W724–W731 *Nucleic Acids Research, 2010, Vol. 38, Web Server issue* *Published online 6 June 2010*
doi:10.1093/nar/gkq503

Opal web services for biomedical applications

Jingyuan Ren*, Nadya Williams, Luca Clementi, Sriram Krishnan and Wilfred W. Li*

National Biomedical Computation Resource, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA

Received March 25, 2010; Revised May 12, 2010; Accepted May 18, 2010

ABSTRACT

Biomedical applications have become increasingly complex, and they often require large-scale high-performance computing resources with a large number of processors and memory. The complexity of application deployment and the advances in cluster, grid and cloud computing require new modes of support for biomedical research. Scientific Software as a Service (sSaaS) enables scalable and transparent access to biomedical applications through simple standards-based Web interfaces. Towards this end, we built a production web server (http://ws.nbcr.net) in August 2007 to support the bioinformatics application called MEME. The server has grown since to include docking analysis with AutoDock and AutoDock Vina, electrostatic calculations using PDB2PQR and APBS, and off-target analysis using SMAP. All the applications on the servers are powered by Opal, a toolkit that allows users to wrap scientific applications easily as web services without any modification to the scientific codes, by writing simple XML configuration files. Opal allows both web forms-based access and programmatic access of all our applications. The Opal toolkit currently supports SOAP-based Web service access to a number of popular applications from the National Biomedical Computation Resource (NBCR) and affiliated collaborative and service projects. In addition, Opal's programmatic access capability allows our applications to be accessed through many workflow tools, including Vision, Kepler, Nimrod/K and VisTrails. From mid-August 2007 to the end of 2009, we have successfully executed 239 814 jobs. The number of successfully executed jobs more than doubled from 205 to 411 per day between 2008 and 2009. The Opal-enabled service model is useful for a wide range of applications. It provides for interoperation with other applications with Web Service interfaces, and allows application developers to focus on the scientific tool and workflow development. Web server availability: http://ws.nbcr.net.

INTRODUCTION

Biomedical data have become increasingly complex, and the applications that analyze them often require large-scale high-performance computing resources with a large number of processors and memory. The recently launched 1000 Genomes Project (http://www.1000genomes.org) promises to increase greatly the effectiveness of genome-wide association studies (GWAS), and the potential for pharmacogenomic discoveries and personalized medicine (1). Efforts to define 'human druggable genome' to identify novel drugs or repurpose existing drugs have resulted in new computational tools and databases that enable access to real time or precomputed analysis results using distributed computing resources (2). The National Biomedical Computation Resource (NBCR, http://www.nbcr.net) provides biomedical computing resources with a focus on multi-scale biomedical research, dealing with scales at the molecular level for Computer Aided Drug Discovery (CADD), subcellular level for calcium signaling and organ level for patient specific modeling. The complexity of application deployment and the advances in distributed computing require new modes of support for biomedical research. The sSaaS enables scalable and transparent access to biomedical applications running remotely on a desktop, cluster, grid or cloud computing resources through open standards-based Web services. The availability of these application-specific services enables the adoption of workflow tools to handle complex analytical procedures. The NBCR web server http://ws.nbcr.net contains a registry of several classes of biomedical application services, including docking and virtual screening using Autodock 4 (3) and Autodock Vina (4); electrostatics analysis using PDB2PQR (5,6) and APBS

*To whom correspondence should be addressed. Tel: +1 858 246 0336; Fax: +1 858 822 1619; Email: j2ren@ucsd.edu
Correspondence may also be addressed to Wilfred W. Li. Tel: +1 858 534 0591; Fax: +1 858 822 1619; Email: Wilfred@sdsc.edu

(7); off-target analysis using SMAP (8,9–11); and motif discovery using the MEME suite (12). While each application may be used in many different scenarios, they are provided by NBCR within the context of CADD, and more applications services dealing with other aspects of multiscale modeling will become available over time. These applications are exposed as Web services using the open source Opal toolkit (13,14), which enables users to access applications using web interfaces automatically generated in the Opal Dashboard or programmatically using the SOAP protocol with clients in Perl, Python, or Java programming languages. The availability, stability and scalability provided by Opal or similar Web services enable the availability of the same or alternative applications to be provided by different providers, resulting in better cyberinfrastructure for biomedical research.

The rest of the paper is organized as follows: the next section deals with the usage scenario and description of the available applications; the subsequent section discusses the programmatic access of the Web services using Python as an example; the following section deals with advanced use cases including workflow composition; thereafter, the Opal toolkit and key new features are dealt with; in the next section, a, there is a discussion of this paper and related works; and the last section discusses the conclusions and future work.

SCIENTIFIC SOFTWARE EXPOSED AS WEB SERVICES

Some of the hurdles in making scientific software accessible have been the installation, maintenance and upgrades, let alone the hardware and environmental cost of maintaining a machine room. Through our experience providing MEME as a biomedical community service, we have developed the Opal toolkit (14), which enables scientific application developers to set up web services with both web form access and programmatic access easily without needing to change any of their own codes. We have since expanded the number of applications provided through the Opal toolkit, and built advanced workflows that leverage these distributed and scalable Web services within the context of CADD, to help increase the translational impact of the computational tools from NBCR and other related activities.

