Using ICENI to run parameter sweep applications across multiple Grid resources

M. Y. Gulamali, A. S. McGough, S. J. Newhouse, and J. Darlington

London e-Science Centre, Imperial College London, London, UK.
http://www.lesc.ac.uk/

Abstract

The Imperial College e-Science Networked Infrastructure (ICENI) provides an end-to-end Grid middleware to facilitate Grid administrators, developers and end-users. It consists of both a service oriented Grid framework and an application toolkit, using a component programming model to represent Grid applications. In this paper we give an overview of ICENI from an application user's point of view, and demonstrate how one may use it to run parameter sweep applications across multiple Grid resources. In particular, we identify the Grid ENabled Integrated Earth system model (GENIE) as an exemplar application that can benefit from running on multiple resources through ICENI and, subsequently, promote the sharing of computational resources between the institutions involved in the project.

1 Introduction

Computational Grids are envisaged as the solution to large scale, distributed computing problems that involve significant compute, network and/or storage requirements, providing on-demand, always on, utility computing capabilities (e.g. see [19]). They consist of a collection of electronic resources such as computers, networks, on-line instruments, data servers and/or sensors etc., that are bound together by a common set of protocols, services and tools – a *Grid middleware* (e.g. see [8]) – that allow the resources to be viewed as a seamless computing and information environment by end users. This concept also introduces the notion of "virtual organisations" (VOs) whereby different organisations can contribute different expertise and resources to different projects, creating a more efficient and productive environment of collaboration (e.g. see [6]).

To this end the Imperial College e-Science Networked Infrastructure (ICENI) has been developed by the London e-Science Centre [12] in order to facilitate Grid administrators, developers and end-users. ICENI provides an end-to-end Grid middleware that contains interfaces that are compliant with the Open Grid Services Infrastructure (OGSI) [7], and consists of both a Grid service framework and an application toolkit [9].

In this paper we demonstrate how ICENI may be used to efficiently run parameter sweep applications across multiple Grid resources, providing the capability of true resource brokering. In particular, we describe our efforts of using ICENI in the Grid ENabled Integrated Earth system model (GENIE) project to enable a group of environmental scientists, from multiple institutions, to share compute resources and collaborate on a particular line of scientific investigation.

We begin with an overview of the GENIE project, and the motivation for the work documented herein, in Section 2. We then introduce the reader to ICENI as a service oriented middleware, and describe some of the core services and tools that are provided by ICENI to allow users to develop and execute Grid applications (Section 3). In Section 4 we illustrate how the GENIE parameter sweep experiments discussed in Section 2.2 may be modeled as a componentised ICENI application, and show how such a component model can be adapted to allow the experiment to run on high-throughput resources. Finally, we provide a summary of our work and conclude with a brief description of our future efforts (Section 5).

2 The GENIE project

2.1 Background

Earth System Models (ESM) are used by environmental scientists to simulate the long term evolution of the Earth's climate by coupling together individual models of the climate system. The constituents often include models for the Earth's atmosphere, ocean, sea-ice, marine sediments, land surface, vegetation and soil, hydrology, ice sheets and the biogeochemical cycling within and between components. Due to the large number of components involved, current ESMs tend to be highly idealised with reduced dimensionality and/or low spatial resolution (e.g. see [18]), or else tend to be too computationally demanding for long-term or ensemble simulations (e.g. see [2]).

Funded by the UK's Natural Environment Research Council (NERC) as part of their e-Science programme, the GENIE project intends to overcome the limitations described above by leveraging the advantages of Grid based computing (e.g. see [1]). The project aims to deliver a Grid-based, modular, distributed and scalable ESM for long-term and paleo-climate studies to the environmental sciences community, with the focus on simulating the (geologically) recent ice-age cycles and the future response of the Earth system to human activities, including global warming.

2.2 Previous work

Initial scientific work in the project was carried out using a prototype ESM (aka *c-GOLDSTEIN* [4]) comprised of a 3-dimensional (frictional geostrophic) ocean model coupled to a (dynamic and thermodynamic) sea-ice model and a 2-dimensional (energy-moisture balance) atmosphere model. This involved running a series of parameter sweep experiments to address the vulnerability of the thermohaline circulation of the world ocean, where each experiment consisted of approximately 10^3 individual instances of the ESM.

