

UNIVERSITY OF WESTMINSTER



## WestminsterResearch

<http://www.wmin.ac.uk/westminsterresearch>

### Legacy code support for production grids.

**Tamas Kiss<sup>1</sup>**  
**Gabor Terstyanszky<sup>1</sup>**  
**Gabor Kecskemeti<sup>1</sup>**  
**Sz. Illes<sup>1</sup>**  
**Thierry Delaitre<sup>1</sup>**  
**Stephen Winter<sup>1</sup>**  
**Peter Kacsuk<sup>2</sup>**  
**G.Sipos<sup>2</sup>**

<sup>1</sup>Cavendish School of Computer Science, University of Westminster

<sup>2</sup>MTA SZTAKI Lab. Of Parallel & Distributed Systems, Budapest, Hungary

Copyright © [2005] IEEE. Reprinted from 6th IEEE/ACM International Workshop on Grid Computing, 13-14 Nov 2005, Seattle, USA.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Westminster's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to [pubs-permissions@ieee.org](mailto:pubs-permissions@ieee.org). By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

---

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners. Users are permitted to download and/or print one copy for non-commercial private study or research. Further distribution and any use of material from within this archive for profit-making enterprises or for commercial gain is strictly forbidden.

---

Whilst further distribution of specific materials from within this archive is forbidden, you may freely distribute the URL of WestminsterResearch. (<http://www.wmin.ac.uk/westminsterresearch>).

In case of abuse or copyright appearing without permission e-mail [watts@wmin.ac.uk](mailto:watts@wmin.ac.uk).

# Legacy Code Support for Production Grids

T. Kiss, G. Terstyanszky, G. Kecskemeti, Sz. Illes, T. Delaittre, S. Winter, P. Kacsuk, G. Sipos

**Abstract**—In order to improve reliability and to deal with the high complexity of existing middleware solutions, today's 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.

**Index Terms**—Grid portal, Grid service, legacy code, production Grid

## I. 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. Today's 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) [1] 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 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 [2] 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 functionality 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 operator's point of view but offers essential functionality for the end-users.

## II. THE UK NATIONAL GRID SERVICE (NGS)

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,

Manuscript received June 3, 2005. This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

T. Kiss, G. Terstyanszky, G. Kecskemeti, Sz. Illes, T. Delaittre and S. Winter are with the University of Westminster, 115 New Cavendish Street, London, W1W 6UW, email: [gemlca-discuss@cpc.wmin.ac.uk](mailto:gemlca-discuss@cpc.wmin.ac.uk)

P. Kacsuk and G. Sipos are with the MTA SZTAKI Lab. of Parallel and Distributed Systems, H-1518 Budapest, P.O. Box 63, Hungary

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) [3], 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) [4] with OGSA-DAI [5] to provide access to data on NGS nodes. It uses the Globus Monitoring and Discovery Service (MDS) [6] to handle information of NGS nodes. Ganglia [7], Grid Integration Test Script (GITS) [8] and Nagios [9] 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) [10]. VOMS supports central management of user registration and authorisation taking into consideration local policies on resource access and usage.

### III. 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 (GEMMLCA) 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 GEMMLCA 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.

GEMMLCA 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 GEMMLCA 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 GEMMLCA 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 **GEMMLCA Resource** layer provides a set of Grid services which expose legacy codes as Grid services.
- 4) The fourth component is the **GEMMLCA Client** that can be installed on any client machine through which a user would like to access the GEMMLCA resources.

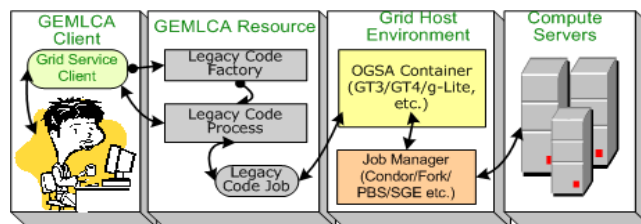


Figure 1 GEMMLCA Conceptual Architecture

The novelty of the GEMMLCA concept compared to other similar solutions like [13] or [14] is that it requires minimal effort from both Compute Server administrators and end-users of the Grid. The Compute Server administrator should install the GEMMLCA 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 GEMMLCA 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 GEMMLCA Grid resources. If they would like to deploy legacy code services on GEMMLCA Grid resources they can do so, but these services cannot be accessed by other Grid users. As a last resort, if no GEMMLCA portal is available for the Grid, a user must install the GEMMLCA Client on their client machine. However, since it requires some IT skills to do this, it is recommended that a GEMMLCA portal is installed on every Grid where GEMMLCA Grid resources are deployed.

