

The Application Hosting Environment: Lightweight Middleware for Grid Based Computational Science ¹

P. V. Coveney, S. K. Sadiq, R. Saksena, and S. J. Zasada

*Centre for Computational Science, Department of Chemistry,
University College London, Christopher Ingold Laboratories,
20 Gordon Street, London, WC1H 0AJ*

M. Mc Keown, and S. Pickles

*Manchester Computing, Kilburn Building,
The University of Manchester, Oxford Road,
Manchester, M13 9PL*

Abstract

Current grid computing technologies have often been seen as being too heavyweight and unwieldy from a client 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. We do this by building a layer of middleware on top of existing technologies such as Globus, and expose the functionality 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. We show how the flexibility of this design has allowed us to create complex workflows when using grid infrastructure to investigate the molecular dynamics of HIV-1 protease.

I 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; the current solutions present a high barrier to entry for the end user.

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.

II The Application Hosting Environment

In response to the issues raised above we have developed the Application Hosting Environment (AHE)², 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

¹This work is funded by the EPSRC “RealityGrid” (GR/R67699) and “Rapid Prototyping of Usable Grid Middleware” (GR/T27488/01) projects, and also by OMII under the Managed Programme RAHWL project.

²The AHE can be downloaded from <http://www.realitygrid.org/AHE>

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 an 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 (Web services-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 via a variety of Distributed Resource Managers (DRMs), 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.

III 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. 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. To install the AHE clients all an end user need do is download and extract the client package, load his or her X.509 certificate into a Java keystore using a provided script and configure their client with the AHE server endpoints supplied by the AHE expert user.

The AHE client attempts to discover which files need to be staged to and from the resource by parsing the application’s configuration file. It features a plug-in architecture which allows new configuration file parsers to be developed for any application that is to be hosted in the AHE. The parser will also rewrite the user’s application configuration file, removing any relative paths, so that the application can be run on the target grid resource. If no plug-in is available for a certain application, then the user can specify input and output files manually.

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.

IV Workflows for the Molecular Dynamics of HIV-1 Protease

As part of our ongoing research on drug resistance in HIV/AIDS we have successfully used the AHE to manage molecular dynamics simulations of the HIV-1 protease, using the NAMD molecular dynamics code actually installed on many TeraGrid sites. The protease encoded by HIV is responsible for the cleavage and subsequent maturation of viral polyprotein precursors into their constituent proteins. The protease is a symmetric dimer (each monomer has 99 amino acids) that encloses a pair of catalytic aspartic acid residues in the active site. The active site is bound by a pair of highly flexible flaps that allow the substrate access to the aspartic acid dyad [15]. Due to its crucial role in the maturation of the virus, the enzyme has been a key target for antiretroviral inhibitors and an example of structure assisted drug design [16]. Unfortunately, the high mutation rate of HIV coupled with its rapid rate of replication has led to the proliferation of drug resistant mutants that severely limit therapy [17]. The number of clinically interesting mutations of HIV-1 protease is larger than available by crystal structure [18]. It is therefore necessary when modelling HIV-1 protease mutants to employ mutational protocols that derive structures from available wildtype crystal structures with closely related sequences, followed by suitable multi-step equilibration protocols that ensure desired mutants are an accurate representation of the actual structure. Standard multi-step protocols, including gentle annealing to physiologically relevant temperatures, removal of force constraints on the protease and establishing a relevant thermodynamic ensemble

[15], are often employed. Here we employ an equilibration protocol composed of a chained sequence of molecular dynamics runs using the AHE to manage each simulation in the chain. We include steps that allow for conformational sampling and relaxation of the incorporated mutations within the framework of their surrounding protease structure.

