Open Grid Computing Environments

Marlon Pierce[1], Suresh Marru[2], Wenjun Wu[3], Gopi Kandaswami[4], Gregor von Laszewski[5], Rion Dooley[6], Maytal Dahan[6], Nancy Wilkins-Diehr[7], Mary Thomas[8]

[1] Community Grids Laboratory, Digital Science Institute, Indiana University, Bloomington, IN 47404
[2] Computer Science Department, Indiana University, Bloomington, IN 47404
[3] Computation Institute, University of Chicago, Chicago, IL
[4] Renaissance Computing Institute, Chapel Hill, NC
[5] Rochester Institute of Technology
[6] Texas Advanced Computing Center, University of Texas, Austin TX
[7] San Diego Supercomputing Center, San Diego CA
[8] San Diego State University, San Diego, CA

1. Introduction and Project Overview

The Open Grid Computing Environments (OGCE) project develops components and services that enable gateways to be built out of reusable parts. This extended abstract overviews our primary components and describes significant new efforts since the work described in [1]. In previous work, the major thrust of the OGCE has been to develop Java programming abstractions for the Grid and reusable portlet components. These are packaged for simple, reproducible builds using Apache Maven. Our current work focuses on extending this to a more comprehensive approach to science gateways. The portlet-based download is still actively developed and packaged. Additionally, we have added several new project components. The complete list of components is summarized below:

- OGCE Portal: this is a portlet component-based portal that provides a simple out-of-the-box science gateway with basic Grid functionality (Web clients to Globus GRAM, GridFTP, MyProxy, GPIR, and Condor).
- The OGCE Workflow Suite: this consists of both services and client components that are derived from the LEAD workflow tools.
- OGCE Axis Services: These are packaged, standalone, Apache Axis 2-based Web services that provide Grid information.
- Cyberaide Javascript Libraries: these are client libraries that can be used to build JavaScript clients to Grid and other services.
- GTLAB: These are client tag libraries that can be used by developers to build portlets, Google gadgets, and similar components to Grid and other Web services. GTLAB includes clients to OGCE Axis Services.
- Open Social Gadgets and Containers: We are developing gadget and containers as the successor to portlet-based components.

Developers can use all of these, or they can pick and choose desired tools. All code is open source and available via academic license. We also have various incubator projects that may graduate to full components. We now discuss the major components.

2. OGCE Workflow Suite

The workflow suite consists of two services and two clients. XRegistry is a service for storing information about other services. GFAC is a factory service that is useful for wrapping applications (binaries) as Web Services. GFAC can be used to generate both persistent and transient services. XRegistry is used to publish and discover these services. Actual workflows are graphically composed and executed using XBaya. XBaya is implemented as a Java Web Start application that can be launched from the OGCE portal. Finally, expert users can interact with the GFAC service to deploy new services through the GFAC portlet.

Both XBaya and the GFAC portlets are undergoing significant revisions. We are decoupling the workflow orchestration parts of XBaya from the graphical composer. The new orchestration engine is an extension of the Apache Orchestration Director Engine project. GFAC portlets are also being rewritten using Java Server Faces. By using the Apache portlet bridge, we can make these components available as portlets. Additionally, they can also be used to build standalone

web interfaces. Finally, the constituent beans that power Java Server Faces can be used in Java Swing interfaces.

The OGCE Workflow Suite tools were originally developed as part of the Linked Environment for Atmospheric Modeling (LEAD) project. The OGCE removes LEAD-specific features and streamlines the packaging. The workflow suite is currently being used by BioVLAB (a bioinformatics portal) and GridChem (a computational chemistry gateway).

3. Gadgets and Open Social Containers

Google gadgets are becoming increasingly popular way for delivering customized Web content through components. The design concepts of portlets and gadgets are very different. Portlet frameworks achieve web content aggregation on the server side, while gadget frameworks enable client-side aggregation following the Web 2.0 paradigm. A gadget can be regarded as a miniature web application and can define its content and control logic in client-side JavaScript and HTML. In this way it has less dependence on its container than a portlet.

A standardized gadget framework is necessary for development and deployment of gadgets. The Google-led OpenSocial [2] consortium defines a framework that standardizes the practices of gadget and social-networking sites, enabling web developers to write gadgets that can run in any OpenSocial compliant container. Many social networking sites have adopted the OpenSocial framework and open their containers to developers. For example, an OpenSocial gadget can be easily added into the iGoogle sandbox [3]. Moreover, the Apache Shindig [4] incubator project provides a reference implementation of OpenSocial container for PHP and Java.