The deployment of a GEMMLCA legacy code service assumes that the legacy application runs in its native environment on a Compute Server. It is the task of the GEMMLCA 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 GEMMLCA 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 GEMMLCA Resource layer to handle the legacy application as a Grid service.

GEMMLCA 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 GEMMLCA functionalities: to deploy, execute and retrieve results from legacy applications. Instead of developing a new custom Grid portal, GEMMLCA 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 GEMMLCA Grid resources. The workflow manager of the portal contacts the selected GEMMLCA Resources, passes them the actual parameter values of the legacy code, and then it is the task of the GEMMLCA Resource to execute the legacy code with the actual parameter values. The other important task of the GEMMLCA Resource is to deliver the results of the legacy code service back to the portal.

#### IV. CONNECTING GEMMLCA TO THE NGS

Two different scenarios were identified in order to execute legacy code applications on NGS sites. In each scenario both GEMMLCA 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 GEMMLCA or P-GRADE portal code on the NGS resources.

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

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

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 GEMMLCA 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 in progress, its detailed description will be the subject of a future publication.

#### A. 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 GEMMLCA, 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

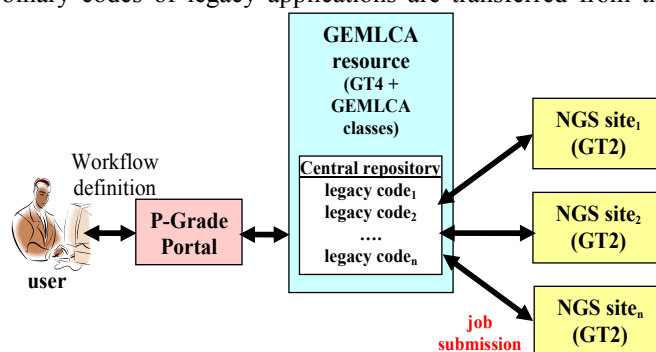


Figure 3: Scenario 1 - Legacy Code Repository for NGS

GEMMLCA server to the NGS sites, and executed as jobs.

#### B. 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 GEMMLCA

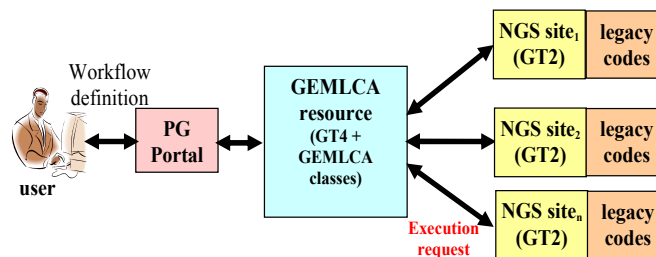


Figure 4: Scenario 2 – Pre-Deployed Legacy Code on NGS Sites

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 GEMMLCA 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.

A. 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 [11]. 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 to 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

