

RealityGrid: An Integrated Approach to Middleware through ICENI

Jeremy Cohen Nathalie Furmento Gary Kong **Anthony Mayer**
Steven Newhouse John Darlington

London e-Science Centre, Imperial College London, South Kensington Campus, London SW7 2AZ, UK
Email: lesc-staff@doc.ic.ac.uk

Abstract

The advancement of modelling and simulation within complex scientific applications is currently constrained by the rate at which knowledge can be extracted from the data produced. As Grid Computing evolves, new means of increasing the efficiency of data analysis are being explored. RealityGrid aims to enable more efficient use of scientific computing resources within the condensed matter, materials and biological science communities. The Imperial College e-Science Networked Infrastructure (ICENI) Grid Middleware provides an end-to-end pipeline that simplifies the stages of computation, simulation and collaboration. Scientists can utilise advanced scheduling mechanisms to ensure efficient planning of computations, visualise and interactively steer simulations and securely collaborate with colleagues via the Access Grid through a single integrated middleware application.

1 Introduction

The advancement of modelling and simulation within complex scientific applications is currently constrained by the rate at which knowledge can be extracted from the data produced. As Grid Computing evolves, new means of increasing the efficiency of data analysis are being explored. RealityGrid [12], an Engineering and Physical Sciences Research Council (EPSRC) project that is exploring the use of Grid technology to investigate the structure of matter at the meso and nanoscale levels, aims to enable more efficient use of scientific computing resources within the condensed matter, materials and biological science communities.

The Imperial College e-Science Networked Infrastructure (ICENI) [9, 8] Grid Middleware provides an end-to-end pipeline that simplifies the stages of computation, simulation and collaboration. Scientists can utilise advanced scheduling mechanisms to ensure efficient planning of computations, visualise and interactively steer simulations and securely collaborate with colleagues via the Access Grid through a single integrated middleware application. In this paper, we explore various computer science issues raised by the RealityGrid project and show how the ICENI Grid Middleware has been developed as a solution to these problems.

The rest of the paper is organised as follows. Section 2 introduces the stages of the application pipeline. These stages are covered in detail in Sec-

tions 3, 4 and 5. Section 6 describes the deployment work that has been undertaken. We conclude in Section 7 by looking at future work that can further enhance ICENI's contribution to the RealityGrid project.

2 The Application Pipeline

The application pipeline consists of a series of operations required during the lifetime of an application. These operations can be grouped into three broad areas:

Deployment. This stage involves the preparation of the code for the execution environment. It also involves the specification of the application meta-data that will be used, for example, during the scheduling phase, as well as the possible componentisation of the code.

Execution. The application is executed within a Grid environment. This stage of the pipeline includes scheduling execution against available resources. In most of the cases, the scheduling algorithm will try to minimise the execution time of the overall application. Different criteria might also be used, such as the location of the resources on which the different components will execute (this might be specified by the end-user when composing and submitting the application).

Results Analysis. While the application is running or at the end of its execution, the output from the application execution might be analysed. This may be done through visualisation – either in real-time or after completion of the execution – or by other means of mining results data.

These three steps provide an end-to-end pipeline that allows end-users to easily deploy their application for execution on the Grid resources, and to interact with the application while it is running. We now present in more detail each of the stages of the application pipeline.

3 Deploying an Application

The first stage of the application pipeline is the deployment of the application. Placing an application onto the Grid raises two issues - how native codes, not designed for remote deployment, can operate remotely within the context of a distributed system and how to select the target resources to provide optimal execution times and resource usage. This section looks at the issue of “Grid-enabling” native applications, while optimal selection of target resources is carried out at execution time and discussed in Section 4.

3.1 Legacy Code Issues

LB3D [6], a three-dimensional Lattice-Boltzmann code written in Fortran90 and parallelised using MPI, was developed by partners within the RealityGrid project and is used extensively in application pipeline development within the ICENI framework. In order to support LB3D and similar legacy application codes, ICENI provides the “binary component”. This is a mechanism by which a legacy code, compiled on a particular machine and linked with particular libraries, is wrapped as an ICENI component. As such, it is published as a component service and made available for composition within the ICENI runtime system. In order to target multiple resources, the native application must be compiled and wrapped for each available platform.