The Grid computing infrastructure developed for these experiments consisted of a flocked Condor pool [22] composed of approximately 200 compute nodes, housed at the London e-Science Centre [12], the Department of Computing at Imperial College London [3], and the Southampton Regional e-Science Centre [20]. A web-based portal was used to facilitate the creation, deployment and management of each experiment, while a data management system based on the Geodise Database Toolkit [10] was used to manage the large volume of data produced by the experiments. A technical account of this infrastructure is given by Gulamali *et al.* [11], while Marsh *et al.* [13] discuss the preliminary results of the parameter sweep experiments in a scientific context.

2.3 Motivation

One of the constraints that were identified in the initial work with GENIE was the lack of any type of resource brokering. In particular, while members of the GENIE project were able to carry out their parameter sweep experiments on the available Condor resources, they were unable to commit their own, locally administered, computing resources to the experiments. In this paper we discuss one possible solution to this and describe how ICENI may be used to efficiently broker an application such as the GENIE parameter sweep experiments [11, 13] on a Grid composed of multiple computational resources.

3 The ICENI middleware

3.1 A service oriented architecture

The Imperial College e-Science Networked Infrastructure (ICENI) is a Grid middleware that provides a dynamic service management framework to aid resource administrators, application developers, and end-users to manage and use Grid environments. ICENI represents compute, storage and software resources as services that can inter-operate using standard protocols (e.g. Jini, SOAP, JXTA), and moreover be used to encapsulate the capabilities, availability and behaviour of a resource. Consequently, resources may be shared using a service level agreement (defined by the respective resource administrators) to create a federated service oriented computational Grid. Readers are referred to Furmento *et al.*, [9], for the technical details of ICENI as an service oriented middleware.

ICENI uses a component programming model to describe Grid applications. This is clearly beneficial because it promotes code reuse and reduces the task of Grid application development to that of application composition (e.g. see [14]). Each component in an application represents an abstract or running software resource service that can communicate to other components in the application through the ICENI middleware.

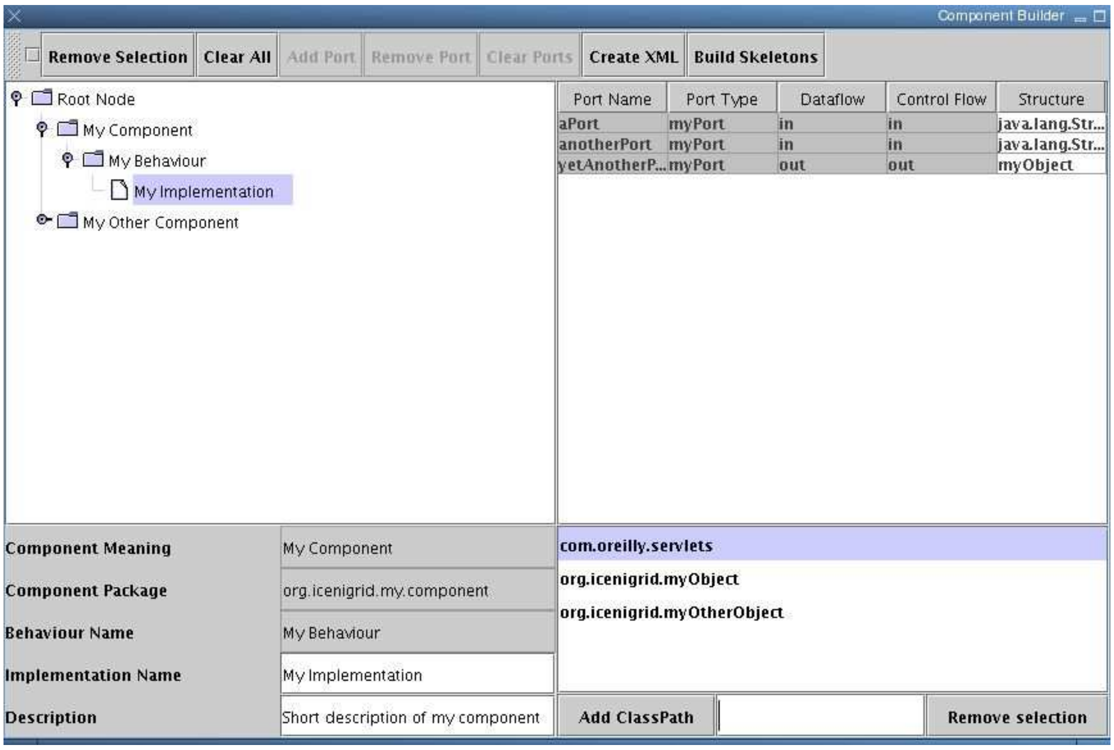
ICENI is rich in metadata which is preserved at all levels within the system. This includes: metadata about how each component in a particular application works, as provided by the component developer; performance characteristics, stored from previous runs of the component within the ICENI environment; and metadata (both static and dynamic) provided by the Grid resources which are available for use by components in an application. Two of the core services provided by the middleware, which make use of this rich metadata environment, as well as being important in the context of the discussion herein, are the *Launching Framework* and the *Scheduling Framework*. The following subsections give an overview of these services, as well as a brief description of the application development and submission environment for ICENI, and the ICENI solution for componentising binary executable applications.

