A Web / Grid Portal Implementation of BioSimGrid: A Biomolecular Simulation Database

Bing Wu[1,2,*], Matthew Dovey[2], Muan Hong Ng[4], Kaihsu Tai[1], Stuart Murdock[3], Paul Jeffreys[2], Simon Cox[4], Jonathan Essex[3] and Mark S.P. Sansom[1]
[1]Department of Biochemistry, [2]e-Science Centre, University of Oxford,
[3]Department of Chemistry, [4]e-Science Centre, University of Southampton
**to whom correspondence should be addressed: bing@biop.ox.ac.uk*

Abstract

The overall aim of the BioSimGrid project (www.biosimgrid.org) is to exploit the Grid infrastructure to enable comparative analysis of the results of biomolecular simulations. In particular this paper presents the implementation of current BioSimGrid Web Portal, which provides a single entry access for Web / Grid services and applications over the BioSimGrid network. The portal has a SOA (Service Oriented Architecture) framework built on the layer of OGSA (Open Grid Service Architecture) and OGSA-DAI (Open Grid Service Architecture Database Access and Integration) middleware. The PortalLib has been developed to allow RAD (Rapid Application Development) of portal applications. The portal also integrates PKI (Public Key Infrastructure) and supports two levels of distributed SSO (Single Sign On): Grid certificate-based SSO for high security, and user/pass based SSO for maximal flexibility.

1. Background

Biomolecular simulations enable us to explore the conformational dynamics of complex molecules such as proteins, membranes and nucleic acids (Fig. 1). However, at present there are considerable problems in comparing the results of multiple simulations, and in integrating simulation results with other (experimentally-derived) sources of data. This problem will become more pressing if simulation studies are to match up to post-genomic approaches such as high throughput protein crystallography.

As protein structure determination becomes more automated, and as advances are made in structural bioinformatics [1] and computational biology, it will become increasingly important that biomolecular simulations do not exist as standalone analyses of single systems, but rather that they become embedded in a matrix of computational and experimental studies of proteins. Furthermore, it will be essential to provide data retrieval and analysis tools that are accessible to a wide community of structural and cell biologists, not just to simulation specialists. It is in this context that the BioSimGrid [2, 3] project is being developed.

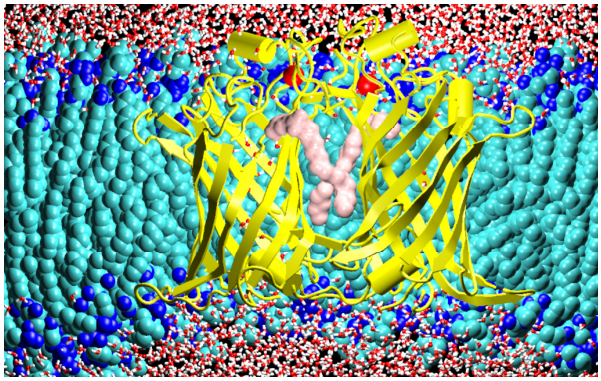

Figure 1. A biomolecular simulation system: the bacterial outer membrane protein OMPLA

Molecular simulations with atom-level resolution have now entered the mainstream of biological research [4]. In particular molecular dynamics (MD) is widely used to investigate nanosecond to microsecond dynamics for a wide range of biomolecules.

MD has benefited considerably from improvements in computer technology. As computers become faster, biologists have become able to explore larger molecules for longer timescales. Currently, a typical simulation may have a system size of ~100,000 particles (atoms), and a nanosecond timescale simulation may require ~1,000,000 timesteps (i.e. iterations of integrating the equations of motion). Such a simulation would take a few weeks on between ~8 and ~64 processors (depending upon the efficiency of

the simulation code and protocols employed) and could generate gigabytes of data for subsequent analysis and visualisation.

The status quo for the archiving of these data is far from optimal. Typically, data is archived in an *ad hoc* fashion at the level of individual laboratories. Furthermore, the reporting of the simulation metadata in journal articles is prone to omission of technical details. Consequently, even medium-scale comparisons between multiple simulations are not possible unless the simulations are performed within a single research group. This excludes simulation results from the domain of structural bioinformatics, where new information is derived by comparisons between the results of individual research endeavors.