The definition of a new binary component is straightforward through the use of a toolset (see Figure 1). The Binary Component Builder is an application that simplifies wrapping of native code applications into a binary component for use within the ICENI framework. Different characteristics of the binary component need to be specified such as the

location of the binary code, the files that need to be staged before and after the execution, as well as a possible resource name on which the code must be executed.

The ICENI component model means that where an application is available on many resources, that plurality may be hidden from the user, who may select a location specific binary component, or may select a location independent binary component, leaving the choice of *which* binary component (and hence hardware resource) to the run-time scheduler¹.

The binary component has ports that allow standard input, output and error to be streamed to and from the native code’s process. However, most scientific codes require files to be made available at initialisation, and following execution they output their results to local files. LB3D is no exception. To overcome this restriction, the binary component wrapping includes a specification of required and provided files, and stages these files using either gridFTP or http protocols, or local file copy. Details of the particular protocols enabled locally are available from each resource’s launcher service and any resource registered within ICENI must specify which of these protocols it provides, and for each of them the address of the server and the path location. The selection of a specific launcher to execute a component will determine which protocol has to be used to transfer files. This process of automatic file staging hides the user from the remote nature of the code execution, and allows the user interface component (in the case of LB3D, a parameter input panel) to execute on a different machine from the binary in a way entirely transparent to the user.

3.2 Componentising LB3D

The LB3D application has been wrapped into a binary component. The configuration of the component is done through a JDML (Job Description Markup Language) [1] document that specifies different parameters for the executable such as the location of the executable, a list of arguments, and an input and output sandbox. For each of the files requiring to be staged to and from the resource before and after the execution, information also needs to be provided on the transport method required to copy the file across. LB3D is configured through its own input file which uses attribute-value pairs to define the application behaviour. An Input Editor component presents the contents of the input

¹As seen in Section 6, LB3D has been compiled for two of the London e-Science Centre’s parallel resources, a 24x750MHz UltraSparcIII processor Sun E6800 and a 64 node Dual Pentium 4 Xeon Myrinet interconnected cluster.

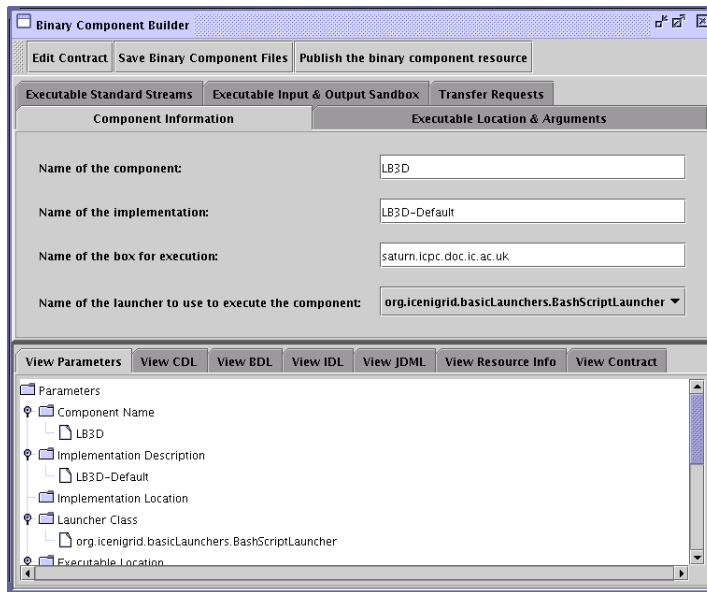


Figure 1: The ICENI Binary Component Builder

file within a graphical interface allowing easy editing of the input values prior to execution of the LB3D application. Simple output components, part of ICENI's built in set of basic components, are then connected to the LB3D binary component to allow visual monitoring of the standard output and error streams. This results in a four component application composition (see Figure 2).

