Available online at www.sciencedirect.com

SCIENCE DIRECT®

Future Generation Computer Systems 22 (2006) 657–664

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/fgcs

BioSimGrid: Grid-enabled biomolecular simulation data storage and analysis

Muan Hong Ng[a,*], Steven Johnston[a], Bing Wu[c], Stuart E. Murdock[b], Kaihsu Tai[d], Hans Fangohr[a], Simon J. Cox[a], Jonathan W. Essex[b], Mark S.P. Sansom[d], Paul Jeffreys[c]

[a] *Southampton e-Science Centre, SO17 1BJ Southampton, UK*
[b] *School of Chemistry, University of Southampton, SO17 1BJ Southampton, UK*
[c] *Oxford e-Science Centre, OX2 6NN Oxford, UK*
[d] *Department of Biochemistry, University of Oxford, OX1 3QU Oxford, UK*

Received 15 August 2005; received in revised form 20 October 2005; accepted 21 October 2005
Available online 2 February 2006

Abstract

In computational biomolecular research, large amounts of simulation data are generated to capture the motion of proteins. These massive simulation data can be analysed in a number of ways to reveal the biochemical properties of the proteins. However, the legacy way of storing these data (usually in the laboratory where the simulations have been run) often hinders a wider sharing and easier cross-comparison of simulation results. The data is commonly encoded in a way specific to the simulation package that produced the data and can only be analysed with tools developed specifically for that simulation package. The BioSimGrid platform seeks to provide a solution to these challenges by exploiting the potential of the Grid in facilitating data sharing. By using BioSimGrid either in a scripting or web environment, users can deposit their data and reuse it for analysis. BioSimGrid tools manage the multiple storage locations transparently to the users and provide a set of retrieval and analysis tools for processing the data in a convenient and efficient manner. This paper details the usage and implementation of BioSimGrid using a combination of commercial databases, the Storage Resource Broker and Python scripts, gluing the building blocks together. It introduces a case study of how BioSimGrid can be used for better storage, retrieval and analysis of biomolecular simulation data.

Keywords: Biomolecular simulation; Database; Grid computing; Storage resource broker; Python

1. Introduction

In the field of biomolecular simulation, massive amounts of data, often tens of gigabytes per simulation, are generated to capture the motion of molecules at different time steps. These simulation results are suitable for reuse in many different analysis studies. One application of such simulation and post-simulation analysis is predictive modeling in drug discovery, where motions of proteins [1] are important.

For many years in the biochemical research community, protein simulation data have been stored locally where they were generated; this severely limits data sharing within the biochemical community. Even if these data are transferable across labs, they cannot easily be compared with pre-existing analysis scripts due to the variety of data formats. These analysis scripts are normally written to process a single data format, thus causing great inconvenience in the comparison between simulation data of dissimilar formats. The constraints of sharing and comparing different simulation data can be a major hindrance to the discovery of new science within the biochemistry community.

The solution to this problem is within the concept of grid computing [2] as the Grid promotes the sharing of computational and storage resources within the scientific research world. In the concept of the data Grid [3], scientists are allowed access to geographically remote storage resources across a network in a uniform and efficient manner. With this data sharing capacity, biochemists are able to perform cross-simulation comparison to explore fully the functional dynamics of biomolecular simulations.

[*] Corresponding address: Southampton e-Science Centre, School of Engineering Sciences, Building 25 Highfield University of Southampton, SO17 1BJ Southampton, Hants, UK. Tel.: +44 2380598520.
E-mail address: muanhong@soton.ac.uk (M.H. Ng).

BioSimGrid [4] is a project that seeks to exploit the potential of the Grid on biomolecular simulation data and tries to solve some of the problems that hamper comparative analysis. It provides a platform for the biochemists for conveniently storing, retrieving and analysing biomolecular simulation data.

The next section provides a general overview of BioSimGrid, Section 3 gives details on the architectural implementation and Section 4 touches on the security and data authorisation issues. We explain the functionalities of BioSimGrid from the user perspective in Section 5 and provide an application example in Section 6. Finally Section 7 discusses various issues involved in the development stage and Section 8 concludes this paper with future work and potential extensibility.

2. BioSimGrid

BioSimGrid currently has multiple data resources spread across six different university research labs in the UK. It deals with simulation data that exists in the form of trajectories (sets of coordinates corresponding to the positions of atoms for a series of time steps) which can go up to the size of 10 GB per trajectory. Each trajectory has its corresponding metadata that describes the topology of its atoms, the parameter set for the simulation and also user defined metadata. All these metadata are essential and form parts of the querying clause in data retrieval.

BioSimGrid is still at its prototype stage with over 20 users. It currently has over 200 trajectories contributing to 450 MB of metadata and approximately 1 TB of flat files. The current system has storage space for 24 TB of data distributed over six sites. The available storage space can be increased by adding further machines.