2. Grid Enabling Database

The BioSimGrid project will establish a formal database for biomolecular simulations within the UK, increasing collaboration via a distributed computing environment. There are three levels of data existing in the database (see Fig. 2):

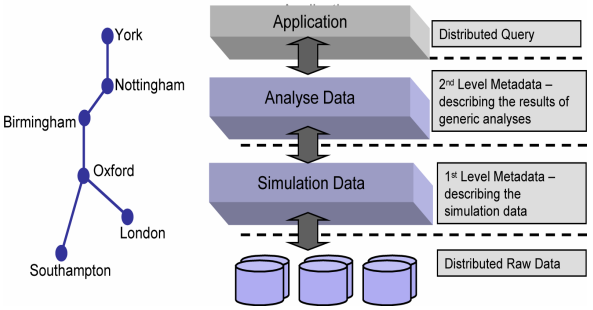

Figure 2. An overview of the BioSimGrid database

• Raw data: generated by biomolecular simulations;

• 1st level metadata: describing the generic properties of raw simulation data, such as data location, simulator configuration, etc;

• 2nd level metadata: describing the results of generic analyses of simulation data. This will be produced by a suite of generic analysis tools and will provide simulation data 'kite-marks'.

The core database has been implemented using IBM DB2 UDB V8.1 ESE [5]. Current datasets contain 3 trajectories (two of which are trajectories of the nervous-system protein AChE; the other is of the bacterial outer-membrane protein OMPLA), 62 thousand atoms, 30 thousand frames at 1 picosecond apart, and about 600 million coordinates. The data size is about 40GB. These large amount of datasets need to be easily accessed by the community. To deliver this, we have designed and implemented a web portal.

A current direction of development focuses on the exploitation of Grid technologies [6, 7, 8] such as OGSA-DAI [9] to enable interrogation of data through distributed queries. An application can send to BioSimGrid a query as if it were to be executed at one instance of a relational database, and OGSA-DAI/BioSimGrid will process it to be distributed on several instances for execution.

3. Portal Architectures

3.1. SOA - Service Oriented Architecture

The current methodology in developing distributed systems is Service Oriented Architecture (SOA), building upon methodologies such as Object Oriented programming, Components and Distributed Object Request Brokers. Within a SOA systems are composed of multiple individual services located and maintained on different heterogeneous machines administered by different organizations. The key in SOA is that the component services should be loosely coupled. That is to say that the service should be well-defined, self-contained, and should not depend on the context or state of other services. This allows the orchestration of systems built up from component services, which should be robust against implementation changes in the underlying services [10]. To achieve this Service Oriented Architectures strive towards a number of goals:

1. The services should implement a small set of simple, ubiquitous and well known interfaces which only encode generic semantics.
2. The interfaces should deliver messages constrained by extensible schema for efficiency. This allows both services and consumers to work with well defined message structures, but allowing new versions of the services to be introduced without breaking existing systems
3. The messages should be descriptive not instructive and the interfaces should not define system behaviour. This allows internals of a service can be viewed as a "black box". You can send a service the details of the problem to be solved and preferences, but need not dictate how the service should solve the problem.
4. Service Oriented Architectures must have mechanisms for the discovery of services matching the consumers requirements

There are a number of emergent technologies which can underpin SOA: REST WebServices; SOAP WebServices; GRID Services.

3.2. REST, SOAP and GRID Services

Representational State Transfer (REST) works on the basic of "resources" which can be references by URIs [11]. A REST web service is limited to using HTTP interfaces (GET to obtain a representation of the resource; DELETE to remove a representation of a resource; POST to update or create a representation of a resource; PUT to create a representation of a resource). REST messages are in XML, constrained by schema definitions in XML Schema [12] or Relax NG[13].