4 Application Execution

4.1 Scheduling an Application

Dynamically selecting a resource that has low load and guaranteed availability along with sufficient processing power in the context of a continuously changing Grid environment is a complex task. The key to approaching even an approximate solution is information, and for this purpose, ICENI is built around the concept of capturing meta-data wherever possible. Each of the components within the LB3D application composition is scheduled by the ICENI scheduling framework and each may execute on a different physical system. Resource information is provided by a published resource service, which is discovered by the ICENI scheduling framework. This process allows continuous updates on the state of the target Grid fabric. In addition, scheduling decisions require performance characteristics of the application [11, 7, 4].

Application performance models are constructed from three sources of meta-data: application composition, component workflow and compo-

nent performance data. Application composition is supplied by virtue of ICENI's component based design model, and is thus provided by the user on an application-by-application basis, while component workflow is encoded within component meta-data provided by the designer.

Application Composition. The composition of the application is accessible through ICENI's component based design model, and is thus provided by the user on an application-by-application basis.

Component Workflow. The activity of each individual component is encoded within component meta-data provided by the component designer.

Component Performance Data The performance characteristics of the component on the particular resources in question are necessary.

The gathering of component performance data is achieved by instrumenting the ICENI runtime system so as to provide events recording the initiation and completion of component threads and all inter-component communication. Together with the composite workflow model of the application, these events can be used to infer activity times for the individual components. This data is captured by a performance repository service which listens to performance events, and stores the event data using a flexible database solution. The event information includes the values of flagged application parameters, as well as the particular resource upon which the execution time is observed.

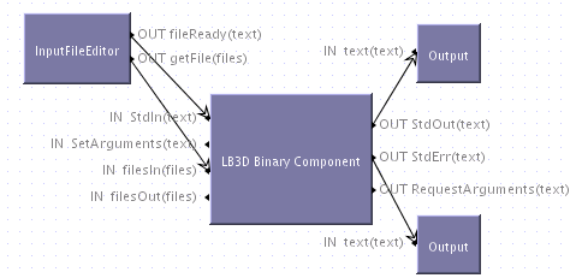


Figure 2: LB3D Component Composition within ICENI

When the ICENI scheduling system requires performance characteristics, the appropriate information is supplied by the repository. The option is also available to use developer supplied component performance models in the place of historical data. The various scheduling mechanisms are described in [13]. Support for advanced reservations is also integrated into the ICENI scheduling framework allowing execution capacity to be reserved as required.

4.2 Execution

At run-time, the LB3D Input Editor component generates its output file and then passes control to the binary component, the next component in the composition. It also passes information about the location of any files that may be required by the next component to execute. The binary component can then obtain the required data before execution using local knowledge about how to transfer data from the resource that was running the Input Editor component. In a similar way, files can be transferred from the resource running the binary component at the end of the execution of the LB3D application. As output is produced on the standard output and error streams, this data is displayed within the output components which may execute on different resources to the binary component. This allows a remote resource to execute the LB3D algorithm while the output data is transmitted back to the user’s local system in real-time.

5 Collaborative Visualisation and Steering

ICENI supports collaborative e-Science by providing a number of mechanisms that enable an LB3D simulation to be visualised and steered by multiple remote partners (see Figure 3), and supports an integrated Access Grid “collaboratory”.

Visualisation. Much of the science enabled by

LB3D concerns features (such as the gyroid phase) that are very difficult to discover automatically from the data; such features are only discernible through visualisation. ICENI enables easy remote visualisation of an existing LB3D simulation through the exposure of running components as services. Collaborators can discover an executing application, and connect visualisation components at run-time without interfering or linking with the initial application.

Visualisation components are provided to allow both local rendering of the visualisation data and to transmit the visualisation over the Access Grid. Multiple display components can be added to a visualisation composition allowing some systems to visualise the output data locally while others receive the visualisation via the Access Grid. Stereo visualisation components are also available.