Users of BioSimGrid are able to deposit their simulation data, which exists in multiple data formats, into this repository. These data can then be made available to the whole community. The BioSimGrid data retrieval component enables a user to retrieve data transparently without knowledge of the database mechanisms behind the scenes. The flexibility of the retrieval tools allows users to access different slices of a trajectory seamlessly. BioSimGrid also provides a set of custom-built analysis tools which can be used to study the functional dynamics of a simulation, e.g. root mean square deviations, volume and average structure, interatomic distance and surface area. Alternatively, users can write their own analysis tools by utilising the retrieval components to access different slices of a trajectory as their analysis requires.

The deposition, retrieval and analysis components are implemented in Python [5]. Users can use BioSimGrid in a Python scripting environment. Alternatively, there is a web based interface to BioSimGrid with limited analysis capabilities and without the functionality of data deposition. The rationale behind choosing Python is pragmatic since several analysis dependent post-processing tools (such as PyMOL [6] and MMTK [7]) were written in Python and the simulation community is moving towards Python as the preferred environment for post-simulation analysis.

In summary, BioSimGrid seeks to fulfil the following requirements in its implementation:

- to provide a transparency of data location to the users, where the knowledge of the actual physical location of the data is not essential to the process of data retrieval,
- to maximise data transfer rate, in terms of the speed of delivering data to the computation element, in this case the analysis toolkit,
- to provide an abstraction of the data layer, where scientists are freed from the complication of using and understanding data querying languages and the data storage structure in their scientific research,
- to provide a general purpose analysis toolkit for operating on this data structure.

3. Architectural implementation

At each site BioSimGrid is running a dual processor AMD 2600 with 4 GB of RAM and 4 TB of RAID 5 storage. As depicted in Fig. 1, BioSimGrid is implemented on a three tier architecture. The first layer is the data layer which consists of relational databases and flat file storage. The trajectory coordinates, which are the larger part of the data, are stored as flat files whilst the metadata is stored in the relational databases. The middle-tier layer is a combination of BioSimGrid purpose-built Python modules and a grid middleware called Storage Resource Broker (SRB) [8,9]. The former manages geographically distributed data deposition and retrieval while the later maintains the distributed flat files. The user application layer comprises a selection of analysis tools, Python scripts for deposition and a web portal that caters for administration purposes as well as graphical user front-ends. The set-up is such that the master site hosts a master Oracle 10g [10] and an MCAT (Metadata Catalogue) enabled SRB server 3.3. All other sites are configured as slaves with a SRB server and a replicated Oracle database installed.

3.1. Data deposition and retrieval

BioSimGrid aims to provide a data abstraction layer for the user where scientists can concentrate on research without the concerns of ‘how’ and ‘where’ their data is stored. One of the challenges is dealing with the simulation data which comes in varying formats, e.g. AMBER [11], Gromacs [12], NAMD [13] and Charmm [14]. These data need preprocessing before they can be stored as one generic format in BioSimGrid. A solution is offered in the BioSimGrid deposition modules as depicted in Fig. 2. The modular approach means a parser component is built for each data format which is extensible to support new data formats. A trajectory is first parsed into a generic input object before it is validated for their data types and finally imported into the storage as one generic format. With this, the underlying complexity of format conversion, data validation and data import are completely hidden from the users.

Once the trajectory is stored, users can access different parts of the trajectory in a uniform manner regardless of the original format of the simulation data. The BioSimGrid retrieval module provides a conceptual way of retrieving a full and partial trajectory through ‘`frame collection`’ and ‘`frame`’ objects.