Opal Dashboard with basic or advanced web interfaces

The Opal Dashboard (Figure 1A) (15) at http://ws.nbcr.net provides a listing of all the Web services, including applications for docking analysis, electrostatics calculations, off-target analysis and motif search/discovery. The 'List of Applications' tab shows the list of applications. The 'Search' textbox allows a user to use keywords to search for applications. Each service has a service name, a basic description of the application, a tutorial link and a programmatic access (Web service) URL. The service name links to an automatically generated simple or a customized web page. The former accepts command line arguments for an application, whereas the latter provides the command line arguments in a web form (Figure 1B). The tutorial link contains instructions for learning and sample input and output for testing. The Web service URL is used only for programmatic access of the application.

Web services for CADD

The application services are provided to facilitate various aspects of CADD. A summary is provided in Table 1, along with the references for the application. While there may be alternative applications for each purpose, due to space constraint, we will only discuss similar applications from the literature or online services in the 'Discussion' section. Some usage scenarios of these services are as follows: a user has a particular drug target receptor in mind, e.g. the influenza neuraminidase (NA), and would like to search the NCI diversity set (NCIDS http://dtp.nci.nih.gov/branches/dscb/div2_explanation.html) library for possible hits (16). The partial charges for the receptor may be added using PDB2PQR, or added with the Prepare Receptor service (part of MGLTools, currently at v1.5.4) using default parameters. A user may then use AutoDockTools (part of MGLTools, currently at v1.5.4), to specify the grid box center, spacing and number of grid points to generate a custom grid parameter file (GPF), or provide a reference GPF template file with the desired grid box information. As the atom types created in the GPF is dependent upon the ligand or a library of ligands, the user would use the Prepare GPF for Library service available at http://kryptonite.nbcr.net/opal2/dashboard, where a user may upload his own library already prepared for use with AutoDock4, or have it prepared with the Prepare Ligand service, or just choose a library already available on the server. The NCIDS and ZINC (17) libraries are some of the example libraries available. Once AutoGrid is complete, the AutoDock Virtual Screening service may be used for the virtual screening. If the user is interested in finding PDB structures that have a similar binding site to the NA protein, then he or she may use the SMAP analysis to perform an off-target analysis. If the user is interested in identifying conserved motifs in the NA protein for all pandemic N1 proteins, the MEME suite of programs may be used for such purposes, followed by mutational analysis, including electrostatic analysis to examine the effect of these mutations.

Opal Web service usage statistics

There has been a growth in the number of jobs for various applications on our web server. Figure 2 shows a comparison of the numbers of jobs successfully executed in 2008 and 2009. We can see from the graph that the usage for our applications increased and the total number of jobs executed in 2009 was approximately double of that in 2008. There were 75 059 jobs successfully executed in 2008, and 150 041 in 2009.

PROGRAMMATIC ACCESS OF OPAL SERVICES

The Opal toolkit consists of the Opal server and the Opal client (Figure 3). The Opal server is responsible to managing and providing the web services, while the Opal client accesses web services on the Opal server. The

