Performance guided scheduling in GENIE through ICENI

M. Y. Gulamali[1], A. S. McGough[1], R. J. Marsh[2], N. R. Edwards[3], T. M. Lenton[4]
P. J. Valdes[5], S. J. Cox[6], S. J. Newhouse[1], J. Darlington[1], and the GENIE team.

http://www.genie.ac.uk/

Abstract

Initial work in the Grid ENabled Integrated Earth system model (GENIE) project involved a series of parameter sweep experiments using a Grid infrastructure consisting of a flocked Condor pool, a web service oriented data management system and a web portal. In this paper we introduce the Imperial College E-Science Networked Infrastructure (ICENI) Grid middleware, and describe how it can be used to increase the efficiency of GENIE parameter sweep experiments. We perform several experiments using a combination of different computational resources and different job deployment mechanisms. Our results suggest that ICENI does not produce any significant overhead in the sojourn time of a GENIE parameter sweep experiment and can promote the sharing of computational resources between institutions.

1 Introduction

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 of an ESM can 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 Petoukhov *et al.*, [18]), or else tend to be too computationally demanding for long-term or ensemble simulations (e.g. see Cox *et al.*, [2]).

The Grid ENabled Integrated Earth system model (GENIE) project [8] intends to overcome the limitations described above by leveraging the advantages of Grid based computing (e.g. see Berman and Hey, [1]). The project aims to provide the environmental sciences community with a Grid-based, modular, distributed and scalable ESM for long-term and paleo-climate studies, 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.

Initial work in GENIE 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 was used to test hypotheses concerning the influence of freshwater transport upon the global ocean circulation and consisted of several parameter sweep experiments. Each experiment consisted of approximately 10^3 individual runs of the prototype ESM and consequently the entire investigation would have taken several years to complete in the absence of any e-Science support.

A Grid computing infrastructure was developed for these experiments and consisted of a flocked Condor pool [21] 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 [19]. The presence of institutional firewalls required the designation and utilisation of port ranges specified by the Condor and firewall administrators at these institutions.

The creation, deployment and management of each experiment upon Condor was facilitated by a web-based portal, while a data management system based on the Geodise Database Toolkit [9] 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.*, [10], while Marsh *et al.*, [13], discuss the results of the parameter sweep experiments in a scientific context.

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

[1]London e-Science Centre, Imperial College London, London, UK.
[2]Southampton Oceanography Centre, Southampton, UK.
[3]Physics Institute, University of Bern, Switzerland.
[4]School of Environmental Sciences, University of East Anglia, Norwich, UK.
[5]School of Geographical Sciences, University of Bristol, Bristol, UK.
[6]Southampton Regional e-Science Centre, Southampton University, Southampton, UK.

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 using the Imperial College e-Science Networked Infrastructure (ICENI) Grid middleware [11], and demonstrate how it may be used to efficiently run experiments across multiple resources, providing the capability of true resource brokering.

We begin in Section 2 by giving an overview of the ICENI middleware. We then describe how a GENIE parameter sweep experiment may be represented as an ICENI application (Section 3), and introduce a new ICENI component that allows experiments to be submitted to Grid resources hosting high-throughput Distributed Resource Managers (DRMs). We use this component to perform several GENIE parameter sweep experiments through ICENI and discuss the performance overhead introduced by ICENI (Section 4). We conclude in Section 5 with a summary of our findings and an outline for future work.

2 The ICENI Grid middleware

2.1 A service oriented architecture

The ICENI Grid middleware 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 can 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.*, [7], 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 Mayer *et al.*, [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.

2.2 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 Mayer *et al.*, [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.*, [22]). 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 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 i.e. optimisation over execution time and/or resource cost, etc. Further information can be found in Young and Darlington, [23].

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.

2.3 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 into a native format that can be submitted to the appropriate resource through either a 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 [6] and Sun Grid Engine (SGE) [20].

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.