3.2 The Scheduling Framework

By making use of the componentised nature of ICENI applications, along with the rich metadata held within the system and the workflow of the given application, the Scheduling Framework service within ICENI is capable of finding an efficient deployment of the components of an application over a subset of the available resources.

ICENI applications are described in terms of a dataflow oriented workflow document called the *Execution Plan* (EP). The EP defines which components an application will be constructed from, along with how data will flow between these components. The Scheduling Framework takes an abstract description of the workflow, in which the type of component (the "meaning" of the component – see [15]) is defined but not it's implementation, and, as the first stage of the enactment process, determines an efficient deployment of the components in the workflow over a subset of the available resources using a specific set of implementations. The reader is referred to Mayer *et al.* [15] for a more comprehensive account of this process.

Schedulers are pluggable into the ICENI middleware and provide the means to choose the set of implementations and resources. A number of schedulers have been studied within the framework, including random, best of n random, simulated annealing and game theory schedulers (see Young *et al.*, [23]). The schedulers can be aware of the workflow of an application in which case they take account of all the components in the application rather than deploying each one separately. Schedulers can also make use of functionality within the Scheduling Framework service in order to perform their tasks. These features include: the *application mapper*, which can locate implementations of the components to be used; the *identity manager*, which determines if a user is allowed to access specific resources or code; and the *performance repository* which provides estimates for the execution times



Port Name	Port Type	Dataflow	Control Flow	Structure
aPort	myPort	in	in	java.lang.Str...
anotherPort	myPort	in	in	java.lang.Str...
yetAnotherP...	myPort	out	out	myObject

Component Meaning	My Component
Component Package	org.icenigrid.my.component
Behaviour Name	My Behaviour
Implementation Name	My Implementation
Description	Short description of my component

com.oreilly.servlets
org.icenigrid.myObject
org.icenigrid.myOtherObject

Figure 1: A screenshot of the ICENI ComponentBuilder.

for application components within a given workflow.