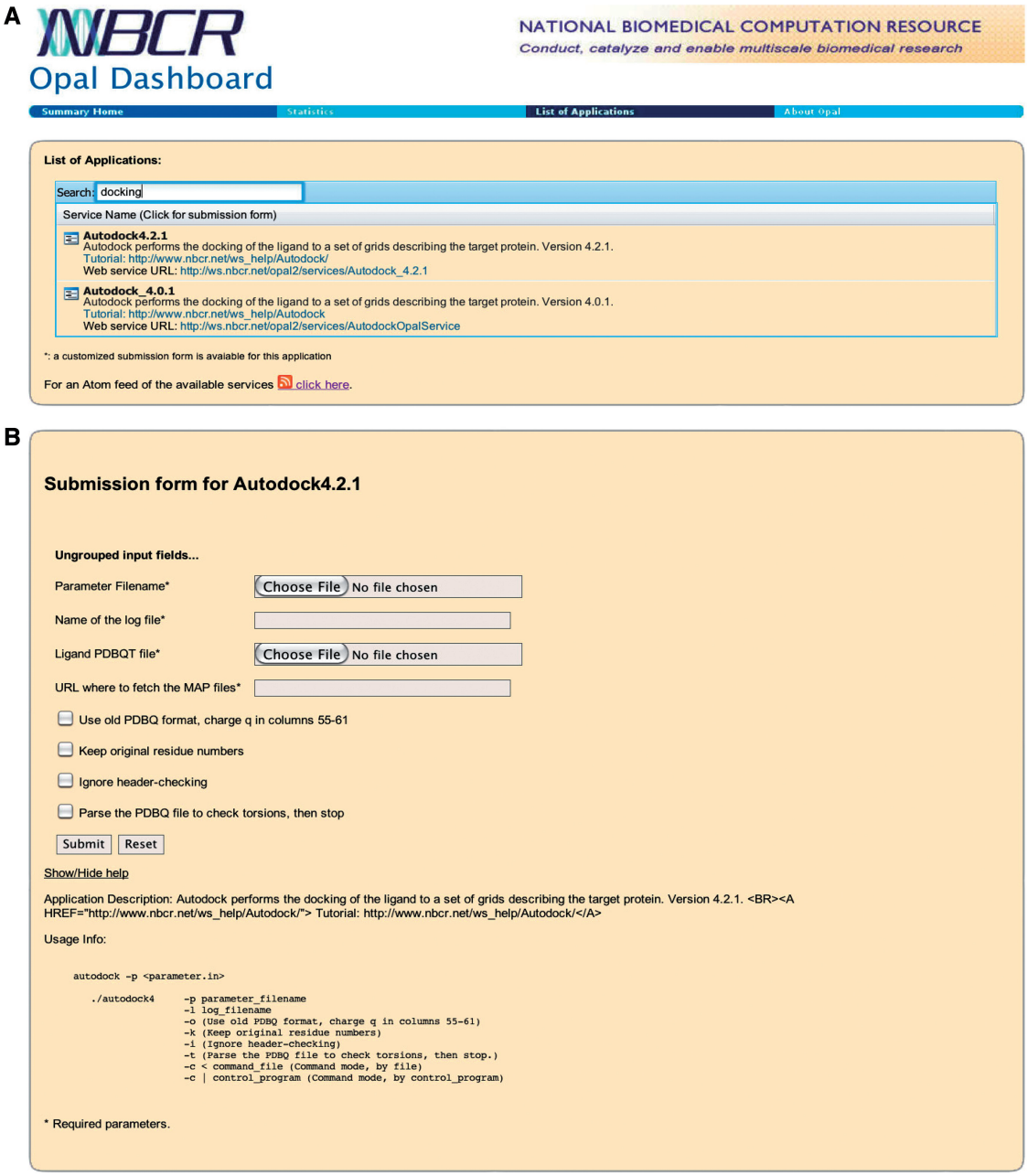

Figure 1. (**A**) The Opal Dashboard lists all the available applications, with different versions where applicable, and provides a search interface using keywords. (**B**) A customized interface is automatically generated based upon required AutoDock 4 command line arguments.

set up of the Opal server is discussed in the 'Opal server management and configuration' section, and is only of interest to users who want to provide Opal services for their own applications. The Opal client enables easy programmatic access of any Opal service. A generic Opal client, currently available in Python and Java, supports user actions such as launching a job, querying job status, getting job outputs and works with any Opal Web service. An Opal client in Perl is provided as an example for MEME, and a generic Perl client will be made available soon.

Figure 3 illustrates how the Python client may be used to access the PDB2PQR service (Table 2). More Opal client documentation is available at http://www.nbcr.net/pub/wiki/index.php?title=Opal_Client.

For example, a user can launch a PDB2PQR job with the following command to convert '1a1p.pdb' to 'output.pqr', assuming '1a1p.pdb' is already in the user's local directory:

```
python GenericServiceClient.py \
 -l    http://ws.nbcr.net/opal2/services/
Pdb2pqrOpalService \
 -r launchJob \
 -a ''-=amber 1a1p.pdb output.pqr'' \
 -f 1a1p.pdb
```

Table 1. Opal web services, basic usage, and the programmatic access URLs

Name	Purpose	Versions	Programmatic access URLs
Autodock (3)	Docking experiment with a single protein and a ligand	4.0.1 4.2.1	http://ws.nbcr.net/opal2/services/AutodockOpalService http://ws.nbcr.net/opal2/services/Autodock_4.2.1
Autodock Vina (18)	Docking experiment with a single protein and a ligand	1.0.2	http://kryptonite.nbcr.net/opal2/services/AutodockVina
Autodock Vina VS	VS of a ligand library using a single receptor	1.0.2	http://kryptonite.nbcr.net/opal2/services/AutodockVina_Screening
Autodock VS	VS of a ligand library using a single receptor	AD4.2.3/ MGL1.5.4	http://kryptonite.nbcr.net/opal2/services/autodock
Autogrid (3)	Generates required MAP files prior to AutoDock docking experiment	4.0.1 4.2.1 4.2.3 special	http://ws.nbcr.net/opal2/services/AutogridOpalService http://ws.nbcr.net/opal2/services/Autogrid_4.2.1 http://kryptonite.nbcr.net/opal2/services/autogrid
APBS (7)	Electrostatics calculations of biomolecules	1.2.1b serial 1.2.1b parallel	http://ws.nbcr.net/opal2/services/ApbsOpalService http://ws.nbcr.net/opal2/services/ApbsParallelOpalService
FIMO (12)	Find individual motif occurrences in a sequence database	4.3.0	http://ws.nbcr.net/opal2/services/FIMO_4.3.0
GLAM2 (12)	Identification of gapped motifs in DNA or protein sequences	4.3.0	http://ws.nbcr.net/opal2/services/GLAM2_4.3.0
GLAM2SCAN (12)	Identify GLAM2 motifs in a sequence database	4.3.0	http://ws.nbcr.net/opal2/services/GLAM2SCAN_4.3.0
GOMO (12)	Identify GO terms linked to target genes with a given DNA binding motif	4.3.0	http://ws.nbcr.net/opal2/services/GOMO_4.3.0
MAST (12)	Identify given motifs in sequence databases	4.3.0	http://ws.nbcr.net/opal2/services/MAST_4.3.0
MEME (12)	Identify conserved motifs given input DNA or protein sequences	4.3.0	http://ws.nbcr.net/opal2/services/MEME_4.3.0
PDB2PQR (5,6)	Utility to convert PDB to PQR format used in continuum electrostatics calculations	1.6.0	http://ws.nbcr.net/opal2/services/Pdb2pqrOpalService
Prepare GPF (19)	Generates Grid Parameter File used for AutoGrid with a single protein and a single ligand	MGLTools 1.5.4	http://ws.nbcr.net/opal2/services/PrepareGPFOpalService
Prepare GPF for Lib	Generates Grid Parameter Files for a single protein and a ligand library	MGLTools 1.5.4	http://kryptonite.nbcr.net/opal2/services/prepareGPF
Prepare Receptor (19)	Prepares a protein (receptor) for use in a AutoDock experiment	MGLTools 1.5.4	http://ws.nbcr.net/opal2/services/PrepareReceptorOpalService
SMAP DB Search (8–11)	Search SMAP database for 3D ligand binding site similarity	2.2alpha	http://kryptonite.nbcr.net/opal2/services/SMAPPairComp
SMAP Pair Comp. (8–11)	Pairwise comparison of 3D ligand binding site similarity	2.2alpha	http://kryptonite.nbcr.net/opal2/services/SMAPDBSearch