The 1TSU crystal structure was used as the starting point for the molecular dynamics equilibration protocol. VMD [19] was used for the initial preparation of the system prior to simulation. The structure was solvated using atomistic TIP3P water [20] in a cubic box with a 10 Å buffer around the protease. The N25D and I84V mutations were incorporated, and the system was electrically neutralized. The molecular dynamics package NAMD2 [21] was used for all equilibration simulations. The long range Coulombic interaction was handled using the particle mesh Ewald summation method (PME) [22]. The SHAKE algorithm [23] was employed, allowing for an integration timestep of 2 fs. The equilibration protocol was adapted from Peryman *et al.* [15] with several modifications. These included a protease-constrained solvation step to prevent premature flap collapse [24] and a mutation-relaxation step allowing conformational sampling of incorporated mutations and their 5 Å environment. The total duration of simulation time in the equilibration protocol was 2 ns.

NAMD configuration files corresponding to each step of the equilibration protocol were set up in such a way that the output files of each step would serve as the input files of the next step. The files were generated automatically using a Perl script, and a naming convention was used for the NAMD configuration file at each step of the equilibration protocol to ease scripting of the workflow.

A Perl script was then created to execute the desired equilibration chain on a remote grid resource, using the AHE middleware to manage the state of each of the steps in the chain. The script executed the `ahe-prepare` command followed by the `ahe-start` command sequentially for each step of the equilibration protocol. This had the effect of preparing a WS-Resource to manage the step, staging input files necessary for the step, and executing the application. The script then polled the AHE server at regular intervals using the `ahe-monitor` command until the simulation step had completed. Once complete, the script staged the files back to the local machine and used them to initiate the next step of the equilibration protocol. The script terminated af-

ter sequentially executing all desired steps in the chained protocol. We have found that the change in RMSD of mutated amino acids (1.02 ± 0.15 Å and 0.92 ± 0.10 Å for D25 and V84 residues respectively) was similar to that of the protease backbone (1.11 ± 0.11 Å) as a whole, indicating a good initial mutational protocol. Differences in the RMSD of mutated residues during minimization and force relaxation also showed that our equilibration protocol assisted in the achievement of equilibration.

V 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.

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. Mischkin-sky, I. Robinson, and I. Sedukin. Web Services Resource Framework. Technical report, OASIS Technical Report, 2006. http://docs.oasis-open.org/wsrp/wsrp-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] A. L. Perryman, J. Lin, and J. A. McCammon. HIV-1 protease molecular dynamics of a wild-type and of the V82F/I84V mutant: Possible contributions to drug resistance and a potential new target site for drugs. *Protein Sci.*, 13:1108–1123, 2004.
- [16] A. Wlodawer and J. Vondrasek. Inhibitors of HIV-1 Protease: A Major Success of Structure-Assisted Drug Design. *Annu. Rev. Biophys. Biomol. Struct.*, 27:249–284, 1998.
- [17] V. A. Johnson, F. Brun-Vezinet, B. Clotet, B. Conway, D. R. Kuritzkes, D. Pillay, J. Schapiro, A. Telenti, and D. Richman. Update of the Drug Resistance Mutations in HIV-1: 2005. *Int. AIDS Soc. - USA*, 13:51–57, 2005.
- [18] V. Zoete, O. Michielin, and M. Karplus. Relation between Sequence and Structure of HIV-1 Protease Inhibitor Complexes: A Model System for the Analysis of Protein Flexibility. *J. Mol. Biol.*, 315:21–52, 2002.
- [19] W. Humphrey, A. Dalke, and K. Schulten. VMD - Visual Molecular Dynamics. *J. Mol. Graph.*, 14:33–38, 1996.
- [20] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.*, 79:926–935, 1983.
- [21] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *J. Comp. Phys.*, 151:283–312, 1999.
- [22] U. Essmann, L. Perera, M. L. Berkowitz, and T. Darden. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103:8577–9593, 1995.
- [23] J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen. Numerical integration of the Cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes. *J. Comp. Phys.*, 23:327–341, 1977.
- [24] K. L. Meagher and H. A. Carlson. Solvation Influences Flap Collapse in HIV-1 Protease. *Proteins: Struct. Funct. Bioinf.*, 58:119–125, 2005.