SOAP Web Services use messages encapsulated in a structure defined by the SOAP specification [14]. This adds additional information in the form of headers for message routing scenarios and mechanisms for reporting errors using faults (a style similar to exceptions in various programming languages). SOAP Web Services use Web Service Description Language (WSDL) [15] to define both the structures (again using schema languages such as XML Schema) but also messaging semantics such as whether the message is initiated by the client or the server, and what messages can be used as a response to a particular message.

GRIDServices are based on WebServices but provide additional semantics. In particular they add some object-oriented and REST concepts. The object-oriented concepts are the ability to inherit service definitions (portTypes in WSDL terminology) and add new messages using a multiple inheritance model and the ability to add properties (or service data elements) to WebServices. The REST concept introduced is that of creating a new representation of resource. In the GRIDServices model this uses a factory model whereby a new instance of a GRIDService can be created by its corresponding factory GRIDService. GRIDServices also offer an extensibility model whereby part of the structure of the message can be left undefined, but the allowed structures can be detemined dynamically by querying the appropriate service data elements.

3.3. OGSI and OGSA Infrastructure

As GRIDServices offer an object-oriented inheritance model, the need for well known interfaces with an SOA can be met by defining base GRIDServices which all GRIDServices inherit. These foundation services form part of OGSI (Open Grid Service Infrastructure) and. OGSA, in addition to defining the extensions to WSDL needed for inheritance and service data elements and the semantics for the factory and extensibility models, also define a core set of foundation services and their behaviour. OGSA builds on top of OGSI to define various common service definitions for essential middleware components such as logging, account management, workflow management and data access [16].

The key middleware for BioSimGrid are the services defined within OGSA-DAI. This defines a set of services for accessing heterogeneous databases but with an abstraction layer so that the client can manage a table spread across multiple remote databases as if it were a table on a single local database [9].

3.4. Portal Infrastructure

The key concept behind portals is content syndication. Portal channel producers publish raw content (with minimal presentation information), and it is the role of the portal to aggregate content and handle the presentation and rendering of content into a form suitable for the user. In this sense portals fit well into an SOA philosophy and provide a configurable and versatile user interface allowing the integration and management of loosely coupled services.

At present there are a number of emerging standards for communication between portal channel provides and consumers – JSR-168 defining Java interfaces [17] and WSRP (WebServices for Remote Portlets) defining a WebService interface [18]. There is activity within the Global Grid Forum for a OGSA interface for portals. JST-168 and WSRP only finalized their specifications late 2003 and any OGSA based interface is still in early development and hence BioSimGrid had to develop its own interim portal library.

4. Portal Implementation

4.1. SOA implementation

The current implementation of BioSimGrid Portal is based on multi-tier SOA architecture.

• GUI: HTTP(S)-based web client, provides user interaction with the system. The client can be either a standard web browser or web-based application. The use of the web interface eliminates development and maintenance of client software. And the user can interact with BioSimGrid Portal from anywhere and with "anything" (laptop, PDA, mobile phone, TV, etc.);

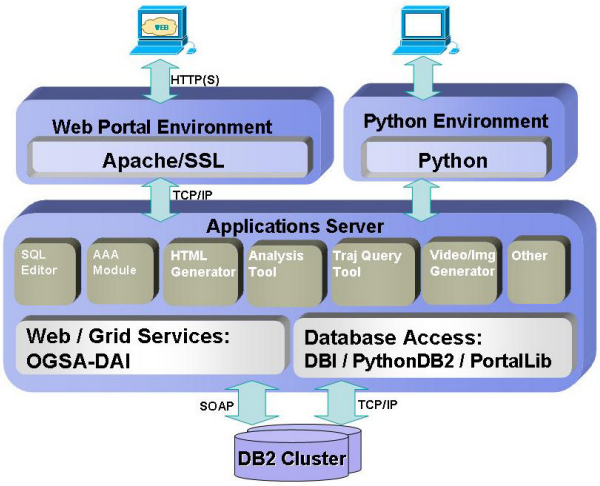

Figure 3. BioSimGrid Portal implementation

• Environment: This tier is a SOA front-end which handles communications between web applications and application servers. We have two environments here. The Web Portal Environment is a standard Apache/Tomcat based hosting environment to deliver web portal communications. The Python Hosting Environment is handling legacy Python communication between clients and application servers.