All these services have individual publications and online tutorials that describe their detailed usage. As with any online predictive programs, the experiments should be prepared carefully, and the results should be interpreted with caution, and within appropriate biological context.

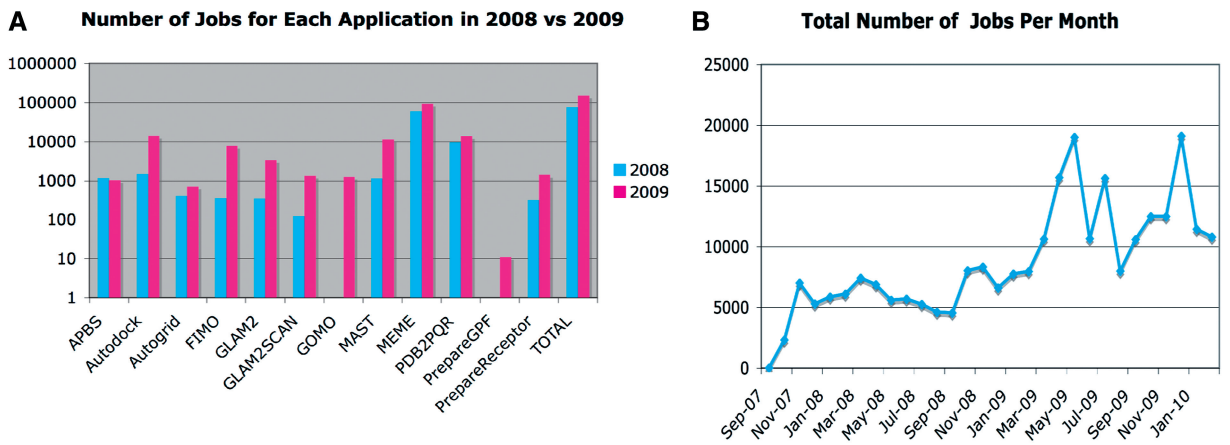

Figure 2. (**A**) Comparison of numbers of jobs successfully executed in 2008 and 2009 in a semi-log graph. (**B**) Number of total jobs successfully executed per month.

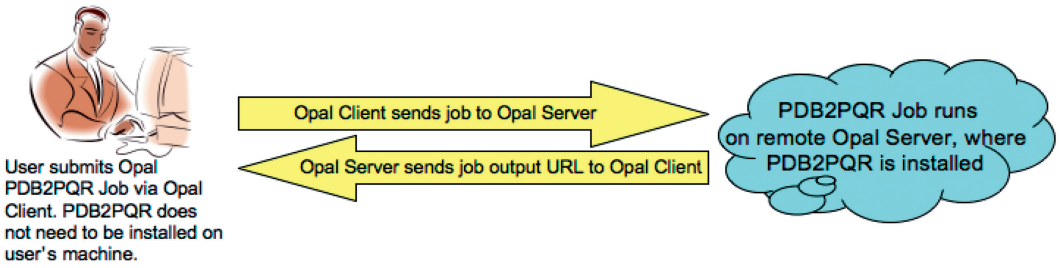

Figure 3. Opal client/server architecture – running a PDB2PQR job as a Web service call.

Table 2. Common Opal Python client commands

Launch job	Query job status	Get job outputs

`python GenericServiceClient.py\`
`-l $opal_service_url\`
`-r launchJob\`
`-a ''$cmd_line_args''\`
`-f ''$inputfiles_list''`
	`python GenericServiceClient.py\`
	`-l $opal_service_url\`
	`-r queryStatus\`
	`-j $job_id`
		`python GenericServiceClient.py\`
		`-l $opal_service_url\`
		`-r getOutputs\`
		`-j $job_id`

The output for this command will include a job ID, e.g. app1234567890, along with a job base output URL. With the job ID the user can query the job status as in the command below:

```
python GenericServiceClient.py \
-l http://ws.nbcr.net/opal2/services/Pdb2pqrOpalService \
-r queryStatus \
-j app1234567890
```

A notable feature of Opal Web services is that the job output URL may be used as the input URL for compatible Web services. This is discussed in the next section for workflow composition.