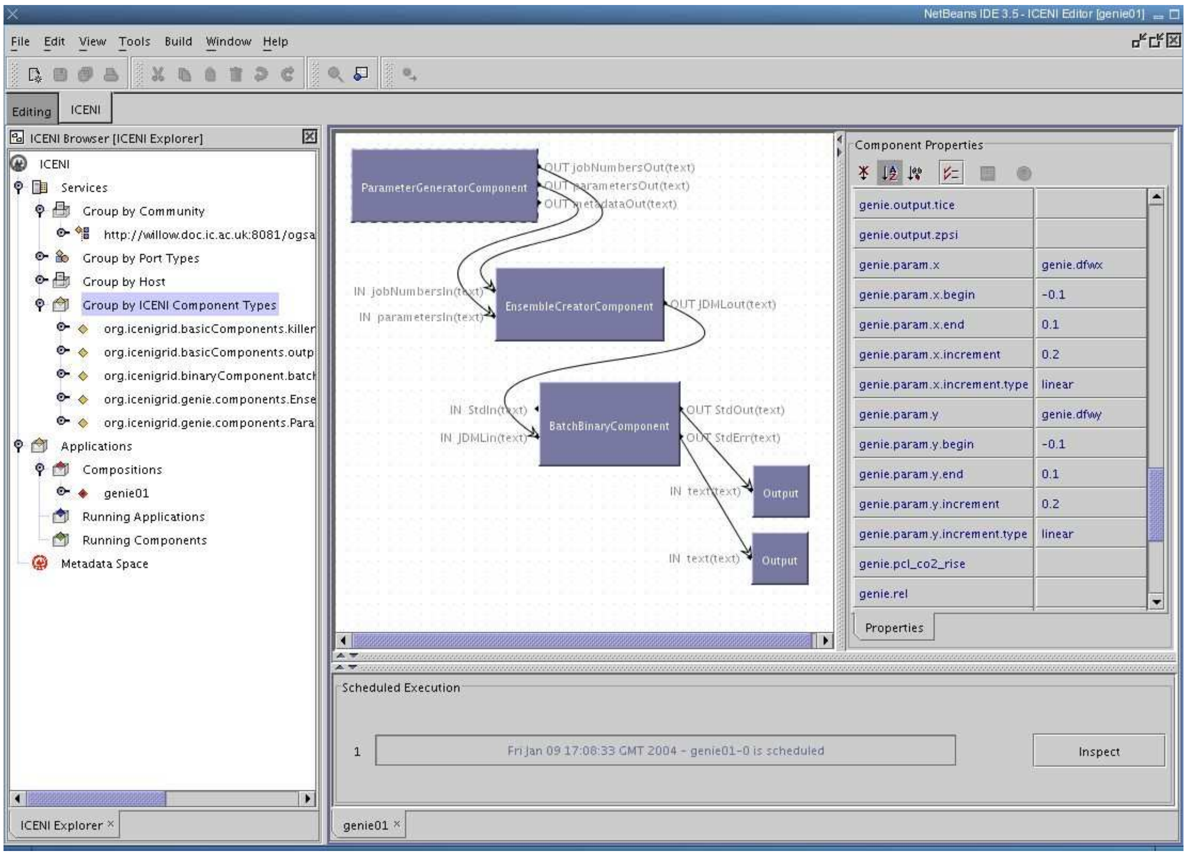
The ordering of schedules is a subjective matter. ICENI uses metrics defined by the user and the resource owners to define the policy for selecting the ordering of schedules. This may comprise of such things as optimising over execution time and/or resource cost in the case where resources have prices. Further information can be found in Young and Darlington [24].

Once the subset of resources for an application to run over has been determined, the Scheduling Framework in ICENI will generate a *Job Description Markup Language* (JDML) [16] document for each resource used. Each document describes how ICENI may deploy an application component onto the particular resource it has been allocated to. The following subsection provides a brief overview of how this entails ICENI to execute an entire application on a set of computational resources.

3.3 The Launching Framework

In order to deploy work onto Grid resources, ICENI uses a Launching Framework service. This provides two functions: that of advertising the resource(s) available through the launcher; and that of taking a JDML document, converting it into a locally understood format, and executing it upon the appropriate resource. There may be many Launching Framework services within an ICENI based Grid, with each service representing one or more resources on that Grid. Pluggable launchers are attached to each of these frameworks in order to translate the JDML down to a native format that can be submitted to the appropriate resource through either a Distributed Resource Manager (DRM) or execution on the local resource. We have been working with launchers for fork (i.e. Shell script), and the following DRMs: Condor, Globus Toolkit 2 (GT2) [5] and Sun Grid Engine (SGE) [21].

The Launching Framework is responsible for staging any files that may be required for job execution to the resource, and staging any appropriate files back afterwards. The framework is also responsible for monitoring the running jobs and reporting back if they terminate abnormally.



genie.output.tice	
genie.output.zpsi	
genie.param.x	genie.dfwx
genie.param.x.begin	-0.1
genie.param.x.end	0.1
genie.param.x.increment	0.2
genie.param.x.increment.type	linear
genie.param.y	genie.dfwy
genie.param.y.begin	-0.1
genie.param.y.end	0.1
genie.param.y.increment	0.2
genie.param.y.increment.type	linear
genie.pcl_co2_rise	
genie.rel	

1 Fri Jan 09 17:08:33 GMT 2004 - genie01-0 is scheduled

Figure 2: A screenshot of the ICENI Netbeans Client.