• Application server: This tier is dedicated to delivering data analysis and data mining services to the biomolecular simulation and structural biology communities through Grid-based Web services. There are also supporting services such as monitoring, transaction, and distributed query services. The protocols used here are XML/SOAP. The application servers also deal with OGSA-DAI Grid middleware to query the database by means of Web services and Grid services. As in the Fig. 3, all the Grid functions are transparent to the applications.

• Database: The IBM UDB DB2 ESE V8.1 has been deployed as the core database. Data resources are distributed across collaborating sites. The OGSA-DAI Grid middleware enables the access of these resources transparently in a format of virtual machine. The database of individual site can be clustered to achieve better performance and reliability.

4.2 Dealing with simulation data

We have selected an initial application that both tests our methodology and also allows us to explore an important biological question. We are currently using BioSimGrid to compare simulations of two enzymes: (i) acetylcholinesterase (AChE), a key enzyme of the nervous system [19]; and (ii) OMPLA, a bacterial enzyme involved in pathogenesis [20]. Structural data show that these two enzymes have similar active sites (a triad of amino acid sidechains involved in their catalytic mechanisms). Otherwise, the structures of the two proteins are completely unrelated.

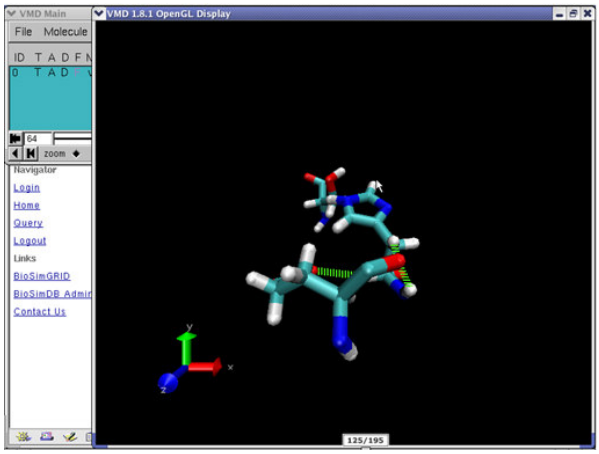

Figure 4. A screenshot of the BioSimGrid portal showing the active site for AChE

We are analysing simulations to compare the patterns of catalytic sidechain dynamics in these two distantly related enzymes, so as to understand the relationship between sidechain mobility and catalytic mechanism. The AChE simulations were performed at UCSD; the OMPLA simulations in Oxford. Normally, such data would never "meet" and would reside in the separate laboratories. Furthermore, the simulations were run using different programs and protocols and the data are in very different formats. Rather than re-run one (or both) of the simulations (which would consume many days of costly supercomputer time), we are using BioSimGrid to make the comparison. Thus this apparently simple test case enables us to test many aspects of the underlying methodology.

4.3. PortalLib - a portal library

A web portal implementation involves a large amount of web pages, which is based on HTML code. Among these, certain amount of web pages have to be generated dynamically based on the user requests. There are basically two ways of implementations: client-side implementation and server-side implementation. JavaScript and VBScript are widely used as client-side implementation, in which scripts only run on the web browser thus avoids the network traffic and improves user interaction. However, since the client-scripts cannot deal with any operation involving server communications, i.e. display of query results. So server-side scripts have to be used. ASP

(Active Server Page), JSP (Java Server Page) and PHP are very popular server-side scripts.

In the BioSimGrid Portal implementation, we have both client-side and server-side scripts. JavaScript is currently used as the client-side scripts as it can run on most web browsers. We use Perl and Python as server side scripts. We developed PortalLib to provide a common interface to generate dynamic web pages on the fly. PortalLib is a Perl library which consists of three classes to use for rapid portal development:

- CHtmlFile - interfaces with static HTML template files to generate dynamic web pages,
- Auth_lib – provides common authentication and Authorisation interfaces,
- DB_lib – provides a database object with common interface to DBI compatible data sources.