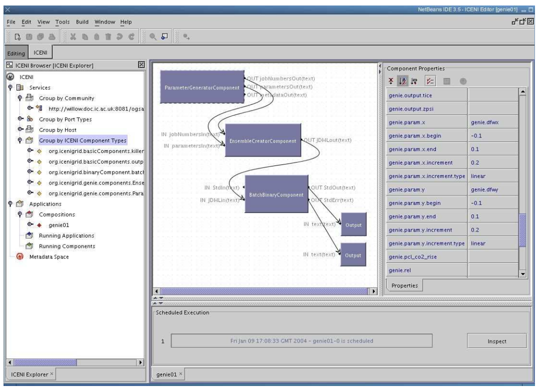
2.4 The Grid Container

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.

2.5 Application development in ICENI

In order to facilitate the development of components that represent software resources in ICENI, a *Component Builder* 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 Mayer *et al.*, [14]) in a systematic and graphical way. Having done this, the Component Builder 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 Component Builder 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 1). The client allows end-users to browse and monitor available services upon ICENI (the left hand panel in 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 1), 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.



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 1: A screenshot of the ICENI Netbeans Client.

2.6 Binary Components

Most of the components used in the GENIE application were developed independently from ICENI and run as separate binary applications. Work is underway to develop 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 2. 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.

The Binary Component is particularly useful in the context of the work presented herein because it allows us to wrap up the GENIE prototype ESM mentioned in Section 1 as an ICENI component. The following section describes how this may be used to perform parameter sweep experiments through ICENI.

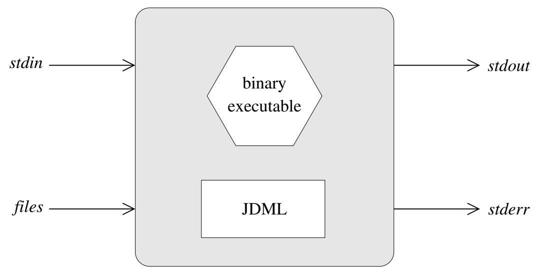


Figure 2: A schematic representation of a typical Binary Component in ICENI. Arrows depict the direction of dataflow in/out of the component. See text for details.

3 GENIE as an ICENI application

A GENIE parameter sweep experiment may be represented in ICENI as a number of components that communicate in a manner such as to capture the workflow of the experiment (as illustrated in Figure 3). A *Setup Component* initialises the experiment, creating the necessary input files for the parameter sweep, using parameters chosen by the user at run time. This passes data to a *Broadcast Component* which delegates the data to multiple Binary Components (only 3 are shown in the figure), each of which execute the GENIE model. As each simulation finishes, the Binary Component passes the resultant data to a *Funnel Component*, which passes it to an *Archive Component* that handles the archiving of the resultant data.

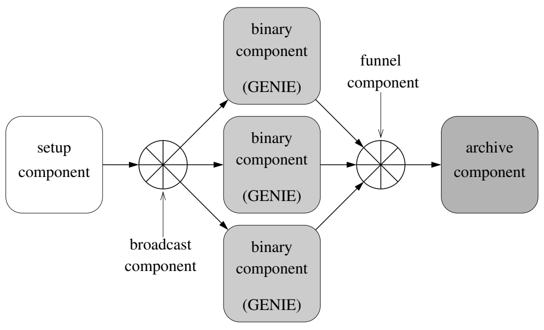


Figure 3: A GENIE parameter sweep experiment as a component-based application. Arrows describe the direction of control and data flow between components.

This application model has the advantage of being both flexible and extensible. Different GENIE models may be substituted into the application during design time (i.e. when the application workflow is being composed using the Netbeans client for ICENI). Moreover, the Setup and Archive Components may also be replaced with ones of the user's own choosing, and the entire parameter sweep experiment re-run without much more work involved (provided that other implementations of these components already exist).

However, a clear disadvantage of the component model presented in Figure 3 is that it does not scale as the number of Binary Components in the application are increased i.e. a typical GENIE parameter sweep experiment involving $\sim 10^3$ individual runs would potentially require $\sim 10^3$ individual Binary Components. Moreover, the Launchers developed so far for ICENI have been designed for the deployment of single jobs onto resources. These do not utilise the high-throughput feature of DRM systems such as Condor or SGE, in which a large number of jobs may be submitted with a single command. In particular, while the Condor and SGE Launchers in ICENI can perform the task of submitting a set of jobs to their respective resources, each job in the set is launched onto the DRM system as a separate task. This is not only inefficient use of the scheduling/launching mechanism within ICENI, but it also reduces the advantages of using a DRM system to execute high-throughput tasks.

