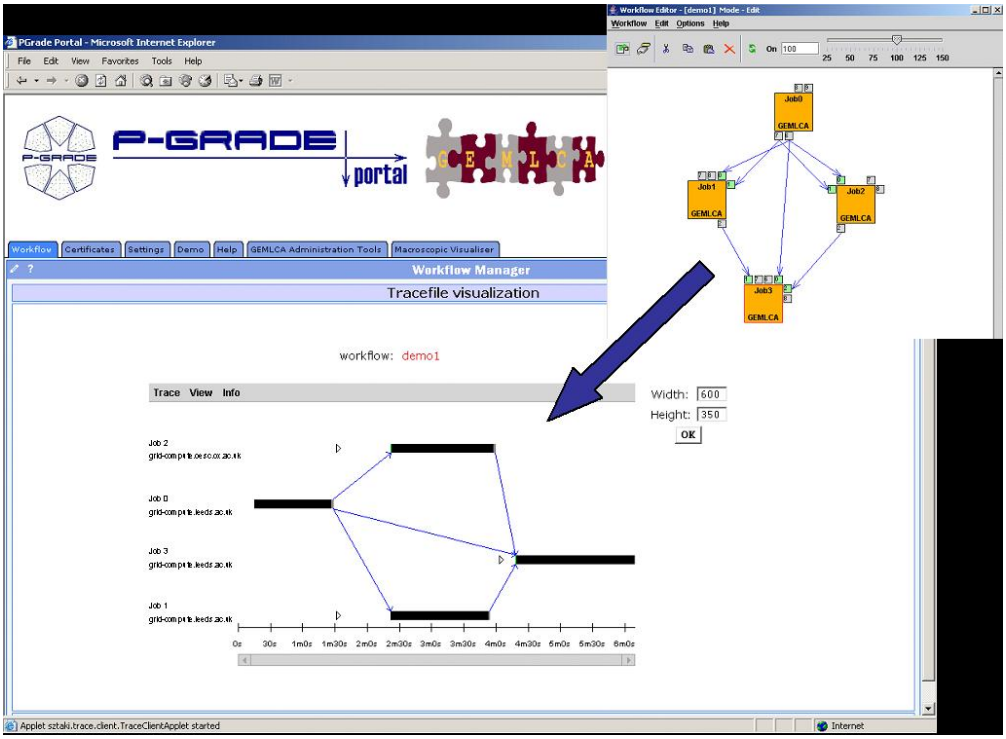


Figure 7: Workflow Graph and Visualisation of its Execution on NGS Resources

When creating the workflow the end-user selected the appropriate application from the repository, set input parameters and mapped the execution to the available NGS resources.

During execution the NGS script run, contacted the remote GT2 GRAM, and instructed the portal to pass executables and input parameters to the remote site. When finishing the execution the output files were transferred back to Westminster and were made available for the user.

7 Conclusion and Future Work

The implemented solution successfully demonstrated that additional services, like legacy code support, run and maintained by third party service providers can be added to production Grid systems. The major advantage of this solution is that the reliability of the core Grid infrastructure is not compromised, and no additional effort is required from Grid system administrators. On the other hand, utilizing these services the usability of these Grids can be significantly improved.

Utilising and re-engineering the GEMLCA legacy code solution two different scenarios were identified to provide legacy code support for the UK NGS. The first, providing a legacy code repository functionality, and allowing the submission of legacy applications as jobs to NGS resources was successfully implemented and demonstrated. The final production version of this architecture and its official release for NGS users is scheduled for June 2005.

The second scenario, that extends the NGS with pre-deployed legacy code services, is currently in the design phase. Challenges are identified concerning its implementation, especially the creation and management of virtual organizations that could utilize these pre-deployed services.

References

[1] R. Alfieri, R. Cecchini, V. Ciaschini, L. dellAgnello, A. Frohner, A. Gianoli, K. Lorentey, , and F. Spata. Voms, an authorization system for virtual organizations. http://infnforge.cnaf.infn.it/voms/VOMS-Santiago.pdf.

[2] S. Andreozzi, S. Fantinel, D. Rebatto, L. Vaccarossa, and G. Tortone. A monitoring tool for a grid operation center. In *CHEP 2003*, La Jolla, California, March 2003.

[3] Mario Antonioletti, Malcolm Atkinson, Rob Baxter, Andrew Borley, Neil P Chue, Hong, Brian Collins, Neil Hardman, Ally Hume, Alan Knox, Mike Jackson, Amy Krause, Simon Laws, James Magowan, Norman W Paton, Dave Pearson, Tom Sugden, Paul Watson, and Martin Westhead. The design and implementation of grid database services in ogsa-dai. *Concurrency and Computation: Practice and Experience*, 17:357–376, 2005.

[4] David Baker, Mark Baker, Hong Ong, and Helen Xiang. Integration and operational monitoring tools for the emerging uk e-science grid infrastructure. In *Proceedings of the UK e-Science All Hands Meeting (AHM 2004)*, East Midlands Conference Centre, Nottingham, 2004.

[5] B. Balis, M. Bubak, and M. Wegiel. A solution for adapting legacy code as web services. In V. Getov and T. Kiellmann, editors, *Component Models and Systems for Grid Applications*, pages 57–75. Springer, 2005. ISBN 0-387-23351-2.

[6] C. Baru, Moore R, A. Rajasekar, and M. Wan. The sdsc storage resource broker. In *Proc.CASCON'98 Conference*, November 1998.

[7] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, August 2001. www.globus.org/research/papers/MDS-HPDC.pdf.

[8] A. Gourgoulis, G. Terstyansky, P. Kacsuk, and S.C. Winter. Creating scalable traffic simulation on clusters. In *PDP2004. Conference Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network based Processing*, La Coruna, Spain, February 2004.

[9] A. Goyeneche, T. Kiss, G. Terstyanszky, G. Kecskemeti, T. Delaitre, P. Kacsuk, and S.C. Winter. Experiences with deploying legacy code applications as grid services using gemlca. In P.M.A. Sloot, A.G. Hoekstra, T. Priol,

A. Reinefeld, and M. Bubak, editors, *Conf. Proc. of the European Grid Conference*, pages 851–860, Science Park Amsterdam, The Netherlands, February 2005. LNCS 3470.

[10] Y. Huang, I. Taylor, and D. W. Walker. Wrapping legacy codes for grid-based applications. In *Proceedings of the 17th International Parallel and Distributed Processing Symposium, workshop on Java for HPC*, Nice, France, April 2003. ISBN 0-7695-1926-1.

[11] P. Kacsuk, A. Goyeneche, T. Delaitre, T. Kiss, Z. Farkas, and T. Boczko. High-level grid application environment to use legacy codes as ogsa grid services. In *Conf. Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 428–435, November 2004.

[12] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30, July 2004.

[13] Cs. Nemeth, G. Dozsa, R. Lovas, and P. Kacsuk. The p-grade grid portal. In *Computational Science and Its Applications - ICCSA 2004: International Conference*, pages 10–19, 2004. LNCS 3044.

[14] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services. In *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003. http://www.globus.org/Security/GSI3/GT3-Security-HPDC.pdf.