The code in Tab. 1 is an example of using PortalLib to produce a trajectory refinement form based on previous user inputs and database query results. The HTML code is stored in a HTML file called '*refine.html*'. When running '*refine.cgi*', the template HTML code is parsed by PortalLib and all the variables are replaced by the passing values of *%options*.

Table 1. Example code: refine.cgi

```
#!/usr/bin/perl -w

use PortalLib;
use strict;
use CGI;

my $cgi = new CGI;
my %options = ( $cgi->Vars,);

# process template HTML
my $template = 'refine.html';

my $html = CHtmlFile->new(
   File => $template,
   %options,
   );

$html->Draw();
```

JavaScript can be dynamically passed to CHtmlFile object in the *JavaScript* property or embedded in the template HTML file as usual. The generated form is shown in Figure X. The trajectory list is generated dynamically from the database query result and formatted on the HTML form below.

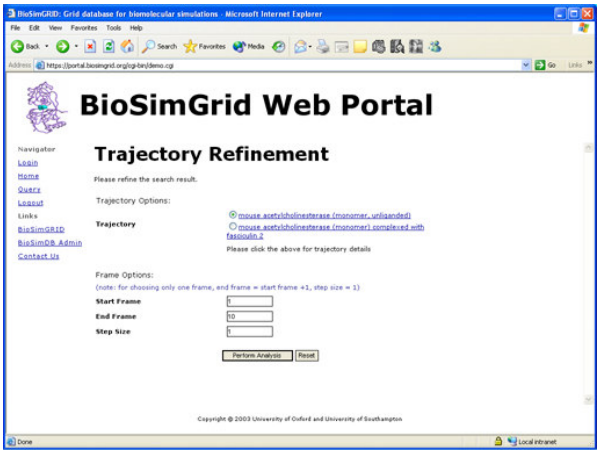

Figure 5. Trajectory refinement form

5. BioSimGrid Security

Given the current distributed Grid implementation, security is a critical element of the project. We integrated various mechanisms to achieve a secure distributed environment. We have implemented PKI (Public Key Infrastructure) and X.509 Digital Certificate [21] based Authentication. All the transactions are based on HTTPS secure channels. The BioSimGrid network has dual firewall protection. Two levels of authentication enable legacy user/pass based authentication. The authorisation is based on a distributed database and implements SSO (Single Sign On).

5.1. PKI

PKI is a system of public key encryption using digital certificates from a CA (Certificate Authority) that verifies and authenticates the validity of each party involved in an electronic transaction. PKI uses an asymmetric key based algorithm. Two different keys are used in the encryption/decryption: a private key is used to encrypt data and a public key can decrypt data encrypted with the private key. A certificate contains information referring to a public key which has been digitally signed by a CA. The information normally found in a certificate conforms to the ITU (IETF) standard X.509 v3 [22], which includes: user's subject name (user ID), user's public key, and a "signature" from a CA proving that the certificate came from the CA. The signature also vouches for the subject name and for the binding of the public key to the subject.

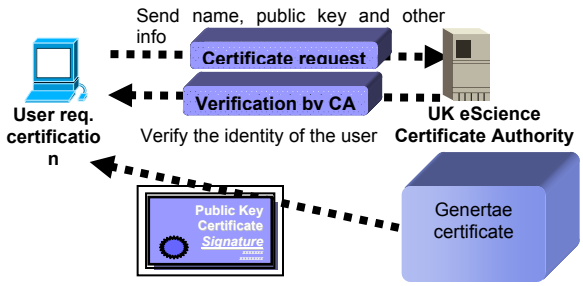

Figure 6. Issuing certificate

The UK e-Science Grid CA [23] provides X.509 certificates for the UK e-Science community. As in Fig X, the certificate issuing process involves the following steps:

1. Creates a public key certificate containing the required info,
2. Runs a computation on the info in the public key certificate to produce a small data string (or hash),
3. Encrypts the small string using its (the CA's) private key (the CA's signature),
4. Sends the public key certificate containing the user's public key and the CA's signature to the user.

