PROCEEDINGS OF SCIENCE

The Application Hosting Environment: Easing the Scientist's Access to the Grid

P. V. Coveney
University College London
E-mail: p.v.coveney@ucl.ac.ukl

R. S. Saksena
University College London
E-mail: r.saksena@ucl.ac.ukl

S. J. Zasada*
University College London
E-mail: stefan.zasada@ucl.ac.ukl

Current grid computing technologies have often been seen as being too heavyweight and unwieldy from an end user's perspective, requiring complicated installation and configuration steps to be taken that are too time consuming for most end users. This has led many of the people who would benefit most from grid technology, namely computational scientists, to avoid using it. In response to this we have developed the Application Hosting Environment, a lightweight, easily deployable environment designed to allow the scientist to quickly and easily run unmodified applications on distributed grid resources. This is done by building a layer of middleware on top of existing technologies such as Globus, and exposing the functionally as web services using the WSRF::Lite toolkit. The scientist can start and manage an application via these services, with the extra layer of middleware abstracting the details of the particular underlying grid resource in use. The scientist can concentrate on performing their scientific investigations, rather than learning how to manipulate the underlying grid middleware.

Lighting the Blue Touchpaper for UK e-Science - Closing Conference of ESLEA Project
March 26-28 2007
The George Hotel, Edinburgh, UK

*Speaker.

PoS(ESLEA)017

1. Introduction

We define grid computing [1] as distributed computing conducted transparently across multiple administrative domains. Fundamental to the inter-institutional sharing of resources in a grid is the grid middleware, that is the software that allows an institution to share its resources in a seamless and uniform way. While many strides have been made in the field of grid middleware technology [2, 3], the prospect of a heterogeneous, on-demand computational grid as ubiquitous as the electrical power grid is still a long way off. Part of the problem has been the difficulty to the end user of deploying and using many of the current middleware solutions, which has led to reluctance amongst many scientists to actively embrace grid technology [4].

Many of the current grid middleware solutions can be characterised as what we describe as 'heavyweight', that is they display some or all of the following features: the client software is difficult to configure or install; the middleware is dependent on lots of supporting software being installed and requires non-standard ports to be opened within firewalls. To address these deficiencies there is now much attention focused on 'lightweight' middleware solutions, such as [5], which attempt to lower the barrier of entry for users of the grid.

2. The Application Hosting Environment

In response to the issues raised above we have developed the Application Hosting Environment (AHE)[1], a lightweight, WSRF [6] compliant, web services based environment for hosting scientific applications on the grid. The AHE allows scientists to quickly and easily run unmodified, legacy applications on grid resources, managing the transfer of files to and from the grid resource and allowing the user to monitor the status of the application. The philosophy of the AHE is based around the fact that very often a group of researchers will all want to access the same application, but not all of them will possess the skill or inclination to install the application on a remote set of grid resources. In the AHE, an expert user installs the application and configures the AHE server, so that all participating users can share the same application.

The AHE focuses on applications not jobs, with the application instance being the central entity. We define an application as an entity that can be composed of multiple computational jobs, for example a simulation that consists of two coupled models which requires two jobs to instantiate it. An application instance is represented as a stateful WS-Resource[6]. Details of how to launch the application are maintained on a central service, in order to reduce the complexity of the AHE client. The design of the AHE has been greatly influenced by WEDS (WSRF-based Environment for Distributed Simulations)[7], a hosting environment designed for operation primarily within a single administrative domain. The AHE differs in that it is designed to operate across multiple administrative domains seamlessly, but it can also be used to provide a uniform interface to applications deployed on both local machines, and remote grid resources.

The AHE is based on a number of pre-existing grid technologies, principally GridSAM [8] and WSRF::Lite [9]. WSRF::Lite is a Perl implementation of the OASIS WSRF specification. GridSAM provides a web services interface, running in an OMII [10] web services container, for submitting and monitoring computational jobs to a variety of Distributed Resource Managers

[1]The AHE can be downloaded from http://www.realitygrid.org/AHE

(DRM), including Globus [2], Condor [11] and Sun Grid Engine [12], and has a plug-in architecture that allows adapters to be written for different types of DRM. Jobs submitted to GridSAM are described using Job Submission Description Language (JSDL) [13].

The problems associated with 'heavyweight' middleware solutions described above have greatly influenced the design of the AHE. The design assumes that the user's machine does not have to have client software installed to talk directly to the middleware on the target grid resource. Instead the AHE client provides a uniform interface to multiple grid middlewares. The client machine is also assumed to be behind a firewall that uses network address translation [14]. The client therefore has to poll the AHE server to find the status of an application instance. In addition, the client machine needs to be able to upload input files to and download output files from a grid resource, but we assume it does not have GridFTP client software installed. An intermediate file staging area is therefore used to stage files between the client and the target grid resource.