3.4 The Grid Container

The final stage of enactment in the ICENI model is the *Grid Container*, which is responsible for starting each of the components that are to be deployed onto that resource and the communication between all components within the application. The Grid Containers are required to discover each other and pass inter-component communications between the components. The Grid Container is also responsible for generating timing information which is collected by the performance repository to improve predictions for future use of the components.

3.5 Application development in ICENI

In order to facilitate the development of components that represent software resources services in ICENI, a *ComponentBuilder* application has been created. This allows an end-user to specify the ports of their component according to the meaning, behaviour and implementation of the component (e.g. see [14]) in a systematic and graphical way (Figure 1). Having done this, the ComponentBuilder can generate the necessary metadata files to allow the component to be expressed as a software resource service in ICENI. Furthermore, a user may also opt to generate a set of skeleton Java source code classes, which can be completed to produce a working Java implementation of their component.

As we have already mentioned, the component programming model employed by ICENI reduces the task of Grid application development to that of component composition. Once a set of components have been created using the ComponentBuilder described above, and have been compiled and deployed on ICENI as a set of software resource services, they may be composed together to form a Grid application. To facilitate this, a Netbeans based [17] client has been developed (Figure 2). The client allows end-users to browse and monitor available services upon ICENI (the left handle panel in

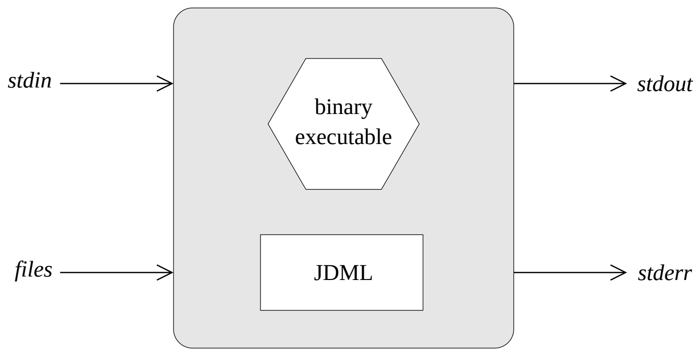


Figure 3: A schematic representation of a typical Binary Component in ICENI. Arrows depict the direction of dataflow in/out of the component. See text for details.

the figure). It also provides an intuitive way for applications to be composed, whereby components can be dragged-and-dropped onto a composition pane (the central region of Figure 2), before being connected together to visually describe the workflow of the application. The composed application can then be submitted and launched onto ICENI through Netbeans in order to execute it.

3.6 Binary Components

Most of the components used in the GENIE application were developed independently from ICENI and work as separate binary applications. Work is underway to develop integrated ICENI implementations of these components which are fully integrated components. However, in the short term, the binary applications can be accessed through the use of the *Binary Component*.

The Binary Component is a way of wrapping up an existing application to use within the ICENI framework. This is especially useful when an application exists as a single non-componentised binary executable, or when the application consists of legacy code for a specific computational architecture. The use of the Binary Component entails that the application is run from within an ICENI component with that component providing the necessary metadata required to schedule and launch it using the frameworks described in the previous subsections.

A schematic representation of a typical Binary Component is shown in Figure 3. Every Binary Component is associated with the binary executable that the component represents, and a JDML file which describes how the application is to be executed and the arguments that it may take. The Binary Component is capable of taking a number of input and output data from other components in ICENI (depicted by the arrows in the figure). This allows a set of arguments to be passed to the binary executable (through the *stdin* port), or the output of the application to be passed back to ICENI (through the *stdout* and *stderr* ports). A list of files to send to the application prior to execution and a list of files to return from the application after completion may also be sent to the Binary Component (through the *files* port) to incorporate in the component's JDML.