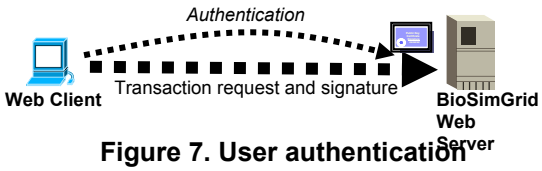

Figure 7. User authentication

Fig. 7 shows the current BioSimGrid Portal certificated based authentication process:

1. User connects to the portal.biosimgrid.org web application,
2. The user's browser creates a digital signature for the request using the local private key,
3. The browser then sends the request and the digital signature to the web server,
4. The web server receive the request and the user's digital signature,
5. The web application then validates the signature against the request using the user's public key.

Once the user digitally signs-on successfully, the BioSimGrid Portal will perform secure web transactions in the order below:

Figure 8. Web transactions

1. The user fills a web form and submit,
2. The user's browser downloads the server's public key from portal.biosimgrid.org,
3. The browser uses this key to encrypt the message containing the transaction details,
4. The browser then sends the encrypted message to the portal.biosimgrid.org web server,
5. The web server receives the message and uses its private key to decrypt the message.

5.2. Grid certificates and Apache SSL

A Grid certificate is an X.509 certificate. We have two kind of Grid certificates used in the portal. Among them, user certificates are used for the user identity and host certificates are used for the servers. Once you have a valid user certificate issued by one of our recognised CAs, you can use this to access the web portal. The details of how to use a user certificate can be found at [24].

A host certificate has similar format as the user certificate except its DN has a hostname instead of the user name in '*CN*'.

```
subject=/C=UK/O=eScience/OU=Oxford/
L=OeSC/CN=portal/portal.biosimgrid.
org/emailAddress=bing@biop.ox.ac.uk
```

The host certificate can be configured to be used by the web server supporting SSL transactions. We use the Apache server in the portal environment. The above host certificate is converted to an Apache-friendly key pair and loaded into the web server. The same key pair can also be used as a host certificate in the Globus Toolkit. Full details can be found at [25].

5.3. SSO (Single Sign-On) and AAA

SSO is designed to provide a foundation that gives users role-based access to multiple Web applications from a single, secure point of contact. Instead of needing to separately log in to multiple applications, SSO enables user sign-on once and gain access to all the authorised resources across the network. This simplifies the user AA (Authentication Authorisation): the user only needs to remember one user id/password. There are various implementations of SSO

technologies. Among them, two of most popular ones are: 3rd party based SSO and centralised SSO. Microsoft Passport technology [26], which is used by Hotmail [27], is a 3rd party based SSO. All the user account details are stored in Passport web site. When a Hotmail user logs in, the account details are passed to Passport.net for authentication and the corresponding AA then returns to Hotmail once completed. While in a centralised SSO, AA is via a centralised local system rather than a 3rd party. For example, Oracle9iAS SSO [28] is a centralised SSO and all AA details are stored in a central Oracle database [29].

In the BioSimGrid project, we have a distributed Grid environment and need to deal with two levels of authentications for high security and easy accessibility. To achieve this, we use SSO based on a distributed database. All the user accounts and corresponding AAA (Authentication Authorisation Accounting) information are stored in the database distributed across the network. This enables a user sign on at any BioSimGrid site and access authorised resources and perform authorised transactions. However, the accounting information of the user access will be stored locally and distributed over the network.