USING OPAL SERVICES WITH WORKFLOW TOOLS

As indicated in the section on usage scenario, CADD processes are very complex, and involve many steps, some automatically, others requiring manual intervention. A number of workflow tools, both commercial and open source, have been developed to support the construction and execution of complex workflows. Some well known examples include Pipeline Pilot (http://accelrys.com/products/pipeline-pilot/), Taverna (20,21), Kepler (22), Vistrails (23) and Vision (4). In all these tools, a user is allowed to run complex experiments automatically, after careful parameter tuning for the individual steps. Often, an experiment step is represented in these workflow tools as a node, which has input and output ports. Two nodes can be connected together so that the outputs of the first node may be inputs of the second node and may trigger the execution of the second node. For example, a user often thinks of an AutoDock VS experiment as three major steps, prepare the receptor, compute the grids, and run the virtual screening. The completion of a 'prepare receptor' node triggers the execution of the 'compute grids' node, which in turn triggers the execution of the 'AutoDock virtual screening' node.

Vision and NBCR CADD pipeline

Many of these Opal web services are easily accessible in Vision (4) using the Vision Web service module. Vision provides a visual programming environment, and is easily integrated with the Python Molecular Viewer (PMV) and AutoDockTools (ADT) to automate many of the setup, visualization and analysis of AutoDock experiments. The Vision Web service module leverages the automatic interface generation feature to provide users with the same user interface in the web form, as configured on the server side.

The workflow illustrated in Figure 4 accesses the Prepare Receptor Web services from http://ws.nbcr.net, and Prepare GPF, Autogrid, and Autodock Virtual Screening Web services from http://kryptonite.nbcr.net. The Opal output URL is often used as the input URL for the next step of the workflow. These workflow units are now packaged into the MGLTools and allow advanced users to build customized workflows.

For users who wish to use preexisting workflows, the AutoDock VS experiment may be accessed using Vision networks developed as part of a prototype of the NBCR CADD pipeline (http://nbcr.net/pub/wiki/index.php?title=CADD_Pipeline). In addition, users may also take advantage pre-built workflows for post-analysis and visualization of docking results. For example, a user may use a Vision network to iterate through a virtual screening result directory with the different docking log files, and play through the different ligand conformations using built-in functionalities of ADT. Thus, more customized analysis routines may be published and shared with many users.

Support of other workflow tools

A full description of workflow tools such as Kepler, Nimrod/K (24) or VisTrails is beyond the scope of this paper. We simply point out that a generic Opal client is available as an actor in Kepler, which can be used to call any Opal service. To get the Opal actor in Kepler, then user can type 'web services' in 'Search Components' textbox and then drag 'OpalClient' to the workflow

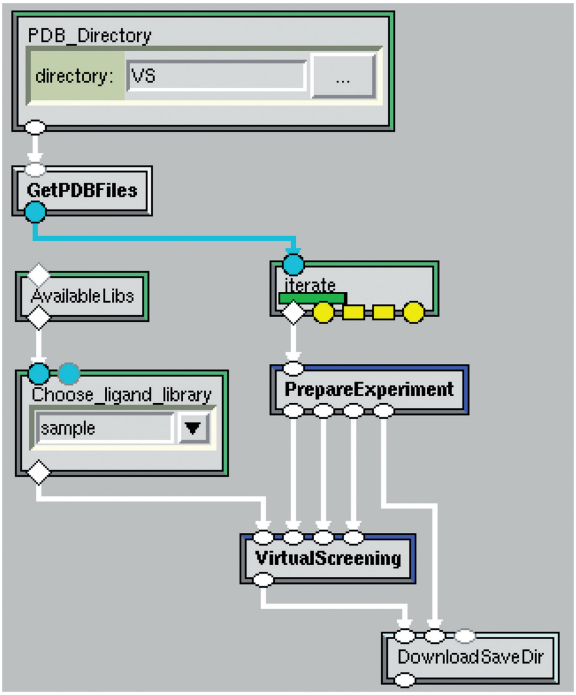

Figure 4. Autodock virtual screening workflow in Vision. A user may specify an input directory, e.g. VS, which contains one or more PDB files, corresponding template GPFs and DPFs. A user may select a ligand library from the list of libraries. After the virtual screening job finishes, the top 500 docking results along with summarized results are automatically downloaded to the user's local machine.

composition canvas in Kepler. Then the user can then click on 'OpalClient' to enter a web service URL. After the user clicks on 'commit' and then clicks on the 'OpalClient' actor again, the updated actor will show the automatically generated Opal web form. Similarly, the Kepler based Nimrod/K allows dynamic concurrent execution of Kepler's actors. A tutorial on using Kepler with Nimrod/K and Opal services can be found on http://nbcr.net/pub/wiki/index.php?title=Kepler. Similar to Vision and Kepler, all Opal services are callable from VisTrails, for which an automatic interface generation of the Opal web form will become available in the near future. Documentation on installing the required components for using VisTrails with Opal services and tutorials can be found on http://nbcr.sdsc.edu/pub/wiki/index.php?title=VisTrails. In VisTrails, the user can use the 'ExecuteOpalJob' node to launch an Opal job. This node has four inputs, command line arguments, number of processors (for parallel jobs only), list of input files and the web service URL.