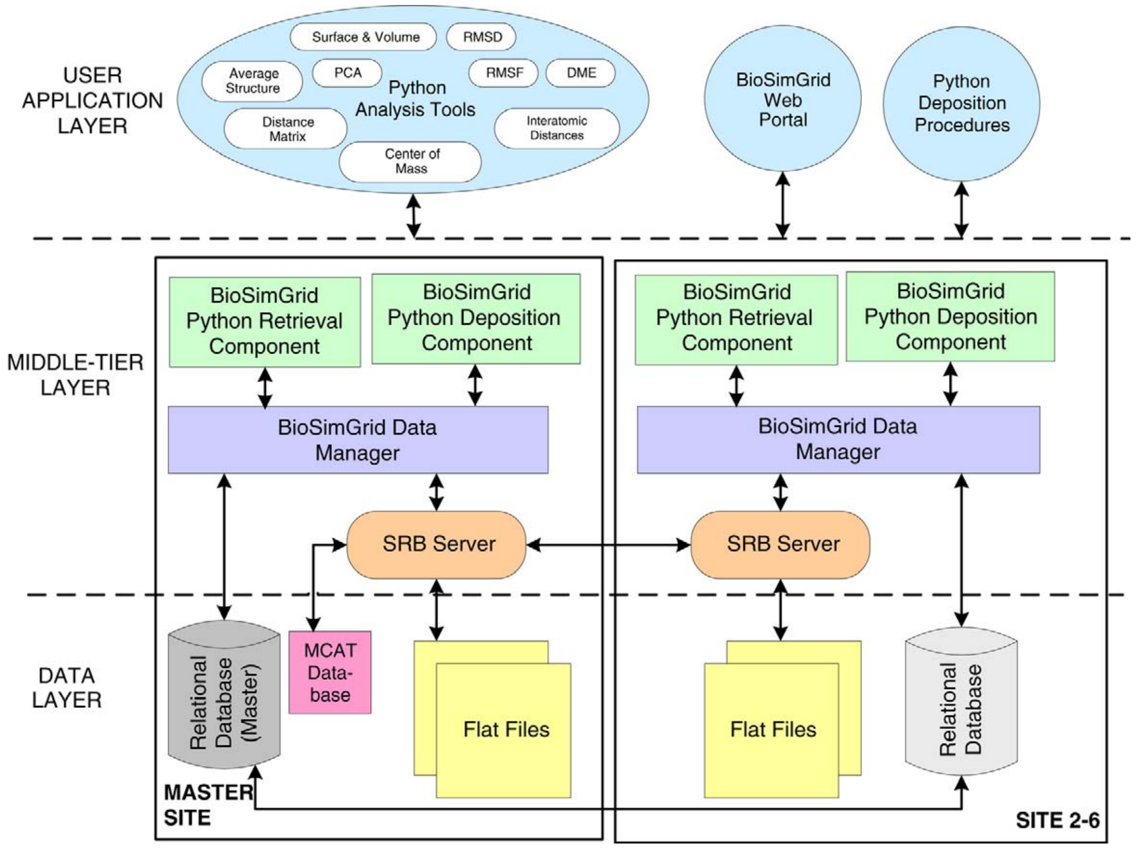


Fig. 1. The three tier architectural diagram of BioSimGrid depicting the data layer, middle-tier layer and application layer.

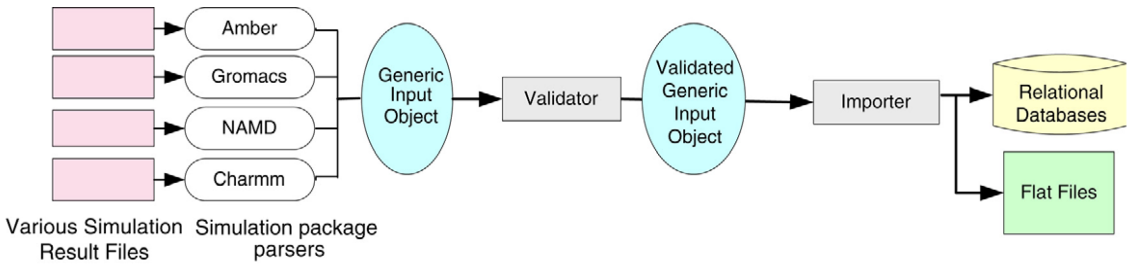


Fig. 2. The modular approach of the BioSimGrid deposition modules comprises simulation data parsers, a data validator and a data importer. New parsers can easily be added to support new data formats. (Source: Phil. Trans. R. Soc. A.)

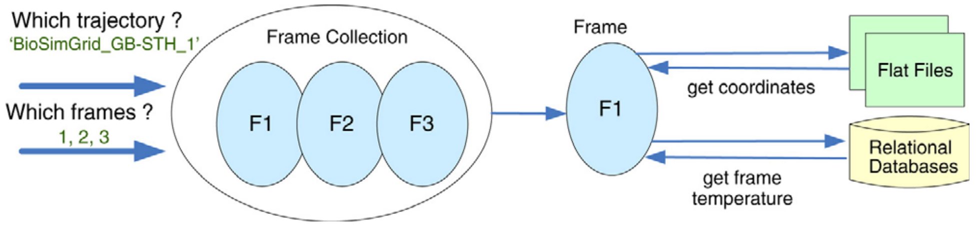


Fig. 3. A retrieval scenario: a frame collection is used to specify which frame from which trajectory is to be accessed; the frame object is used to access its coordinates or metadata.