The OGCE has undertaken a pilot project to test the feasibility of using the Open Social framework for science gateways. We have developed a set of gadgets for both Open Life Science Gateway (OLSG) [5] and the Social Informatics Data Grid (SIDGrid) [6]. OLSG is a computational portal that integrates a group of bioinformatics applications and data collections. SIDGrid provides cyber-infrastructure to help social and behavioral scientists collect and annotate data, collaborate and share data, and analyze and mine large data repositories.

The OLSG includes three gadgets: *ClustalW, BLAST,* and *JobHistory*, which can be loaded in iGoogle or any compatible container. Both BLAST and ClustalW are very commonly used sequence alignment tools in bioinformatics. Through these two gadgets, users can post DNA or protein sequences and run the OLSG's alignment services. The *JobHistory* gadget allows users to check the status of their computing tasks and retrieve the result reports from the finished tasks.

Open Social gadgets are associated with OAuth [7], an emerging standard for Web security. To support OAuth in our gadgets, we have developed an OAuth provider that consists of a group of Java Servlets. By using OAuth tokens, an OpenSocial container and our science gateway build up their mutual trust so that the science gateway can authorize requests from the container to access the restricted data and services.

Based on our initial experiments, we conclude that Open Social and related standards like OAuth are a suitable platform for building science gateways. Code originally developed for the OLSG and SidGrid have been contributed back to the OGCE, and we are examining ways to generalize these contributions for new gateways.

4. OGCE Axis Services

A science gateway must go beyond user interface components and programming layers to also provide portal services that maintain and deliver persistence information. The OGCE Axis Services component is a self-contained build that currently includes the Resource Discovery and Resource Prediction Services, described below. The packaging is general, however, and can be extended to include other Axis 2 services. The OGCE's Grid Portal Information Repository service is currently being repackaged to be included in this release bundle. Web clients to these services are available from the OGCE's GTLAB sub-project.

Resource Prediction Service: Large, complex scientific workflows characteristically have parallel computations, distributed data transfer and management, and soft real-time constraints. However, the computational grids on which these workflows may run have disparate and heterogeneous systems and software and few guarantees of performance and reliability. This necessitates new approaches to scheduling complex scientific workflows on computational grids. The Resource Prediction Service determines the optimal resources for running a scientific

application on a computational grid like the TeraGrid. An optimal resource is one that minimizes the total expected time to completion of the application, which is the sum of the data transfer time, the batch-queue wait time and the compute time as described below.

Data transfer time: Workflow execution usually involves moving large amounts of data between workflow steps, which takes a significant amount of time [8]. Hence, we consider the data-transfer time while calculating the total expected time to completion for an application.

Compute time: Compute time can be obtained from the performance models of the application, which could be as simple as estimates of execution times of the application on different resources. Since grid resources are heterogeneous, the same application may have significantly different execution times on different resources.

Batch-queue wait time: Batch-queue systems like PBS and LSF are used to submit jobs to most grid resources. An application may have to wait for a significant amount of time on a queue before it starts executing. We incorporate the batch-queue prediction service [9] for estimates of batch-queue wait times.

Resource Discovery Service: The TeraGrid Resource Discovery Service is a tool designed to provide user-specific compute, storage and visualization listings to science gateways. It was created to fill the need of many gateways that were otherwise unable to quickly discover which resources a user could access. The service essentially aggregates two distinct functions: discovery and monitoring. It first looks up the resources to which the user has access in the TeraGrid Central Database and then prunes the list of unavailable resources found by querying Inca (http://inca.sdsc.edu/).

For instances where the service is not used within the context of the TeraGrid, configuration options are available to use either GPIR (http://www.gridport.net/services/gpir/) or a static listing as the discovery mechanism. Further, in the event Inca is not available, the service bypasses the pruning stage, returning the full list of user-accessible resources. The Resource Discovery Service is currently in production and used by the File Manager within the TeraGrid User Portal (http://portal.teragrid.org). The service works out of the box with the OGCE file manager portlet.

5. Cyberaide JavaScript Grid Clients

In previous OGCE and related work, we developed programming abstractions for the Grid within the Java CoG Kit [10]. Given the popularity of Perl, PHP- and Ruby-based Web frameworks, Open Social, Facebook, client-side mash-ups, and JavaScript improvements generally, we saw the opportunity to provide client libraries to Grid and related tools using JavaScript. This is the Cyberaide component of OGCE.