OPAL SERVER CONFIGURATION AND MANAGEMENT

The Web services described earlier have been deployed using the Opal toolkit, which enables advanced users to automatically wrap scientific applications running on cluster and Grid resources as Web services, and to provide Web-based and programmatic access to them, without any modification to the scientific codes. Thus, together with the generic clients or workflow tools described above, a user may access or develop workflows that leverage not only those Opal services provided by NBCR, but also the ones provided by anyone using the Opal toolkit. As the advantages of using the Opal toolkit has been published extensively elsewhere (15,13,14), we will only highlight a few key points relevant to the management and usage of the Opal services. Additional documentation on installation may be found online at http://opal.nbcr.net.

Automatic Web interface generation

The Opal Dashboard (15), available with any Opal installation, provides a consolidated point of entry for accessing information about Opal services, and invoking remote scientific applications (Figure 1A). Opal also provides a mechanism to automatically create Web interfaces for job submission using provided application specific metadata. A detailed description of automatic interface generation is presented in ref. 15. Briefly, advanced users can make sophisticated configuration files to generate advanced web forms, which contain an input field for each input parameter, as specified by the command-line arguments of the scientific application. Parameters may be grouped together and check boxes and radio buttons may be added (Figure 1B). This feature is great for making an application quickly available to the intended users through the web, in addition to the programmatic access through SOAP Web service calls. Users may also design their own web pages and make Web service calls to Opal services if desired, as is the case for MEME and PDB2PQR.

Resource management and statistics report

The Opal server logs some information for job, including job ID, job URL, start time, activation time, end time, client IP, service name and job result status. Using this information, the Opal Dashboard provides a variety of charts showing usage information, as shown in Figure 5. Note that these charts are completely anonymous—no user information is displayed on the charts. Each Opal job is associated with a unique random ID given only to the user who submitted the job. Immediately after the user submits a job on the web form, the user gets a unique URL that automatically updates the status of the job. More details about Opal state management is available in ref. 13. A new Opal version to be released in May 2010 includes email notification of the job URL and status updates. Another feature already available is necessitated by the popularity of our service. Opal 2.2 and higher supports a new IP address filtering feature that helps prevent Denial of Service attacks. An administrator may choose to enable this feature to deny all job requests from certain IPs, once the number of jobs per IP for the past hour is greater than the limit has been specified. IPs may also be exempted from any restrictions.

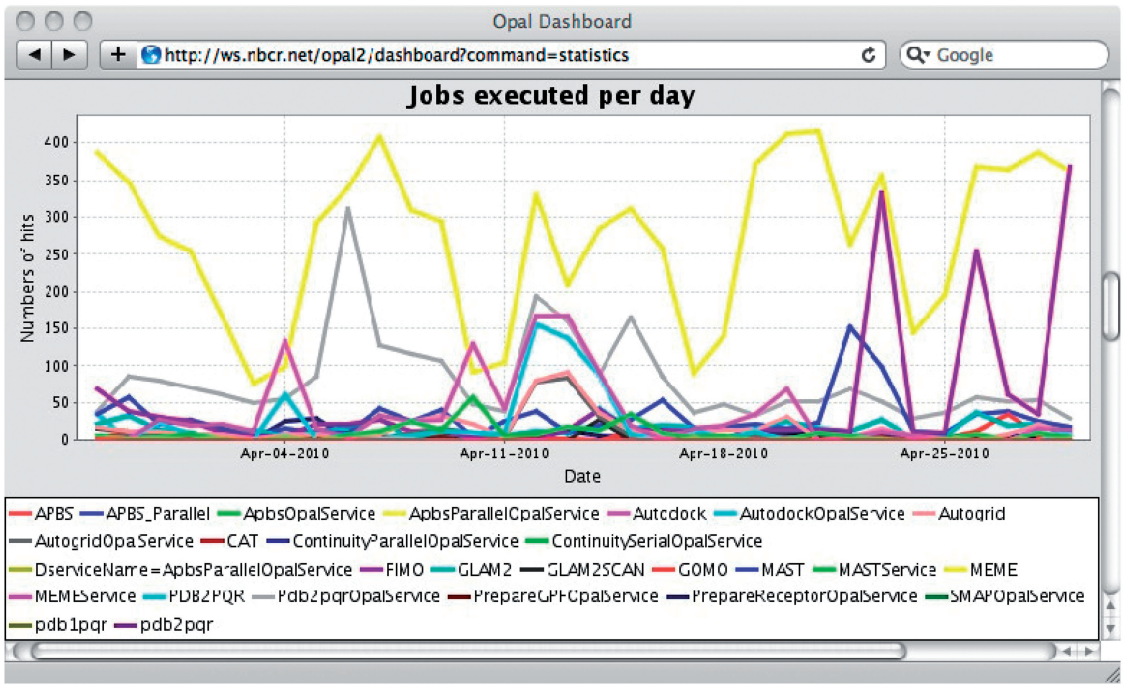

Figure 5. The 'Jobs Executed per Day' graph from the Opal server 'Statistics' page.