We have addressed these issues by developing a new ICENI component, the *GENIE Launcher Component*, that is capable of submitting a GENIE parameter sweep experiment to a single resource, and possibly launching them as a single task on a high-throughput DRM system (illustrated in Figure 4). The introduction of this component into an application workflow can greatly reduce the complexity of the workflow (cf. Figure 3). Data from the Setup Component is received by the GENIE Launcher Component which creates the necessary files in order to submit the parameter sweep experiment to a DRM system chosen by the user during the application composition phase. This component then sends data to a single Binary Component in order to submit all the jobs in the parameter sweep experiment as a single task to the DRM system. Upon completion, once all of the simulations in the parameter sweep experiment have completed, the resultant data is sent to the Archive Component as before.

The results of several GENIE parameter sweep experiments which utilise the new ICENI component are discussed in the following section.

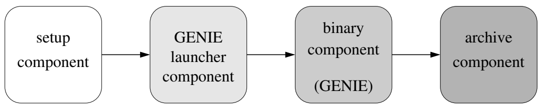


Figure 4: A GENIE parameter sweep experiment as a component-based application with the new GENIE Launcher Component (see text for details). Arrows describe the direction of control and data flow between components.

4 Performance experiments

4.1 Methodology

We ran eight different types of GENIE parameter sweep experiments using the new GENIE Launcher Component described above. These are summarised in Table 1. Each experiment consisted of a series of 9 individual simulations configured to run for 500 years of model time from a uniform cold state. In each of the 9 simulations the surface freshwater flux between the Atlantic ocean and the Pacific Ocean (hereafter denoted as `DFWX`), and the overall atmospheric moisture diffusivity (hereafter denoted as `DIFF`), were varied with the following quantities in (`DFWX`,`DIFF`) parameter space:

$$\begin{array}{lll}(-0.3,\,5\times10^4), & (-0.3,\,5\times10^5), & (-0.3,\,5\times10^6),\\ (\ \ 0.0,\,5\times10^4), & (\ \ 0.0,\,5\times10^5), & (\ \ 0.0,\,5\times10^6),\\ (\ \ 0.3,\,5\times10^4), & (\ \ 0.3,\,5\times10^5), & (\ \ 0.3,\,5\times10^6).\end{array}$$

Here `DFWX` is measured in units of Sv (1 Sv = 10^6 m^3s^{-1}) and `DIFF` is measured in units of m^2s^{-1}.

The experiments were run on two different types of computational resources maintained by the London e-Science Centre:

1. a Solaris based shared memory server with 8 $\times$ 900MHz UltraSparc II CPUs and 16Gb of memory (hereafter referred to as the "Solaris resource"),
2. a Linux based Beowulf cluster consisting of $16\times$2GHz Intel Dual Xeon CPUs and 2Gb of memory (hereafter referred to as the "Linux resource").

Both resources were behind the same institutional firewall but did not share a common filesystem.

Experiments were run as sequential Bash jobs, or submitted to Condor pools on each of the resources above as Condor jobs. The Condor pool on the Solaris resource consists of 28 processors, with 20 of these processors being flocked from a similar Solaris resource being maintained by the London e-Science Centre. The Condor pool on the Linux resource consists of just 2 processors. Both pools are configured to run jobs on their respective processors when CPU activity on them is low.

GENIE parameter sweep experiments were either run at the command line as a shell script (hereafter referred to as "Non-ICENI") or through the ICENI middleware (hereafter referred to as "ICENI"). Where ICENI was used to perform experiments, the ICENI server was running on the Solaris resource with Launching Framework services on both the Solaris and the Linux resources. GENIE Binary Component implementations were provided for both OS architectures.

Resource	Job Type	Middleware
Solaris	Bash	Non-ICENI
		ICENI
	Condor	Non-ICENI
		ICENI
Linux	Bash	Non-ICENI
		ICENI
	Condor	Non-ICENI
		ICENI

Table 1: Different types of GENIE parameter sweep experiments carried out. See text for details.