As shown in Fig. 3 a frame collection is an interface for accessing trajectories. Once the user has specified which frames of the trajectories are to be accessed, the frame collection acts as a temporary buffer space for the collection of frames to be used during an analysis. The frame object, in turn, gives users a series of access options for getting different pieces of data, e.g. coordinates, temperature, atom names and atom masses. The underlying retrieval stages include:

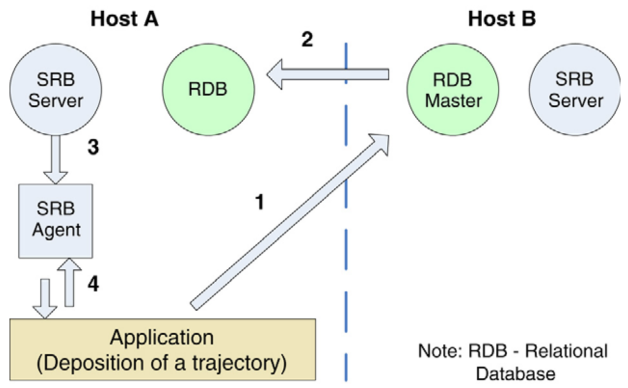

Host A
Host B
SRB Server
RDB
RDB Master
SRB Server
SRB Agent
Application (Deposition of a trajectory)
1
2
3
4
Note: RDB - Relational Database

Fig. 4. This illustrates the federated BioSimGrid in operation during a data deposition routine.

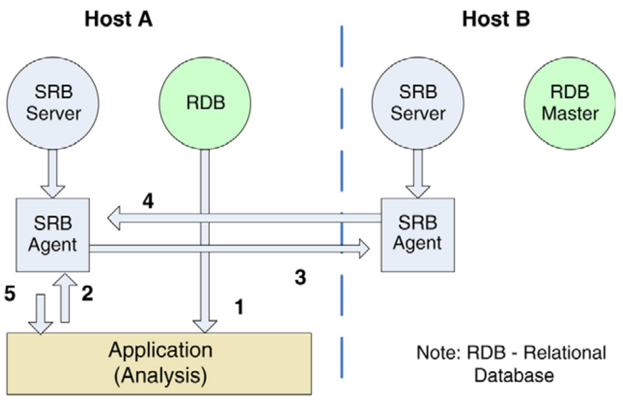


Fig. 5. This illustrates the federated BioSimGrid in operation during an analysis where data is retrieved.

- the initialisation of the BioSimGrid retrieval module with the relational database to retrieve metadata or coordinate indexes for flat file access,
- the coordinate indexes are used by the BioSimGrid retrieval module to interact with the SRB flat file storage via the SRB python interface to retrieve coordinates.

A frame collection is an abstract layer which stores a series of frame objects, each frame object contains data about a specific frame. Apart from the frame metadata which is stored in the relational database, the frame data which is the coordinates are stored in SRB flat files. To reduce the volume of data that is transmitted across the network a frame object is only populated with a partial or full frame of the data as it is required. This is possible using the Python SRB interface which enables the underlying code to open and seek specific locations in an SRB file. Thus there is no need to retrieve the whole SRB file. The frame object is also responsible for caching pre-fetched data to speed up repeat requests. The end result is an abstraction layer which efficiently retrieves data from different locations to produce objects which the user can manipulate without knowledge of the underlying distributed SRB infrastructure.

3.2. *Data federation and replication*

As illustrated in the previous section, users have no knowledge of the actual physical location of the data. The data layer is completely transparent to the user and it is managed by the SRB and BioSimGrid middle-tier layer. Each participating site has a copy of metadata stored in a relational database (RDB) and an SRB storage vault designated for the flat files. All Oracle databases are set-up in a single-master-replication (SMR) mode. The master Oracle database has both read and write permissions whilst the slaves' Oracle databases are configured in read-only mode. Each site has an SRB server that handles read and write request to/from a local or remote flat file resource vault. As illustrated in Fig. 4 the following events take place when a trajectory is deposited at host A:

(1) The metadata part of the trajectory is first written into a master RDB (using Oracle Database) at host B.
(2) This metadata is then replicated to all other hosts using the built-in replication technology of Oracle 10g Database Server.
(3) The SRB master forks out an SRB agent to handle the communication channel established between the SRB and the user application.
(4) Through this channel, the user application writes the coordinates into the flat files on host A.

The replication of the flat files is performed by the daily cronjob at each host. The replication script will pick a random target host and use the SRB in-built replication mechanism to replicate a copy of the flat files on host A to one other host. (It is also possible to pick the replica site which has the most disk space available.)

To perform analysis computation, the deposited data needs to be retrieved. For a data retrieval operation in a federated BioSimGrid scenario, both coordinates and the metadata part of the simulation data are fetched from the storage. Fig. 5 shows an example of an analysis running on host A:

(1) The metadata is retrieved directly from the RDB on host A.
(2) In retrieving the coordinates, the SRB agent on host A issues a request for the flat files.
(3) In this case the flat files are located remotely on host B, the SRB agent on host A forwards the request to the SRB Master on host B.
(4) A file handler is returned to the agent on host A.
(5) The application on host A now has access to the flat files on host B through this file handler.