4 Executing parameter sweep experiments on multiple resources

4.1 GENIE as an ICENI application

The ICENI framework and tools described in the previous section allowed us to represent the GENIE ESM as a Binary Component. We were then able to use *splitter* and *collator* components (e.g. see [14]), together with some custom components to handle input and output data (see below), to perform the GENIE parameter sweep experiments described in Section 2.2 through ICENI. The workflow for these experiments is illustrated in Figure 4.

The parameter sweep experiment in ICENI begins with a *setup component* which initialises the Grid compute resources for each of the individual ESMs, creating the necessary data for the input files

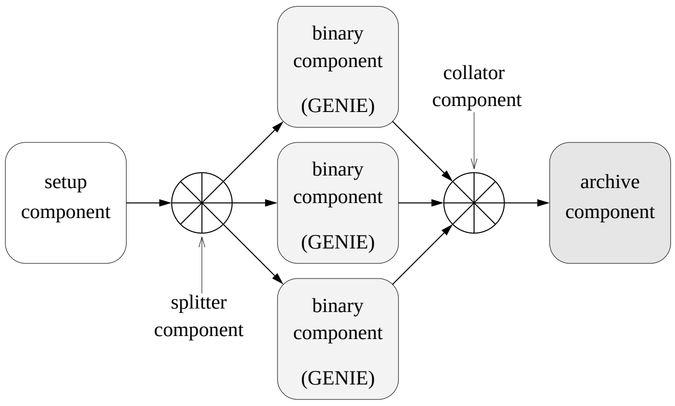


Figure 4: The GENIE parameter sweep experiment as a component-based application. Arrows describe the direction of control and data fbw between components.

using parameters chosen by the user at run time. This data is passed to a splitter component which delegates the data to multiple Binary Components (only three have been shown in Figure 4 for sake of clarity). As each ESM finishes execution, the Binary Component associated with that ESM passes the resultant data to a collator component, which passes it to an *archive component* that archives the data in a fashion chosen by the user.

The advantages of this type of application model is immediately obvious. Different ESMs may be substituted into the application at design time (i.e. when the application is being created by the user with the ICENI Netbeans client). Moreover, the setup or archive components may also be replaced with ones of the user's choosing, and the entire parameter sweep application re-run without much more work involved (provided that other implementations of these components already exist).

However, the Launchers developed so far for ICENI have been designed for the deployment of single jobs onto resources. This does not take into account the high-throughput DRM systems such as Condor and SGE, in which a large number of jobs may be submitted from a single command. Although the current Condor and SGE Launchers can perform the task of submitting large numbers of jobs, each job is launched separately onto the DRM system with a separate JDML document. This is not only an inefficient use of the scheduling/launching mechanism within ICENI, but it also removes the ability to make use of any DRM functionality for high-throughput tasks.

We propose, and have developed here, a new set of ICENI launchers which are capable of submitting multiple jobs through a single JDML submission. This allows the scheduler to place a number of components efficiently onto a single DRM. The launchers are discussed in more detail in the next subsection.

4.2 GENIE on multiple resources

Figure 5 shows the workflow for a GENIE parameter sweep experiment that employs our new high-throughput launchers (the *launcher components*). In particular, the figure shows two different launcher components, each of which may be hosted on a different resource, possibly employing different DRM systems.

As with Figure 4, the setup component in Figure 5 is responsible for creating the necessary input files for the parameter sweep experiment, and the archive component is responsible for archiving the resultant data from the individual GENIE ESMs. However, in this scenario the splitter component delegates data to our new launcher component, which creates the necessary JDML and job submission data to run the parameter sweep application (or a subset of it) as a single job (a single Binary Component) on a high-throughput DRM system, possibly chosen by the end-user during the application composition stage in the ICENI Netbeans client.