DISCUSSION

The Web services described in this paper provides a set of popular computational tools that may be used within the context of CADD, though not limited to it. The use of Opal toolkit may help provide a common platform for rapid deployment of scientific Web services. For example, many existing applications that perform similar functions, such as AlignAce (25) as opposed to MEME in motif discovery; use of semi-empirical quantum-chemical techniques (26) as opposed to PDB2PQR or AutoDockTools in partical charge calculation; Dock (27) as opposed to AutoDock in docking; webPIPSA (28) as opposed to APBS in electrostatics analysis, may be coupled in workflows or compared for consensus. Users are encouraged to study the original references for these application services, acquire the appropriate training and validate any predictions with experiments.

As the Web services we provide increase in popularity, there may be several ways to overcome the problem with scalability. While we have implemented a filter based upon the number of requests per hour, we are preparing an increased allocation of compute resources in response to any increased usage. In addition, we explore the use of Amazon EC2 commercial providers to become on-demand purveyor of the services through virtual machines that contain preconfigured Opal services in multicore machines or virtual clusters. On the other hand, as more identical or alternative applications become available as Web services, with the help of the Opal toolkit, it may relieve the pressure on any one provider. In other words, the Opal Web service interface remains unchanged, yet the resources behind the services will be dynamically allocated from different providers.

While Opal provides a simple toolkit to deploy Web services, other methods are also available (14). For example, EBI has a number of services deployed using the SoapLab toolkit (http://www.ebi.ac.uk/soaplab/), with good support by the Taverna workflow engine. The Protein Data Bank (PDB) provides a plethora of Web services related to structure based drug design. We anticipate the proliferation of Web services, and emerging standards to allow researchers to focus more on the biological problems, rather than deploying applications, and advance the cyberinfrastructure for biomedical research.

CONCLUSION AND FUTURE WORK

Since August 2007, the NBCR web server http://ws.nbcr.net has been continuously providing biomedical scientists easy access to a large collection of biomedical applications, with about a quarter of a million of jobs completed by the end of 2009. The key features of Opal Web services include the availability of sample clients for accessing Opal services, ease of access from workflow tools, and scalability through distributed computing using desktop, cluster, grid and cloud resources.

We plan to add several new features to Opal. Most notably, we plan to provide a REpresentational State Transfer (REST)-based API to invoke Opal services. REST is much more lightweight than SOAP-based implementations, currently used by Opal. Opal users will then be able to invoke Opal services with the help of basic HTTP tools such as curl. Second, we will develop a registry that can aggregate access to Opal services hosted at various locations. Third, we are in the process of improving the automatic interface generation to have better

support for Ajax (29), making the user interface more contemporary and easier to use. Additional effort may also be required to ensure fair use of the resources, and to ensure the best possible security and privacy of user data through the development of personalized data services. These improvements will make the Opal toolkit easier to use, and increase the number of Web services available for biomedical research.

More scientific applications will be provided in support of the Relaxed Complex Scheme in CADD (16), and the NBCR Summer Institute (http://si.nbcr.net) provides annual training on tools and services described in the paper, and expose users to emerging technologies that may benefit biomedical computing in the near future.

ACKNOWLEDGEMENTS

We appreciate the constructive suggestions of the editor in improving the manuscript. In addition, we thank Stefano Forli and Alex Perryman for AutoDock4 ligand libraries; John Irwin from UCSF for the ZINC library. We also wish to thank the reviewers of this manuscript for critical comments and suggestions.

FUNDING

Funding for open access charge: National Center for Research Resources (NCRR); National Institutes of Health (NIH); (P41 RR08605 award to NBCR).

Conflict of interest statement. None declared.