The federation plan of BioSimGrid includes storing a replica for each simulation data set at a different site for redundancy. The metadata is replicated across all participating sites and the coordinates are duplicated to one other site, instead of all sites, to save storage requirements. The replication is an important measure for a data intensive application such as that of BioSimGrid since re-running a simulation can be both computational intensive and time-consuming. The replica is also used in load balancing as the closest replica is always selected during a data query.

```
1    from BioSim.Deposit.NAMDDeposit import NAMDDeposit
2    from BioSim.Settings import UserSettings
3
4    #set up input files
5    sim_files = {'coordinates':['file1.crd','file2.crd'],
6                 'topology':'file.pdb',
7                 'parameter':'file.par'}
8    #create an instance of user settings
9    uSettings =  UserSettings.UserSettings('Bob')
10   #deposit
11   NAMDDeposit.NAMDDeposit(uSettings, sim_files)
```

Fig. 6. An example of a Python deposition routine implemented by user 'Bob' to deposit a NAMD trajectory into BioSimGrid. The underlying complexity of parsing, validating and importing the simulation data is hidden from the user.

4. Security and data authorisation

BioSimGrid has adopted the Linux style username–password security measures for authenticating a potential user into the system within the scripting environment. A Linux user account is created for a new user and they are also registered into the BioSimGrid database before they can start depositing trajectories. Hence unauthorised users are prevented from writing into the database. Each trajectory has an owner and only the owner has the permission to publish their trajectory to the public. Each user is allowed to write to the storage. After the first write is completed users can flag their own trajectory for deletion. Only the system administrator has the permission to delete a trajectory. This is a preventive measure to avoid users accidentally deleting their valuable sets of data. All users are granted read permission to trajectories published by their owners. The web interface accepts both digital certificate and username–password authentication. Various permission levels can be set for different user groups for authorising them into different transactions. Only users with administration privilege are allowed to add new users into the system via the web interface.

5. User perspective

BioSimGrid offers two user front-ends: a Python scripting environment and a web interface [15]. The scripting environment is suitable for those who would like to use BioSimGrid in a programatic manner, for example, the advanced users who wish to deposit data and have the intention to write their own analysis tools. Fig. 6 shows an example of a Python deposition routine written by the user to deposit an AMBER trajectory into BioSimGrid. The first and second lines import the required packages. The fifth to seventh lines specify the input files, the ninth line defines the user (the owner of the trajectory) in the user settings, and the last line creates a NAMD deposit object to initiate the trajectory deposition process.

A global id is used to uniquely identify a trajectory in BioSimGrid. It has the form of `'BioSimGrid_GB-X_Y'` where `X` is the ISO 3166-2 code [16] for the administrative subdivision in which the site is located (e.g. `'GB-OXF'` for Oxford) and `Y` is a positive integer number. A trajectory global id is returned to the user (printed on the logged output file) upon the successful completion of a deposition. It is then used to query the trajectory for analysis.

Fig. 7 shows how the global id is used to refer to a particular trajectory in a data query. Lines 1 and 2 import the required packages, line 5 instantiates a user setting, line 8 specifies frames 1, 2 and 3 to be fetched from trajectory `'BioSimGrid_GB-STH_1'` and line 11 instantiates a frame collection object to act as a cache for the required frames. At this point, the user can utilise the frame object to access the data or metadata required in their analysis as shown in line 14. Line 16 returns the frame's coordinates in Python Numeric array [17] format and line 18 returns the frame's temperature.

Alternatively, a user can use BioSimGrid analysis tools instead of writing their own. Fig. 8 shows a script for analysing the average structure of a molecule. The first five lines import the relevant packages and create the user settings. Lines 7, 8 and 9 specify the trajectory, frames and residues to be used in the analysis. Line 10 creates an instance of the average structure analysis tool, and lines 11 and 12 specify the calculation result to be generated as a text file called 'av.txt'.

The web front-end caters for two groups of users: (a) administrators, who wish to perform administration tasks on the system (e.g. adding a new user) and (b) users, who prefer a graphical user interface as opposed to the scripting environment. The web front-end offers restricted BioSimGrid functionalities and it is not possible to deposit trajectories through the web interface. However, it is a convenient tool for performing certain standard analyses with a few mouse clicks.