Moreover, the componentised nature of the this application allows the splitter and collator components to be used to submit subsets of the experiment to different Grid resources. For example, with

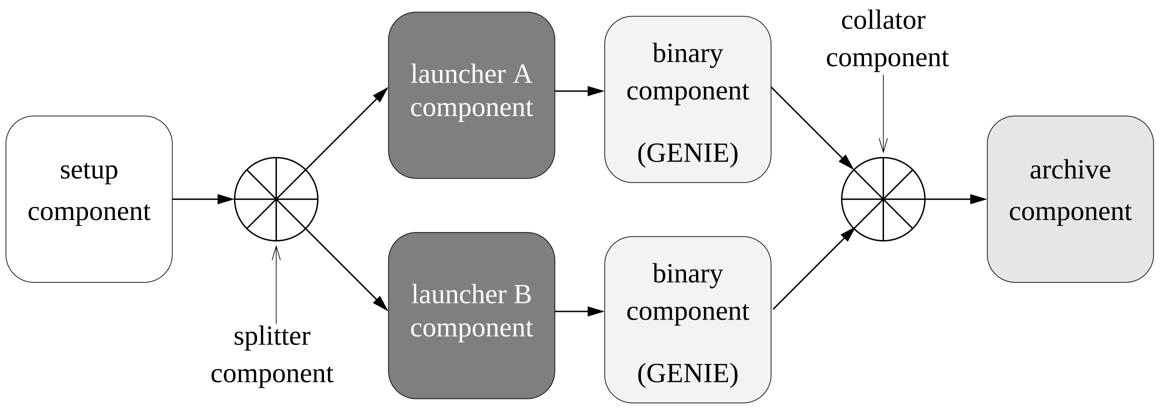


Figure 5: The GENIE parameter sweep experiment across multiple launchers chosen by the user at design time. Arrows describe the direction of control and data fbw between components.

respect to Figure 5, the *launcher A component* might serve to submit a subset of jobs (possibly representing a subset of the parameter space under investigation) to a flocked Condor pool similar to the one described in Section 2.2. Meanwhile the *launcher B component* might submit the remaining jobs to a dedicated computing resource (e.g. a Beowulf Cluster) located and managed by the institution performing the parameter sweep experiment. Thus the scientists on the GENIE project, from multiple institutions, may share compute resources and efficiently collaborate, through ICENI.

5 Summary and conclusions

In this paper we have given an overview of the ICENI middleware from an end-user perspective and demonstrated how it may be used to run a parameter sweep application, such as those found in the GENIE project, across multiple Grid compute resources. We began by giving an overview of the GENIE project and highlighted the work that has already been carried out in the project. We then introduced the reader to the ICENI middleware and discussed the services and tools that it provides to end-users and application developers. Thereafter we illustrated how a GENIE parameter sweep experiment may be modeled as a componentised ICENI application, and showed an adaptation of this model to allow high-throughput DRMs to be used efficiently in ICENI. Thus we were able to demonstrate how ICENI can promote efficient resource sharing among members of a VO.

Our test implementations of the GENIE parameter sweep experiment described in Section 4.2, using Condor and SGE launchers, together with the simpler fork launchers, have shown that the workflows described above can indeed be executed in ICENI. However, at the time of writing our results are incomplete and consequently they have not been discussed in any detail here. We hope to show them at the Applications Workshop for which this paper is a precursor.

We are currently attempting to integrate the high-throughput launching components with the core ICENI middleware so they may be used without the user having to explicitly express them in their application workflow. We also aim to develop high-throughput aware schedulers and splitter components, such that ICENI is capable of splitting up data to these high-throughput launchers.

Another, more generic, way of implementing the multiple launching capabilities that we described in Section 4.2 is to build some form of intelligence into the Scheduling Framework, such that when a scheduler recognises a workflow pattern similar to that in Figure 4, it automatically deduces that the application is a parameter sweep application and can schedule and launch it upon the appropriate high-throughput DRM automatically as a single job. The development of such a Scheduling Framework and scheduler is also underway.

References

[1] Berman, F., and A. Hey, The Scientific Imperative, in *The Grid 2: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselmann, pp. 13-24, Elsevier, San Francisco, California, USA, 2004.

[2] Cox, P. M., R. A. Betts, C. D. Jones, S. A. Spall, and I. J Totterdell, Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model, *Nature*, **408**, 184-187, 2000.

[3] Department of Computing, Imperial College London, London, UK.: `http://www.doc.ic.ac.uk/`

[4] Edwards, N. R., and R. Marsh, An efficient climate model with three-dimensional ocean dynamics, *Clim. Dyn.*, submitted.

[5] Foster, I., and C. Kesselman, Globus: A metacomputing infrastructure toolkit, *Intl. J. Supercomputer Applications*, **11**, 115-128, 1997.