Steering. The RealityGrid “fast-track” steering library [5] has been wrapped into an ICENI component. This library provides hooks and instrumentation into LB3D (and many other RealityGrid applications), and provides the link from the infrastructure to the application. As an ICENI component, the steering library is published as a service, and may be discovered and invoked by anyone with the correct access privileges. By utilising the ICENI framework, multiple clients can invoke the steering library, and their commands are passed in a coherent way to the application. This provides added value over using the steering library alone.

Since ICENI is Java-based, the steering library methods have been exposed to ICENI using JNI. An ICENI steering proxy component wraps the code of the native library using JNI. The ICENI steering client components, which provide a graphical interface for a user to easily steer the LB3D computation, interact

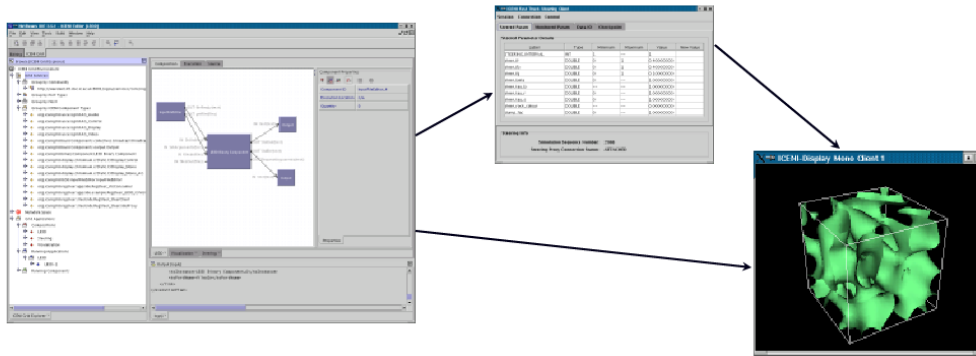


Figure 3: The Netbeans interface is used to start the LB3D, Visualisation and Steering applications, the Steering GUI is then used to steer the LB3D execution, affecting the visualisation output.

with the ICENI proxy component in order to get access to the steering library functionalities (see Figure 4). Multiple steering clients can be connected to a single steering proxy component.

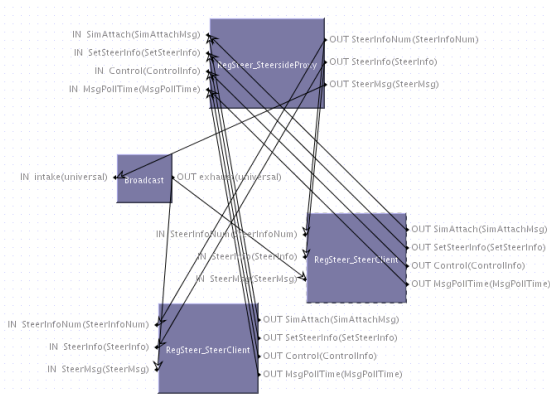


Figure 4: Component composition for the LB3D steering application

Access Grid. The visualisation tool uses the Visualization Toolkit (VTK) [3] and is written on top of the Chromium library [2]. For those not willing or able to run Chromium and VTK, a facility is provided to stream the video of the visualisation through the Access Grid.

A further series of ICENI components allow Access Grid control through the ICENI framework. The Access Grid nodes are started through ICENI and the control of the different nodes is done centrally through the Access Grid Controller, another ICENI component. Any modifications to the Access Grid session from one node are automatically propagated to all other nodes. The controller

interface, a Java Swing GUI, provides a full complement of Access Grid control functions through an easy to use interface.

The ICENI Access Grid components also provide encryption of the data through a video and an audio key. These keys can be generated from any control component, they are then sent to all the display nodes that are part of the session. Any third-party user connected to the AG room will then not be able to connect to the ICENI session without access to the correct key.

The ability to broadcast visualisation data over the Access Grid is of particular importance when working with collaborators in different locations. Visualisation data appears alongside the video windows of the other users in the Access Grid session simplifying the analysis of the data. The ability for a single operator to startup and configure the Access Grid node at one or more remote locations, along with generating and distributing encryption keys to those locations, provides security and reduced administration costs.