6. BioSimGrid application example

This section gives an application example which demonstrates BioSimGrid as a technology for analysing multiple trajectories in biomolecular research. Here is a comparative analysis of Molecular Dynamics (MD) simulations for four biomolecules (Fig. 9): acetylcholinesterase (AChE, a key enzyme of the nervous system), outer-membrane phospholipase A (OMPLA, a bacterial enzyme involved in pathogenesis), outer-membrane protease T (OmpT, belonging to the category of peptide hydrolases) and PagP (an enzyme that promotes transfer of a chemical group from one molecule to another). The patterns of catalytic side chain dynamics in these four superficially unrelated enzymes are studied to investigate the relationship between side chain motions. A set of metrics, namely distance

```
from BioSim.DataRetrieval import FrameCollection, FCSettings, Frame
from BioSim.Settings import UserSettings

#create an instance of user settings
USettings =  UserSettings.UserSettings('Bob')

#this line is set to fetch frames 1,2,3 from trajectory BioSimGrid_GB-STH_1
datasettings = FCSettings.FCSettings(USettings,[['BioSimGrid_GB-STH_1',[1,2,3]]])

#we have a frame collections object that will store the frames fetched
FC = FrameCollection.FrameCollection(datasettings)

#get first frame
frame1 = FC.getNextFrame()
#get frame 1's coordinates
xyz = frame1.getCoordinates()
#get frame 1's metadata - frame's temperature
temp = frame1.getFrameTemperature()
```

Fig. 7. An example of a Python retrieval routine for getting frames 1, 2, 3 from trajectory 'BioSimGrid_GB-STH_1' in lines 1–11. Lines 13–18 illustrate how the frame object is used to access the actual data.

```
from BioSim.DataRetrieval import FrameCollection, FCSettings
from BioSim.Analysis import AverageStructure
from BioSim.Settings import UserSettings

USettings = UserSettings.UserSettings('Bob')
#The necessary initial imports and user specific settings are made.
datasettings = FCSettings.FCSettings(USettings, [[ 'BioSimGrid_GB-STH_1' ,[11,12,13,14,15]]])
datasettings.setResidueSerialNo([110,111,112])
FC = FrameCollection.FrameCollection(datasettings)
myAS = AverageStructure.AverageStructure(FC)
myAS.textFilename = 'av.txt'
myAs.createAsText()
```

Fig. 8. A Python analysis script that calculates the average structure of a molecule. This shows how an existing analysis tool (BioSim.Analysis.AverageStructure) can be used in a BioSimGrid script.

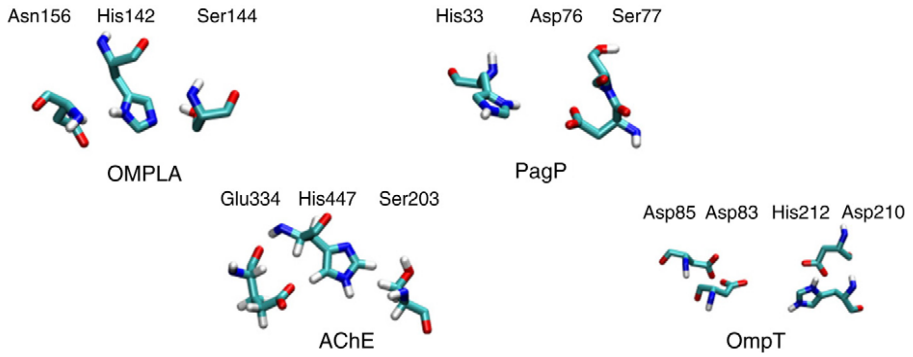


Fig. 9. The four enzymes: OMPLA, AChE, OmpT and PagP used in a comparative analysis of molecular dynamics simulations.

measurements for the intactness of the active site, is applied across four enzymes of similar function. BioSimGrid has allowed these four enzymes from distinct origins to be compared seamlessly in a uniform fashion over 17 trajectories. This comparative analysis would otherwise be complicated to the extent where it is virtually impossible with conventional lab resources. This cross-trajectory analysis has enabled us to explore functional patterns of conformational dynamics showing the similarities and differences of the dynamics of these four unrelated enzymes [18].

7. Discussion

BioSimGrid deals with two different types of data storage for its metadata and data. It has multiple homogeneous relational databases (Oracle databases) at different sites storing identical metadata. Grid middleware such as OGSA-DAI [19] is not used because the functionalities of distributed data querying (joint table querying) and integration of heterogeneous data sources are not immediately required. In order to maximise data delivery speed, BioSimGrid opted to deliver slices of trajectory on demand. It seeks to achieve a finer granularity of data access through the concept of frame collection (see Section 3.1) as compared to moving a complete trajectory to the processing element. This is why BioSimGrid develops its own retrieval components instead of using a grid component such as GridFTP [20] which is more suited for file-based transferring. Hence, for its specific purposes, BioSimGrid deposition and the retrieval tools are built on a combination of the Oracle in-built replication mechanism and the SRB replication functions to federate and replicate its data and metadata as described in Section 3.2. Another advantage of using SRB is that it facilitates a finer granularity of data access. The SRB interface allows an opening of a file remotely with on-demand data streamed across the network to mimic a local file access. It comes with a Python SRB Interface which conveniently integrates into the frame collection retrieval modules.