REFERENCES

1. Zhang,W. and Dolan,M.E. (2010) Impact of the 1000 genomes project on the next wave of pharmacogenomic discovery. *Pharmacogenomics*, **11**, 249–256.
2. Li,L., Bum-Erdene,K., Baenziger,P.H., Rosen,J.J., Hemmert,J.R., Nellis,J.A., Pierce,M.E. and Meroueh,S.O. (2009) BioDrugScreen: a computational drug design resource for ranking molecules docked to the human proteome. *Nucleic Acids Res.*, **38**, D765–D773.
3. Morris,G.M., Huey,R., Lindstrom,W., Sanner,M.F., Belew,R.K., Goodsell,D.S. and Olson,A.J. (2009) AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J. Computat. Chem.*, **30**, 2785–2791.
4. Sanner,M.F., Stoffler,D. and Olson,A.J. (2002) *ViPEr*, a visual Programming Environment for Python. In Proceedings of thre 10th International Python conference, 103-115. February 4-7, 2002. ISBN 1-930792-05-0.
5. Dolinsky,T.J., Czodrowski,P., Li,H., Nielsen,J.E., Jensen,J.H., Klebe,G. and Baker,N.A. (2007) PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res.*, **35**, W522–W525.
6. Dolinsky,T.J., Nielsen,J.E., McCammon,J.A. and Baker,N.A. (2004) PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Res.*, **32**, W665–W667.
7. Baker,N.A., Sept,D., Joseph,S., Holst,M.J. and McCammon,J.A. (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl Acad. Sci. USA*, **98**, 10037–10041.
8. Ren,J., Xie,L., Li,W.W. and Bourne,P. (2010) SMAP-WS: a parallel web service for structural proteonme-wide ligand binding site comparison. *Nucleic Acids Res*, **38**, W441–W444.
9. Xie,L. and Bourne,P.E. (2007) A robust and efficient algorithm for the shape description of protein structures and its application in predicting ligand binding sites. *BMC Bioinform.*, **8(Suppl. 4)**, S9.
10. Xie,L. and Bourne,P.E. (2008) Detecting evolutionary relationships across existing fold space, using sequence order-independent profile-profile alignments. *Proc. Natl Acad. Sci. USA*, **105**, 5441–5446.
11. Xie,L. and Bourne,P.E. (2009) A unified statistical model to support local sequence order independent similarity searching for ligand-binding sites and its application to genome-based drug discovery. *Bioinformatics*, **25**, i305–i312.
12. Bailey,T.L., Boden,M., Buske,F.A., Frith,M., Grant,C.E., Clementi,L., Ren,J., Li,W.W. and Noble,W.S. (2009) MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Res.*, **37**, W202–W208.
13. Krishnan,S., Clementi,L., Ren,J., Papadopoulos,P. and Li,W. (2009) Design and Evaluation of Opal2: A Toolkit for Scientific Software as a Service. *The 2009 IEEE Congress on Services (SERVICES-1 2009)*, July, 2009.
14. Krishnan,S., Stearn,B., Bhatia,K., Baldridge,K.K., Li,W. and Arzberger,P. (2006) Opal: Simple Web Services Wrappers for Scientific Applications. *In Proceedings of ICWS 2006, IEEE International Conference on Web Services*, 2006.
15. Clementi,L., Ding,Z., Krishnan,S., Wei,X., Arzberger,P., and Li,W. Providing Dynamic Virtualized Access to Grid Resources via the Web 2.0 Paradigm. *GCE07, Grid Computing Environments Workshop (Supercomputing 2007)*, Nov 2007.
16. Cheng,L.S., Amaro,R.E., Xu,D., Li,W.W., Arzberger,P.W. and McCammon,J.A. (2008) Ensemble-based virtual screening reveals potential novel antiviral compounds for avian influenza neuraminidase. *J. Med. Chem.*, **51**, 3878–3894.
17. Irwin,J.J. and Shoichet,B.K. (2005) ZINC–a free database of commercially available compounds for virtual screening. *J. Chem. Inform. Model.*, **45**, 177–182.
18. Trott,O. and Olson,A.J. (2010) AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Computat. Chem.*, **31**, 455–461.
19. Sanner,M.F. (1999) Python: a programming language for software integration and development. *J. Mol. Graphics Mod.*, **17**, 57–61.
20. Hull,D., Wolstencroft,K., Stevens,R., Goble,C., Pocock,M.R., Li,P. and Oinn,T. (2006) Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.*, **34**, W729–W732.
21. Qinn,T., Greenwood,M., Addis,M., Alpdemir,N., FerrisJ., Glover,K., Goble,C., Goderis,A., Hull,D., Marvin,D. *et al.* (2006) Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, **18**, 1067–1100.
22. Ludascher,B., Altintas,I., Berkley,C., Higgins,D., Jaeger,E., Jones,M., Lee,E., Tao,J. and Zhao,Y. (2005) Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience*, 1039–1065.
23. Bavoil,L., Callahan,S., Crossno,P.J., Freire,J., Scheidegger,C.E., Silva,C. and Vo,T. (2005) VisTrails: Enabling Interactive Multiple-View Visualizations. *In Proceedings of IEEE Visualization*, 135–142.
24. Abramson,D., Enticott,C. and Altinas,I. (2008) Nimrod/K: Towards Massively Parallel Dynamic Grid Workflows. *Conference on High Performance Networking and Computation, Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, 1–11.
25. Hughes,J.D., Estep,P.W., Tavazoie,S. and Church,G.M. (2000) Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.*, **296**, 1205–1214.
26. Bikadi,Z. and Hazai,E. (2009) Application of the PM6 semi-empirical method to modeling proteins enhances docking accuracy of AutoDock. *J. Cheminform.s*, **1**, 15.
27. Irwin,J.J., Shoichet,B.K., Mysinger,M.M., Huang,N., Colizzi,F., Wassam,P. and Cao,Y. (2009) Automated docking screens: a feasibility study. *J. Med. Chem.*, **52**, 5712–5720.
28. Richter,S., Wenzel,A., Stein,M., Gabdoulline,R.R. and Wade,R.C. (2008) webPIPSA: a web server for the comparison of protein interaction properties. *Nucleic Acids Res.*, **36**, W276–W280.
29. Holdener,A.T. III (2008) *Ajax: The Definitive Guild*, O'Reilly Media, 1st edn.