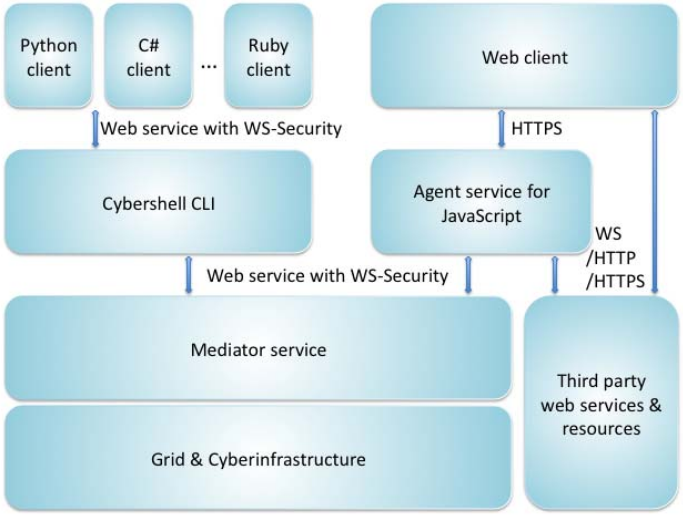

Figure 1 Cyberaide's layered architecture.

Design: We have designed our framework as a layered architecture (Figure 1). The architecture contains a Web client providing access to basic and advanced Grid functionality, as well as the necessary components to deploy them in a secure Web server. To keep the footprint of the library small, most of the functionalities are executed on a secure *mediator service*, described below.

The Web client provides high-level application programming interfaces (API) to the Grid, which include important functionalities such as authentication, file transfer, job management, and workflow management.

Based on the API a portal to Teragrid is constructed providing components to access simple Grid functionalities through graphical user interfaces from inside a web browser. Grid developers can utilize the API to build domain-specific Grid portals or scientific gateways.

We are conducting all interactions to the backend Grid or cyberinfrastructure through an *agent service* and the *mediator service*. The *mediator service* is responsible for communicating with Grid services through underlying Grid middleware such as the Java CoG Kit, Globus toolkit, and even SSH. The *agent service* is an intermediate component that handles all communications between the web client and the mediator service. The agent service acts both as a service provider to the web client and as a service consumer of the mediator service. The agent service forwards the action requests to the mediator service, which ultimately sends requests to the Grid services. The results come back from the Grid through the mediator to the agent, which in turn forwards the information back to the client.

The reason why we need an agent service is that, first we have to host the JavaScript files, CSS files and other resources in some web container, and according to the same-origin policy for JavaScript we need to put the web service that the JavaScript calls in the same web container. Second, by separating the real application logic and putting it into the mediator service, it is possible to host the services on different machines, which increases security and scalability. Finally, this simplifies mashing-up with other third party web services and web resources.

6. Further Information

The OGCE website is http://www.collab-ogce.org. News announcements are available from the OGCE RSS/Atom feed, http://collab-ogce.blogspot.com/feeds/posts/default. The SourceForge project page is http://sourceforge.net/projects/ogce and the SVN code repository can be viewed at http://ogce.svn.sourceforge.net/viewvc/ogce/. Releasable versions of the code are tagged with version numbers and form an SVN branch. Packaged versions of tagged major components are downloadable directly from SVN. We schedule two major releases per year, to coincide with TeraGrid (June) and Supercomputing (November) conferences.

7. References

[1] Jay Alameda, Marcus Christie, Geoffrey Fox, Joe Futrelle, Dennis Gannon, Mihael Hategan, Gopi Kandaswamy, Gregor von Laszewski, Mehmet A. Nacar, Marlon E. Pierce, Eric Roberts, Charles Severance, Mary Thomas: The Open Grid Computing Environments collaboration: portlets and services for science gateways. Concurrency and Computation: Practice and Experience 19(6): 921-942 (2007)

[2] OpenSocial Specification, http://www.opensocial.org/

[3] iGoogle sandbox, http://code.google.com/apis/igoogle/docs/igoogledevguide.html

[4] Shindig, http://incubator.apache.org/shindig/

[5] Wenjun Wu, Rob Edwards et al, TeraGrid Open Life Science Gateway, TeraGrid 2008 conference, http://www.teragrid.org/events/teragrid08/Papers/papers/24.pdf , June 9-13, 2008, Las Vegas.

[6] Social Informatics Data Grid, Bennett Bertenthal, Robert Grossman, David Hanley, et al, E-Social Science 2007 Conference, October 7-9, 2007, Ann Arbor, Michigan

[7] OAuth, http://oauth.net/core/1.0

[8] G. E. Fagg, E. Gaberiel, G, Bosilca, T. Spring, Z. Angskun, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, vol. 15(5-6), pp. 757-768, 1999

[9] J. Brevik, D. Nurmi, and R. Wolski, "Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines," *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principle and Practice of Parallel Programming*, 2006

[10] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," Concurrency and Computation: Practice and Experience, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.mcs.anl.gov/<gregor/papers/vonLaszewski--cog-cpe-final.pdf