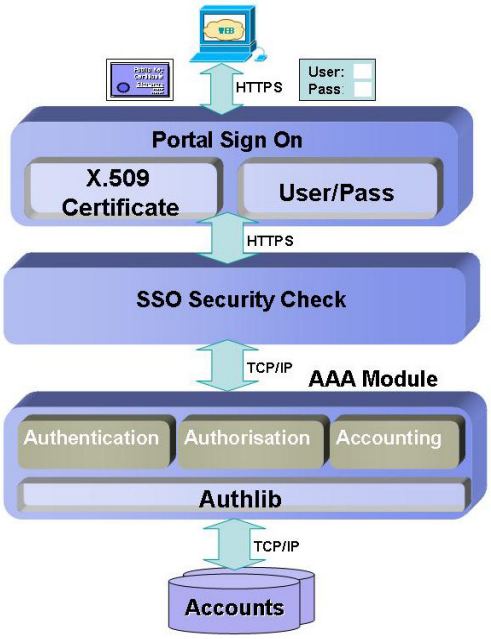

Figure 9. BioSimGrid SSO implementation

As in Fig. 9, two levels of authentication infrastructures have been implemented in the system. The first level is based on digital certificate. A Grid certificate-based authentication mechanism (OpenCA) has been integrated across the system. This requires a valid X.509 digital certificate which is signed by UK e-Science Center. When a user signs on to access specific BioSimGrid services, the subject of his/her X.509 personal digital certificate is passed to the SSO Security Check module, which then calls the AAA module to verify the certificate against the one stored in the accounts database. Only if the security check is successful will the user can have the access to the services.

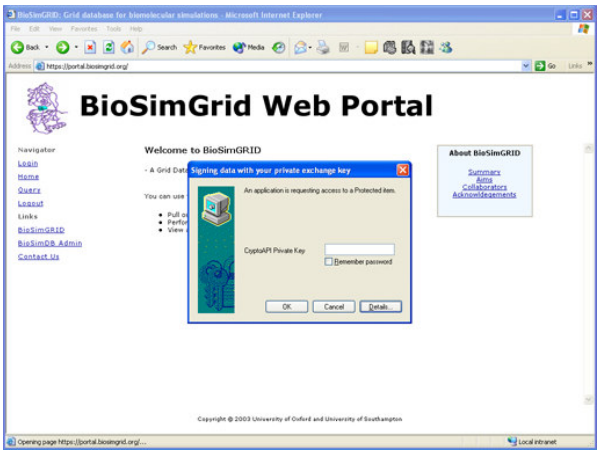

Figure 10. Certificate based SSO

The second level is the traditional user/password based authentication, which is designed for those who have no digital certificates installed in the client machines. This level of authentication enables web access of the system via a public PC anywhere in the world.

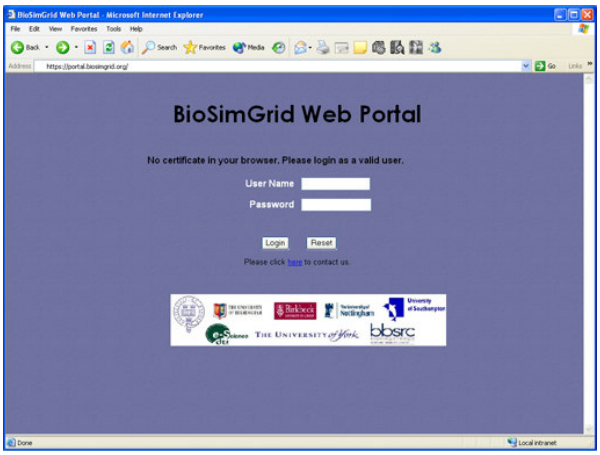

Figure 11. User/pass based SSO

Once the user has been granted access, the AAA module will store the user credentials in the database and generate a unique access token for this session. The token has a limit life time to enhance the security. We are also investigating the integration of user

credential delegation using MyProxy [30]. All the transactions of the system and user activities are logged in the accounts database, which will then be distributed across the BioSimGrid sites for maximal efficiency and high availability.

6. Conclusions

The BioSimGrid project is still in its infancy. However, the AChE vs. OMPLA comparison provides a microcosm of the many comparisons that will become possible once BioSimGrid is fully operational. Biomolecular simulation groups will be able to deposit their simulation data for wide-ranging comparative analyses that so far have been impossible. This will help to propel biomolecular simulation studies into the post-genomic era.

The OGSA and OGSA-DAI architectures underlying BioSimGrid are still in early stage of development. Therefore we have both legacy programs and new web services co-existing in the current portal environment. Once web / Grid services are stable, it would be worthwhile to migrate all legacy programs to web services under the proposed SOA architecture.

We are also investigating the possibility of integrating existing Condor clusters (available in Oxford and Southampton) as computing nodes of the project. This will involve allowing access of BioSimGrid databases by Condor clusters and delivering results back to the system.