The AHE client maintains no knowledge of the location of the application it wants to run on the target grid resource and should not be affected by changes to a remote grid resource, for example if its underlying middleware changes from Globus version 2 to Globus version 4. The client does not have to be installed on a single machine; the user can move between clients on different machines and access the applications that they have launched. The user can even use a combination of different clients, for example a command line client to launch an application and a GUI client to monitor it. The client must therefore maintain no information about a running application's state.

These constraints have led to the design of a lightweight client for the AHE which is simple to deploy and does not require the user to install any extra libraries or software. It should be noted that this design does not remove the need for middleware solutions such as Globus on the target grid resource; indeed we provide an interface to run applications on several different underlying grid middlewares so it is essential that grid resource providers maintain a supported middleware installation on their machines. What the design does is simplify the experience of the end user.

3. Deploying the AHE

To host an application in the AHE, the expert user must first install and configure it on the target grid resource. The expert user then configures the location and settings of the application on the AHE server and creates a JSDL template document for the application and the resource. To complete the installation the expert user runs a script to repopulate the application registry; the AHE can be updated dynamically and does not require restarting when a new application is added.

The AHE is designed to interact with a variety of different clients. We have developed both GUI and command line clients in Java and is designed to be easy to install. The GUI client uses a wizard to guide a user through launching their application instance. The wizard allows users to specify requirements for the application, such as the number of processors to use, the choice of target grid resource to run their application, staging required input files to the grid resource, specification of any extra arguments for the simulation, and job execution. Once an application instance has been prepared and submitted, the AHE client allows the user to monitor the state of the application by polling its associated WS-Resource. After the application's execution has finished,

the user can stage the application's output files back to their local machine using the client. For further discussion of AHE use cases see [15].

4. Summary

The AHE is designed to facilitate grid based computational science by extension of existing approaches to the use of high-end computing resources. Production codes, few in number but used by significant numbers of people, are installed on a set of grid enabled resources and accessed via a lightweight client which enables simple and also arbitrarily complex application workflows to be developed. We are currently working to integrate the AHE with the HARC [16] co-schedluing system, providing users with a single interface to reserve the resources that they want to use and run their applications.

References

[1] P. V. Coveney, editor. *Scientific Grid Computing*, pages 1701–2095. Phil. Trans. R. Soc. A, 2005.

[2] http://www.globus.org.

[3] http://www.unicore.org.

[4] J. Chin and P.V. Coveney. Towards tractable toolkits for the grid: a plea for lightweight, useable middleware. Technical report, UK e-Science Technical Report UKeS-2004-01, 2004. http://nesc.ac.uk/technical_papers/UKeS-2004-01.pdf.

[5] J. Kewley, R. Allen, R. Crouchley, D. Grose, T. van Ark, M. Hayes, and Morris. L. GROWL: A lightweight grid services toolkit and applications. 4th UK e-Science All Hands Meeting, 2005.

[6] S. Graham, A. Karmarkar, J Mischkinsky, I. Robinson, and I. Sedukin. Web Services Resource Framework. Technical report, OASIS Technical Report, 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.

[7] P. V. Coveney, J. Vicary, J. Chin, and M. Harvey. Introducing WEDS: a Web services-based environment for distributed simulation. In P. V. Coveney, editor, *Scientific Grid Computing*, volume 363, pages 1807–1816. Phil. Trans. R. Soc. A, 2005.

[8] http://gridsam.sourceforge.net.

[9] http://www.sve.man.ac.uk/research/AtoZ/ILCT.

[10] http://www.omii.ac.uk.

[11] http://www.cs.wisc.edu/condor.

[12] http://gridengine.sunsource.net.

[13] http://forge.gridforum.org/projects/jsdl-wg/document/draft-ggf-jsdl-spec/en/21.

[14] http://www.faqs.org/rfcs/rfc1631.html.

[15] P. V. Coveney, R. S. Saksena, S. J. Zasada, M. McKeown, and S. Pickles. The application hosting environment: Lightweight middleware for grid-based computational science. *Computer Physics Communications*, 176(6):406–418, 2007.

[16] J. Maclaren, M. Mc Keown, and Pickles, S. Co-allocation, Fault Tolerance and Grid Computing. 5th UK e-Science All Hands Meeting, 2006.