The replication of the metadata to each site and the duplication of the flat files is used to protect the system against hardware failure and data corruption. For example if one site fails, SRB will automatically switch to a secondary site to retrieve the replicated copy of the flat file. The use of RAID 5 storage for the flat files offers additional protection as we are able to recover from a hard disk drive failure without interruption to the service. In addition each file has an associated checksum which is validated between each of the replicated copies to ensure data integrity.

The heart of the SRB system, which is the MCAT database, is backed up at frequent intervals throughout the day, to an offsite machine. In the event of a Oracle master failure, any data deposition will be suspended but data retrieval can still be performed using the replicated slaves' Oracle databases. Once the Oracle master is brought back on line deposition can be performed again.

8. Conclusions and future work

BioSimGrid has enabled a more efficient sharing and post-processing of biomolecular simulation data within the research community. It allows access to geographically remote trajectories in a coordinated way and provides a set of uniform analysis tools for facilitating comparative analysis on different simulation data types. Future work on this project will focus on extending the current architectural design to allow users to use BioSimGrid from their desktops instead of logging into the host machines. We envisage a greater transparency in terms of data access from the repository by exposing the current services as web services to the users. This will in turn provide a platform and language independent way of accessing data in BioSimGrid. There is also the potential of extending BioSimGrid to utilise the UK National Grid Service (NGS) [21] which offers over 2000 processors and 36 TB of data storage capacity. This will imply a significant speed up in data simulation and promise a better handling of larger data sets to contribute to new science discoveries in biomolecular simulation research.

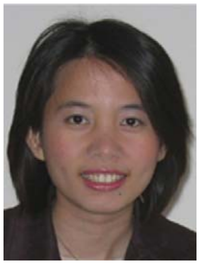
Acknowledgements

This work is supported by BBSRC and DTI. We thank our collaborators in the BioSimGrid Consortium: Charles Laughton, Adrian Mulholland, Leo Caves, David Moss, Oliver Smart, Marc Baaden and Katherine Cox.

References

[1] C.F. Wong, J.A. McCammon, Protein simulation and drug design, Adv. Protein Chem. 66 (2003) 87–121.
[2] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the Grid: Enabling scalable virtual organizations. Int. J. Supercomput. Appl. (2001).
[3] I. Foster, C. Kesselman (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.
[4] K. Tai, S. Murdock, B. Wu, M.H. Ng, S. Johnston, H. Fangohr, S.J. Cox, P. Jeffreys, J.W. Essex, M.S.P. Sansom, BioSimGrid: towards a worldwide repository for biomolecular simulations, Org. Biomol. Chem. 2 (2004) 3219–3221.
[5] Python, http://www.python.org, 1995.
[6] PyMOL, http://www.pymol.org, 2002.
[7] The Molecular Modeling Toolkit, http://starship.python.net/crew/hinsen/MMTK, 1997.
[8] C. Baru, R. Moore, A. Rajasekar, M. Wan, The SDSC storage resource broker, in: Proceedings of CASCON Conference, Toronto, Canada, 1998.
[9] SDSC Storage Resource Broker, http://www.sdsc.edu/srb/, 2005.
[10] Oracle Database 10g, http://www.oracle.com/technology/products/database/oracle10g/index.html, 2005.
[11] P.K. Weiner, P.A. Kollman, AMBER: Assisted model building with energy refinement. A general program for modeling molecules and their interactions, J. Comput. Chem. 2 (1981) 287–303.
[12] H.J.C. Berendsen, D. van der Spoel, R. van Drunen, GROMACS: A message-passing parallel molecular dynamics implementation, Comp. Phys. Comm. 91 (1995) 43–56.
[13] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L.V. Kal, R.D. Skeel, K. Schulten, NAMD — A parallel, object-oriented molecular dynamics program, Int. J. Supercomput. Appl. High Perform. Comput. 10 (1996) 257–268.
[14] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: A program for macromolecular energy, minimization, and dynamics calculations, J. Comput. Chem. 4 (1983) 187–217.
[15] B. Wu, M. Dovey, M.H. Ng, K. Tai, S. Murdock, H. Fangohr, S. Johnston, P. Jeffreys, S. Cox, J. Essex, M.S.P. Sansom, A Web/Grid portal implementation of BioSimGrid: A biomolecular simulation database (abstract), J. Digit. Inf. Manag. 2 (2) (2004) 74–78.
[16] International Organization for Standardization Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code, http://www.iso.org/iso/en/prods-services/iso3166ma/04background-on-iso-3166/iso3166-2.html, 1998.
[17] An Open Source Project Numerical Python, http://numeric.scipy.org/numpy.pdf, 2001.
[18] K. Tai, M. Baaden, S. Murdock, B. Wu, M.H. Ng, S. Johnston, S. Cox, J.W. Essex, M.S.P. Sansom, Active-site dynamics of hydrolases: comparative analysis of molecular-dynamics simulations via the BioSimGrid database, Biochemistry (2006) (in preparation).
[19] The OGSA-DAI Project, http://www.ogsadai.org.uk, 2003.