We need to develop methods for data deposition, data exchange and for quality control of simulation data. This will enable us to run initial comparative analyses on complex simulations (e.g. of biological membranes) in order to evaluate the current prototype in real world applications.

Acknowledgements

Many thanks to our BioSimGrid partners (L. Caves, C. Laughton, D. Moss and O. Smart) for their numerous inputs to this project. BioSimGrid is funded by BBSRC and DTI. Our thanks to all of our colleagues in the Oxford and Southampton simulation labs, to Ivaylo Kostadinov in the Oxford e-Science Center, and to Steven Johnston and Hans Fangohr in the Southampton Regional e-Science Centre for their encouragement and advice. Our special thanks to 1st GGF International Summer School [31] on Grid Computing for comprehensive trainings on Grid technologies.

References

[1] Bourne, P.E. and Weissig, H. (2003) Structural Bioinformatics, Wiley-Liss, Hoboken.
[2] Bing Wu, Kaihsu Tai, et al., BioSimGrid: A Distributed Database for Biomolecular Simulations. Simon J Cox (editor), Proceedings of UK e-Science All Hands Meeting 2003. EPSRC Sept 2003, ISBN 1-904425-11-9.
[3] http://www.biosimgrid.org
[4] Karplus, M.J. and McCammon, J.A. (2002) Nature Struct. Biol., 9, 646-652.
[5] http://www-3.ibm.com/software/data/db2/udb/
[6] http://www.globus.org
[7] Berman, F., Fox, G. and Hey, T., Eds., Grid Computing: Making the Global Infrastructure a Reality (Wiley, 2003).
[8] Foster, I. and Kesselman, C., Eds., The GRID: Blueprint for a New Computing (Morgan-Kaufmann, 1999).
[9] http://www.ogsa-dai.org
[10] Hao He, What is Service-Oriented Architecture, http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html
[11] http://www1.ics.uci.edu/%7Efielding/pubs/dissertation/-rest_arch_style.htm
[12] http://www.w3.org/XML/Schema
[13] http://www.oasis-open.org/committees/tc_home.php?w-_abbrev=relax-ng
[14] http://www.w3.org/2000/xp/Group/
[15] http://www.w3.org/2002/ws/desc/
[16] Foster, I., Kesselman, C., Nick, J. and Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Global Grid Forum (2002).
[17] http://www.jcp.org/aboutJava/communityprocess/reiew-/jsr168/
[18] http://www.oasis-open.org/committees/tc_home.php?w-_abbrev=wsrp
[19] Kaihsu Tai, Tongye Shen, Richard H. Henchman, Yves Bourne, Pascale Marchot, J. Andrew McCammon (2002) Mechanism of acetylcholinesterase inhibition by fasciculin: a 5-ns molecular dynamics simulation. J. Am. Chem. Soc. 124:6153-6161.
[20] Baaden M, Meier C, Sansom MSP (2003) A molecular dynamics investigation of mono- and dimeric states of the outer membrane enzyme OMPLA. J Mol Biol 331:177-189.
[21] Tuecke, S., Engert, D., Foster, I., Thompson, M., Pearlman, L. and Kesselman, C., Internet X.509 Public Key Infrastructure ProxyCertificate Profile, IETF (2001).
[22] ftp://ftp.isi.edu/in-notes/rfc2459.txt
[23] http://www.grid-support.ac.uk/ca/
[24] Bing Wu, User Certificate Installation Guide, http://ww-w.biosimgrid.org/docs/2003/NOTES/user_certificates.pdf
[25] Bing Wu, Digital Certificate Installation Notes, http://w-ww.biosimgrid.org/docs/2003/NOTES/certificates.pdf
[26] http://www.passport.net/
[27] http://www.hotmail.com/
[28] http://www.oracle.com/appserver/index.html
[29] http://www.oracle.com/ip/deploy/database/oracle9i/
[30] http://www.ncsa.uiuc.edu/Divisions/ACES/MyProxy/
[31] http://www.dma.unina.it/~murli/SummerSchool/