Each of the eight different types of GENIE parameter sweep experiments were run several times in order to obtain an average sojourn time i.e. the time taken from the start of a parameter sweep experiment to the end. In those cases where we used ICENI to perform experiments, we included the time taken for ICENI to schedule and launch experiments as well as the time taken for ICENI to return the resultant data to the user and inform them that their experiment had finished.

A ninth experiment was run which involved running a GENIE parameter sweep experiment across the Condor pools on both the Solaris resource and the Linux resource using ICENI, with 4 simulations running on the Solaris resource and 5 simulations running on the Linux resource. It was clearly not possible to perform this experiment through a shell-script at the command line without an alternative Grid middleware solution (e.g. Globus Toolkit 2) due to the different OS architectures and filesystems of the resources used.

4.2 Results and discussion

The mean sojourn times (in hours) for the eight different types of GENIE parameter sweep experiments described above are shown in Table 2.

		Mean sojourn time (hours)	
Resource	**Job type**	**Non-ICENI**	**ICENI**
Solaris	Bash	11.46	11.65
	Condor	2.44	2.49
Linux	Bash	2.50	2.52
	Condor	2.33	2.51

Table 2: Mean sojourn time of GENIE parameter sweep experiments.

Examining the Non-ICENI results in the table we find that the GENIE parameter sweep experiments that were performed as Bash jobs on the Solaris resource took considerably longer than those that ran as Condor jobs on the same resource. This result is as expected because simulations in a Bash job experiment are executed sequentially, whereas those in a Condor job may potentially be executed in parallel.

We notice that experiments that were performed as Bash jobs on the Linux resource were considerably faster than Bash jobs on the Solaris resource. This is due to the different architectures of the two resources, with the processors on the Linux resource being faster (see the previous subsection).

In contrast to the Solaris experiments, we find no significant difference in performance between experiments that were performed as Bash jobs and those that were performed as Condor jobs, on the Linux resource. We believe that this is because the Condor pool on the Linux resource consists of relatively fewer nodes than on the Solaris resource. Consequently, simulations that were running on the Linux Condor pool spent more time in the Condor queue than actually being executed, with little or no opportunity to execute in parallel.

Comparing the ICENI results to the Non-ICENI results in Table 2 we find that, in general, experiments that were run through ICENI took longer to complete than those that were executed through a shell-script at the command line. This extra time can be attributed to the overhead of using ICENI as a Grid middleware. Unlike the Non-ICENI experiments, ICENI was required to schedule and launch parameter sweep experiments onto the compute resources, going through the enactment process described in Section 2. It was also responsible for communicating data between the resources when experiments were run on a different resource to that hosting the ICENI server; copying the necessary files between the file systems; and monitoring the status of the experiments on a regular basis before reporting their completion back to the user.

The relative difference and the absolute percentage difference in mean sojourn time between Non-ICENI and ICENI experiments is given in Table 3. These values give a measure of the overhead of using the ICENI Grid middleware to perform GENIE parameter sweep experiments.

		Difference	
Resource	**Job type**	**Time (min)**	**%**
Solaris	Bash	11.42	1.66
	Condor	2.98	2.03
Linux	Bash	1.08	0.72
	Condor	10.72	7.67

Table 3: The relative difference and the absolute percentage difference in mean sojourn time between Non-ICENI and ICENI experiments.

We find that on the Solaris resource there is a notable variation between the relative time difference for Bash based experiments and Condor based experiments. This is also true for the experiments on the Linux resource. This result was not anticipated because it was expected that the overhead of using ICENI to deploy GENIE parameter sweep experiments on the same resource, with either Bash or Condor execution mechanisms, would be independent of the manner in which simulations in the experiment actually ran i.e. sequentially for Bash jobs and parallel for Condor jobs. Consequently, the observed variation between the relative difference in mean sojourn time for Bash based experiments and

Condor based experiments, on the same resource, is not entirely understood and requires further investigation.