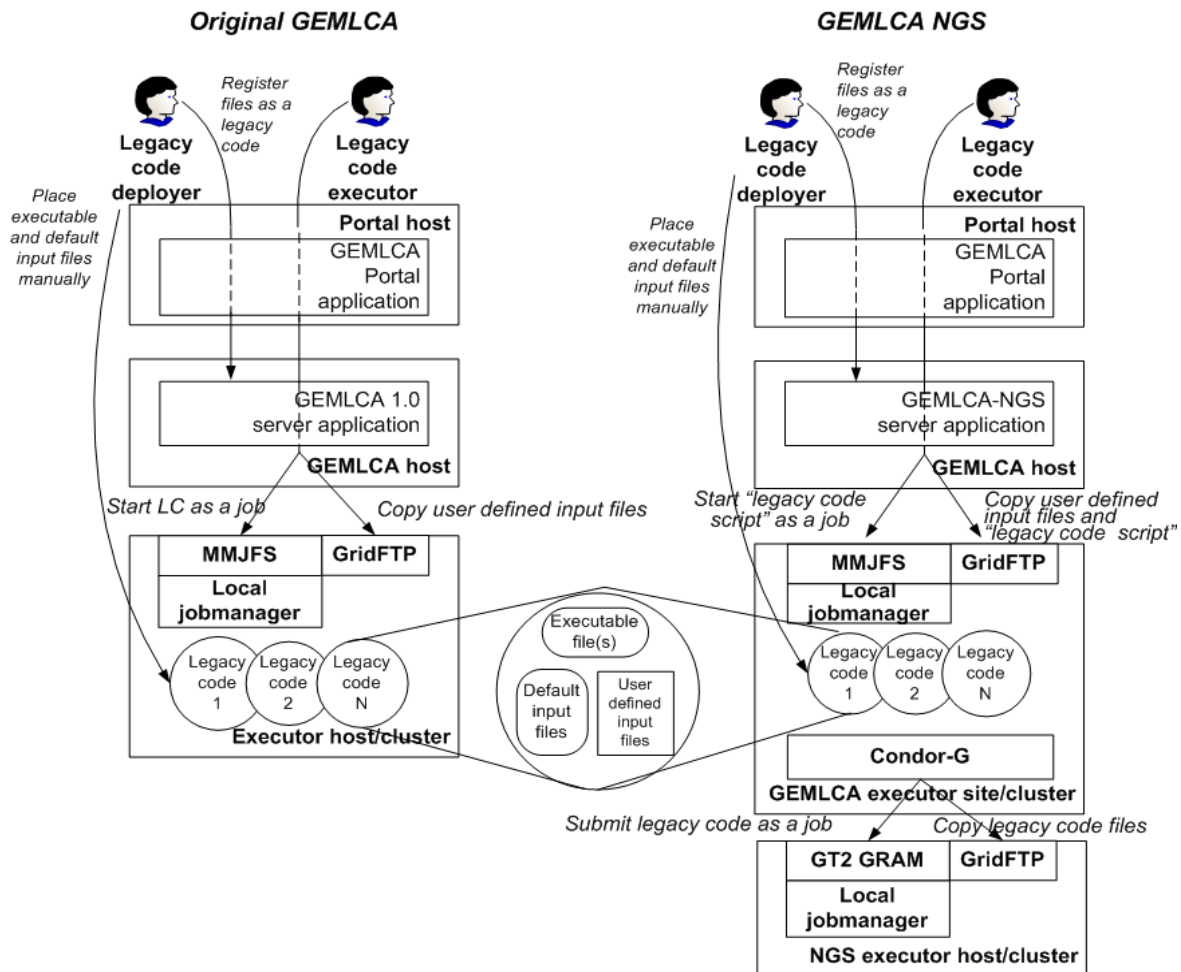


Figure 5 Comparison of the Original and the NGS GEMLCA Concept

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.

### B. Implementation of the Solution

To fulfil these objectives some modifications and extensions of the original GEMMLCA architecture were necessary. Figure 5 compares the original and the extended GEMMLCA 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 GEMMLCA 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 GEMMLCA architecture, including the development of a special script that interfaces with Condor G.

The major challenge when connecting GEMMLCA to the NGS was that NGS sites use Globus Toolkit version 2 (GT2), however the current GEMMLCA 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 GEMMLCA legacy code, instructs the portal to copy the binary and input files from the central repository to

the remote NGS site. The “*config.xml*” should also notify the GEMMLCA 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 GEMMLCA core layer.

In order to utilise the new GEMMLCA NGS solution:

- 1) The owner of the legacy application deploys the code as a GEMMLCA 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 GEMMLCA solution here we concentrate on the second step, the code execution. The following steps are performed by GEMMLCA 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 GEMMLCA portal transfers the input files to the NGS site.
- 3) The GEMMLCA portal forwards the user’s request to a GEMMLCA Resource.
- 4) The GEMMLCA resource creates and submits the NGS script as a GEMMLCA 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.