[6] Foster, I., C. Kesselmann, and S. Tuecke, The anatomy of the Grid: Enabling scalable virtual organizations, *Intl. J. Supercomputer Applications*, **15**, 200-223, 2001.

[7] Foster, I., C. Kesselmann, J. Nick, and S. Tuecke, The physiology of the Grid: An Open Grid Serivces Architecture for distributed systems integration, Open Grid Service Infrastructure Working Group, Global Grid Forum, June 22, 2002.

[8] Foster, I., and C. Kesselmann, Concepts and architecture, in *The Grid 2: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselmann, pp. 37-63, Elsevier, San Francisco, California, USA, 2004.

[9] Furmento, N., W. Lee, A. Mayer, S. Newhouse, and J. Darlington, ICENI: An Open Grid Service Architecture implemented with Jini, in *SuperComputing 2002*, Baltimore, Maryland, USA, 2002.

[10] Grid Enabled Optimization and DesIgn Search for Engineering (GEODISE): `http://www.geodise.org/`

[11] Gulamali, M. Y., T. M. Lenton, A. Yool, A. R. Price, R. J. Marsh, N. R. Edwards, P. J. Valdes, J. L. Wason, S. J. Cox, M. Krznaric, S. Newhouse, and J. Darlington, GENIE: Delivering e-Science to the environmental scientist, in *Proc. UK e-Science All Hands Meeting 2003*, pp. 145-152, Nottingham, UK, 2003.

[12] London e-Science Centre, Imperial College London, London, UK.: `http://www.lesc.ic.ac.uk/`

[13] Marsh, R. J., A. Yool, T. M. Lenton, M. Y. Gulamali, N. R. Edwards, J. G. Shepherd, M.Krznaric, S. Newhouse, and S. J. Cox, Bistability of the thermohaline circulation identified through comprehensive 2-parameter sweeps of tan efficient climate model, *Clim. Dyn.*, submitted.

[14] Mayer, A., S. McGough, M. Gulamali, L. Young, J. Stanton, S. Newhouse, and J. Darlington, Meaning and behaviour in Grid oriented components, in M. Parashar, editor, *Grid Computing – GRID 2002: Third International Workshop, Baltimore, MD., USA., November 18, 2002. Proceedings, Lecture Notes in Computer Science*, **2536**, 100-111, 2002.

[15] Mayer, A., S. McGough, N. Furmento, W. Lee, S. Newhouse, and J. Darlington, ICENI dataflow and workflow: Composition and scheduling in space and time, in *Proc. UK e-Science All Hands Meeting 2003*, pp. 627-634, Nottingham, UK, 2003.

[16] McGough, A. S., A common Job Description Markup Language written in XML: `http://www.lesc.ic.ac.uk/projects/jdml.pdf`

[17] Netbeans IDE: `http://www.netbeans.org/`

[18] Petoukhov, V., A. Ganopolski, V. Brovkin, M. Claussen, A. Eliseev, C. Kubatzki, and S. Rahmstorf, CLIMBER-2: A climate system model of intermediate complexity. Part I: Model description and performance for present climate, *Clim. Dyn.*, **16**, 1-17, 2000.

[19] Smarr, L., Grids in context, in *The Grid 2: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselmann, pp. 3-12, Elsevier, San Francisco, California, USA, 2004.

[20] Southampton Regional e-Science Centre, University of Southampton, Southampton, UK.: `http://www.e-science.soton.co.uk/`

[21] Sun Grid Engine Software: `http://wwws.sun.com/software/gridware/`

[22] Thain, D., T. Tannenbaum, and M. Livny, Condor and the Grid, in F. Berman, A. J. G. Hey, and G. Fox, editors, *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley, Chichester, UK, pp. 299-335, 2003.

[23] Young, L. R., S. McGough, S. Newhouse, and J. Darlington, Scheduling architecture and algorithms within the ICENI Grid middleware, in *Proc. UK e-Science All Hands Meeting 2003*, pp. 5-12, Nottingham, UK, 2003.

[24] Young, L. R., and J. Darlington, Scheduling componentised applications on a computational Grid, MPhil/PhD Transfer Report, Imperial College London, University of London, 2004.