Examining the values for the absolute percentage difference between Non-ICENI experiments and ICENI experiments on the Solaris resource in Table 3 we find that ICENI adds $\sim 2\%$ to the mean sojourn time of an experiment. On the Linux resource ICENI adds $\sim 1\%$ to Bash based experiments and $\sim 8\%$ to Condor based experiments. Although this latter value appears significant, in real terms it corresponds to approximately 11 minutes in an experiment lasting ~ 2.5 hours. We consider these overheads to be insignificant when compared to the advantages of using the ICENI Grid middleware to deploy such high-throughput jobs such as the GENIE parameter sweep experiments.

In particular, our ninth experiment showed that we could relatively easily deploy experiments across multiple heterogenous Grid resources through ICENI. An experiment that we could not perform otherwise. Consequently we use this experiment to demonstrate how the ICENI Grid middleware can bring together separate Condor pools to act as a single pool, as an alternative to "flocking" them together (see Epema *et al.*, [5]). While the enhanced job management features of Condor (i.e. automatic job migration) might not be enabled across the pools, the presence of institutional firewalls would not be an issue (cf. Gulamali *et al.*, [10]).

5 Conclusions and further work

In this paper we introduced the GENIE project and described the work that has already been carried out in the project. We then gave an overview of the ICENI Grid middleware and showed how GENIE parameter sweep experiments may be modeled as a componentised ICENI application. We also presented a new ICENI component to allow us to submit multiple tasks as a single job to a high-throughput DRM through ICENI. We used this component to perform several GENIE parameter sweep experiments through ICENI across different Grid computing resources, and compared the mean sojourn time of these with respect to the mean sojourn time of the same experiments executed in the absence of any Grid middleware. An experiment involving multiple Condor pools was also performed through ICENI.

We found that the ICENI Grid middleware does not introduce any significant performance overhead to the GENIE parameter sweep experiments and can be used to run experiments across multiple heterogenous Grid resources. Thus ICENI allows members of the GENIE project to commit their own computational resources to the experiments, and make more efficient use of them for the aims of the project. Our results also showed that the performance overhead on a GENIE parameter sweep experiment run through ICENI using different launching mechanisms, on the same resource, was variable. We did not expect this result and need to explore it further in order to explain it.

We hope to run several more GENIE parameter sweep experiments through ICENI. At the time of writing we have been able to use the SGE DRM on both the Solaris and Linux resources with our GENIE Launcher Component, but we have not yet run any long running applications such as the GENIE parameter sweep experiments. The results of such experiments would inform us further about the performance overhead on the experiments due to ICENI, as well as allow us to demonstrate how we might be able to use ICENI to "flock" together SGE resources.

We also wish to perform experiments on distributed Grid resources including those computing resources administered and managed by our collaborators on the GENIE project. This effort would be useful from a user perspective and advise us as to how well ICENI performs in the presence of firewalls and significant network latencies.

We are currently attempting to integrate the GENIE Launcher Component with the core ICENI Grid middleware so that the ability, of the component to submit multiple jobs as a single task to a high-throughput DRM, can be exploited without a user having to explicitly use the component in their application workflow. We also aim to develop some form of intelligence into the Scheduling Framework, such that when a scheduler recognises a workflow pattern similar to that in Figure 3, it automatically deduces that it is a parameter sweep experiment and can schedule and launch it upon the most appropriate high-throughput DRM automatically as a single job. We hope to present this work and the other efforts outlined above in a follow-up to this paper.

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, U.S.A., 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, U.K.: `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] Epema, D. H. J., M. Livny, R. van Dantzig, X. Evers, and J. Pruyne, A worlwide flock of Condors: Load sharing among workstation clusters, *Journal of Future Generations of Computer Systems*, **12**, 53-65, 1996.

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

[7] 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, U.S.A., November 2002.

[8] Grid ENabled Integrated Earth system model (GENIE): `http://www.genie.ac.uk/`

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

[10] 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, U.K., 2003.

[11] The Imperial College e-Science Networked Infrastructure (ICENI): `http://www.lesc.ic.ac.uk/iceni/`

[12] London e-Science Centre, Imperial College London, London, U.K.: `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 an 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, U.K., 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] Southampton Regional e-Science Centre, University of Southampton, Southampton, U.K.: `http://www.e-science.soton.co.uk/`

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

[21] 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, U.K., pp. 299-335, 2003.

[22] 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, U.K., 2003.

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