6 Test Deployment

LB3D has been compiled for two of the London e-Science Centre's parallel resources, a 24x750MHz UltraSparcIII processor Sun E6800 and a 64 node Dual Pentium 4 Xeon Myrinet interconnected cluster. Since performance testing is not the focus of this paper, this issue was not considered during the test deployment. The focus was on the complete application pipeline, providing more than one parallel resource for the scheduling framework to consider. This allowed scheduling to take into account a choice of two available implementations at different execution locations. By submitting the application

to run several times, performance data can be fed into the performance database enabling more accurate decisions about how to schedule the application in the most efficient manner.

Combining LB3D with various additional elements of the ICENI Middleware, a full end-to-end Grid infrastructure deployment has been possible. A Java Web Start installation process is available allowing simple download and installation of both server and client-side ICENI packages. The ICENI Control Centre provides a graphical interface with wizards to simplify the process of configuring and starting ICENI services. The binary component builder allows wrapping and deployment of native binaries into an ICENI Grid infrastructure. ICENI is available at <http://www.lesc.ic.ac.uk/iceni/>.

7 Conclusion and Further Work

The combination of LB3D, the “fast-track” steering library and ICENI has enabled the deployment of a complete end-to-end Grid infrastructure showcasing many advances in collaborative working environments for scientists. The ability for several geographically distributed scientists to come together via the Access Grid and jointly monitor and steer a complex computation in real-time is an important advancement of the Grid. In addition, there is the provision of tools aimed directly at making the complex installation and configuration procedures of the software practical for end users. This adds a vital element for future wider deployment of the software.

Future work includes the transition from the current fast-track file-based steering approach to the use of service-based steering. ICENI’s service oriented architecture is ideal for this approach, making the movement of steering data between nodes within a Grid easier and more transparent to the user. The use of lightweight Web Services to launch jobs is important and an early prototype has been demonstrated in the form of the WS-JDML [10] service developed at the London e-Science Centre.

References

- [1] A Common Job Description Markup Language written in XML. <http://www.lesc.doc.ic.ac.uk/projects/jdml.pdf>.
- [2] SourceForge: The Chromium Project. <http://sourceforge.net/projects/chromium/>.
- [3] The Visualization ToolKit. <http://public.kitware.com/VTK/>.
- [4] A. Afzal, J. Darlington, S. McGough, S. Newhouse, and L. Young. Workflow Enactment in ICENI. In *UK e-Science All-Hands Meeting*, August 2004.
- [5] J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning, and A. R. Porter. Computational Steering in RealityGrid. In *UK e-Science All Hands Meeting*, August 2003.
- [6] J. Chin, J. Harting, S. Jha, P. V. Coveney, A. R. Porter, and S. M. Pickles. Steering in Computational Science: Mesoscale Modelling and Simulation. *Contemporary Physics*, (44):417–434, 2003.
- [7] J. Darlington, S. McGough, S. Newhouse, and L. Young. Performance Architecture within ICENI. In *UK e-Science At All-Hands Meeting*, August 2004.
- [8] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. ICENI: An Open Grid Service Architecture Implemented with Jini. In *SuperComputing 2002*, Baltimore, USA, November 2002.
- [9] N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington. ICENI: Optimisation of Component Applications within a Grid Environment. *Journal of Parallel Computing*, 28(12):1753–1772, 2002.
- [10] W. Lee, S. McGough, S. Newhouse, and J. Darlington. Standards Approach to Job Submission through Web Services. In *UK e-Science All-Hands Meeting*, August 2004.
- [11] S. McGough, A. Afzal, N. Furmento, A. Mayer, S. Newhouse, and L. Young. Making the Grid Predictable through Reservations and Performance Modelling. Submitted to a special issue of *The Computer Journal* on Grid Performability.
- [12] RealityGrid Project. <http://www.realitygrid.org/>.
- [13] L. Young, S. McGough, S. Newhouse, and J. Darlington. Scheduling Architecture and Algorithms within ICENI. In *UK e-Science All Hands Meeting*, August 2003.