[20] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, The globus striped GridFTP framework and Server, in: Proceedings of Super Computing (SC05), Seattle, 2005.
[21] National Grid Service, http://www.ngs.ac.uk/, 2005.


M.H. Ng is a research fellow at the Southampton Regional e-Science Centre. She received her B.Eng. degree in computer engineering from the University of Southampton in 1999 and doctorate in electronics and computer science at the same university in 2003. Currently, her interest is in interdisciplinary e-science research focusing on developing the Grid database for biomolecular simulation data. Her role is in the software design and implementation of the BioSimGrid project.

S. Johnston is currently finishing his Ph.D. at the Southampton Regional e-Science Centre. He received an M.Eng. degree in software engineering from the Department of Electronics and Computer Science (ECS) at the University of Southampton. His research interests include the management and organisation of large volumes of data enabling Data Grid functionality to non-technical users. He works on the design and implementation of the BioSimGrid project, assisting with the management of large simulation data sets.

B. Wu is a research associate in the Department of Biochemistry and the e-Science Centre at the University of Oxford. He received his first class B.Sc. in mathematics from the University of Zhejiang and M.Sc. in software engineering from the Beijing University of Technology in China. He obtained a Ph.D. in computing from the School of Informatics at the University of Bradford. His interests are large bioinformatic systems, Grid computing, distributed computing and formal software engineering. He has been architecting and developing the distributed BioSimGrid system since March 2003 and designed and implemented the BioSimGrid Web Portal.

S.E. Murdock is a research fellow in the Department of Chemistry in the University of Southampton. He received a B.Sc. in theoretical physics (1998) and a Ph.D. in computational chemistry (2001) from Queen's University Belfast. His main research interests are in the fields of biomolecular modelling and software development. He is heavily involved in developing the analysis toolkit of BioSimGrid.

K. Tai is a Research Associate in the Department of Biochemistry, University of Oxford. He developed the deposit module in BioSimGrid and has been in charge of the scientific design of the database. He received a B.Sc. (Hon) from California Institute of Technology (1998) and Ph.D. in chemistry from the University of California, San Diego (2002) with Prof. J. Andrew McCammon. His research interests include simulations of biomolecular systems. He is a member of the Royal Society of Chemistry and an advisory member of the Green Economics Institute.

H. Fangohr is a computational physicist and lecturer in computational methods at the University of Southampton in the United Kingdom. He obtained his undergraduate degree in physics at the University of Hamburg (Germany) before he pursued his Ph.D. in the Computer Science Department at the University of Southampton. His interests range from high-performance computer simulations of condensed matter systems to improvements of numerical methods and algorithms.

S. Cox is a Professor of Computational Methods in the Computational Engineering Design Research Group within the School of Engineering Sciences at the University of Southampton. He is also the Technical Director of the Southampton Regional e-Science Centre. He received his doctorate in electronics and computer science, and a degree in mathematics and physics. His research interests include computational algorithms, commercial distributed computing, engineering informatics and the Grid.

J.W. Essex is a Reader in the School of Chemistry at the University of Southampton. He received his undergraduate degree in chemistry from the University of Oxford in 1989, and his doctorate from the same institution in 1992. He then worked at Yale University from 1992 to 1994 as a NATO/SERC Postdoctoral Fellow, before coming to Southampton as a Royal Society University Research Fellow. His research interests involve the development and application of computer simulation methods for problems of biological relevance.

M.S.P. Sansom is in the Department of Biochemistry at the University of Oxford, where he is head of the recently formed Structural Bioinformatics and Computational Biochemistry (SBCB) unit. He received his degree from Oxford in 1983, and worked for seven years in the University of Nottingham, before returning to Oxford. He has interests in many aspects of computational studies of membrane proteins (website: http://sansom.biop.ox.ac.uk) with special interest in ion channels, bacterial outer-membrane proteins, and the development of e-science and high-end computing for simulation and modelling of complex membrane systems.

P. Jeffrey is a Professorial Fellow at Keble, Director of the Oxford University Computing Services, Director of the Oxford University e-Science Centre and Co-Director of the e-Horizons Institute. Previously, his research interests were in the broad field of particle physics research, but they are now focused in e-science, http://e-science.ox.ac.uk/. Paul established the Oxford e-Science Centre and recently the Oxford Interdisciplinary e-Research Centre which will expand on the work of the e-Science Centre. His current research is in the area of creating a University-wide infrastructure to support new initiatives made possible by a new computing paradigm, the 'Grid'. He is also a primary investigator in a number of e-science projects across the University and acts as a consultant for the Department of Trade and Industry.