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.

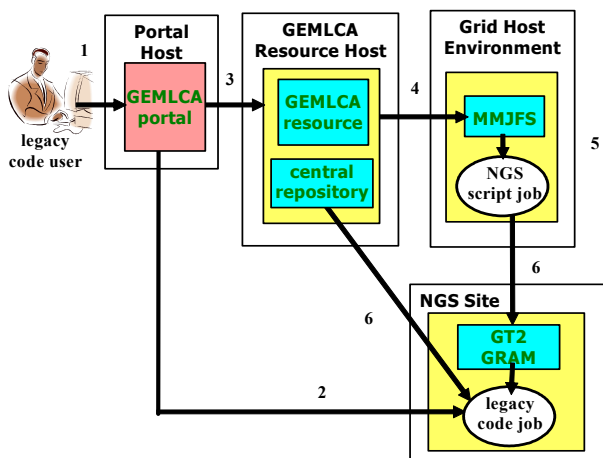


Figure 6: Execution of Legacy Codes on an NGS Site

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. GEMMLCA 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

## VI. 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* [12] 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.

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.

## VII. 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] 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", Conf. Proc. of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 428-435, November 8, 2004, Pittsburgh, USA
- [2] Cs. Nemeth, G. Dozsa, R. Lovas, P. Kacsuk, "The P-GRADE Grid portal", In: Computational Science and Its Applications - ICCSA 2004: International Conference, Assisi, Italy, 2004, LNCS 3044, pp. 10-19.
- [3] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke, "Security for Grid Services" Twelfth International Symposium on High Performance

- Distributed Computing (HPDC-12), IEEE Press, June 2003. <http://www.globus.org/Security/GS13/GT3-Security-HPDC.pdf>
- [4] Baru, C., R. Moore, A. Rajasekar, M. Wan: "The SDSC Storage Resource Broker," Proc.CASCON'98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada.
- [5] 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-DAP", Concurrency and Computation: Practice and Experience, vol. 17, 2005 pp. 357-376
- [6] Karl Czajkowski, Steven Fitzgerald, Ian Foster and Carl Kesselman. "Grid Information Services for Distributed Resource Sharing" Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001. [www.globus.org/research/papers/MDS-HPDC.pdf](http://www.globus.org/research/papers/MDS-HPDC.pdf)
- [7] Matthew L. Massie, Brent N. Chun, David E. Culler: "The GAnglia Distributed Monitoring System: Design, Implementation, and Experience", Parallel Computing, Vol. 30, Issue 7, July 2004
- [8] David Baker, Mark Baker, Hong Ong and Helen Xiang: "Integration and operational monitoring tools for the emerging UK e-Science Grid infrastructure", Proceedings of the UK e-Science All Hands Meeting (AHM 2004), East Midlands Conference Centre, Nottingham, from 31st August - 3rd September 2004
- [9] S. Andreozzi, S. Fantinel, D. Rebatto, L. Vaccarossa, G. Tortone: "A monitoring tool for a GRID operation center", CHEP 2003, La Jolla, California, March 24-28 2003
- [10] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, 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>
- [11] A. Goyeneche, T. Kiss, G. Terstyanszky, G. Kecskemeti, T. Delaitre, P. Kacsuk, S.C. Winter, Experiences with Deploying Legacy Code Applications as Grid Services using GEMLCA, Conf. Proc. of the European Grid Conference, February 14 -16, 2005, Science Park Amsterdam, The Netherlands, Volume editors: P.M.A. Sloot, A.G. Hoekstra, T. Priol, A. Reinefeld, M. Bubak, pp 851-860, LNCS 3470
- [12] 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.
- [13] Y. Huang, I. Taylor, D. W. Walker, "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
- [14] B. Balis, M. Bubak, and M. Wegiel, "A Solution for Adapting Legacy Code as Web Services, in "Component Models and Systems for Grid Applications" edited by V. Getov and T. Kiehlmann, Springer, 2005, pp 57-75, ISBN 0-387-23351-2