Experiences with OGSA-DAI: Portlet Access and Benchmark

Deepti Kodeboyina and Beth Plale
Computer Science Department
Indiana University

Abstract

Portals have proven to be useful client-side applications for providing user-oriented services for accessing the grid. Grids are increasingly being used for collaborative work within the scientific community. The job processing time for high performance computations can be reduced by the usage of computational grids, with its wide availability to resources. Grid users would similarly benefit from having access to databases, mainly, those involved in collaborative data analysis of large datasets and those requiring sharing of data. OGSA-DAI provides an extension to the OGSA framework by allowing access to and integration of data held in heterogeneous data resources. In this paper, we describe our experiences in designing and building a portlet to OGSA-DAI and in testing the grid services access to a relational database by means of a synthetic database workload.

1. Introduction

Grid computing addresses the issue of collaboration, data and resource sharing. Grid services are the middleware that supports the sharing and allocation of these resources. Portals are being developed to access grid services on behalf of users. Grid portals exist for job-submission, resource allocation and management, and file management. Computational grids have greatly influenced the development of science portals that are designed to provide researchers with access to distributed resources. But a large portion of the scientific community requires the analysis of large datasets for solving many interesting scientific problems. Presently, the databases used by the grid users are varied and the sharing of data between them enforces the need for a service that allows for working with the data collaboratively. For instance, researchers may reap benefits of already existing data instead of delaying their research for want of experimental data if that data is shared between those within the same community. Large-scale data analysis done with distributed resources also requires access to a database.

OGSA-DAI (Open Grid Services Architecture – Data Access and Integration) can be visualized as a set of co-operating grid services that enable databases to be accessible through a grid service interface. The OGSA-DAI implementation has been designed with the aim of allowing external data resources, such as databases, to be incorporated within the OGSA framework and hence assessable via a standard grid services interface. By using OGSA-DAI, heterogeneous disparate resources can be accessed uniformly. It includes support for the registration and discovery of databases and for interaction with those databases. The structure of the results retumed and the method and location of their delivery can be set by the client.

We undertook the building of a portlet for OGSA-DAI for the purpose of providing an interface to allow grid users to share information from various resources and also interact with the database. The portlet we developed functions essentially as a client service that facilitates user interface with databases via web browser. In the

longer term, we envision the portlet as a means for a user to control and monitor a workflow script that as part of its functioning extracts metadata from a database. This paper discusses our experiences with the development of a portlet for OGSA-DAI and on the application of a synthetic database benchmark [2] to a relational database accessible through a grid services interface.

The remainder of the paper elaborates on our experiences. Section 2 provides details regarding the implementation, working of the portlet and interaction of OGSA-DAI with the database. Section 3 describes the application of the benchmark against OGSA-DAI by way of graphed results and the alterations that were made to the benchmark. Section 4 details ongoing and future work.

2. Portal to OGSA-DAI

Portals provide gateways to web services in that they provide user level access and control to services through web browser interfaces. At Indiana University and elsewhere, there is work going on to develop portal access to grid services. The portlet we describe here is part of the Alliance Portal[11]. Its purpose is to provide web browser access to the OGSA-DAI implementation of the emerging GGF DAIS standard[3] for grid service access to databases and other external data resources.

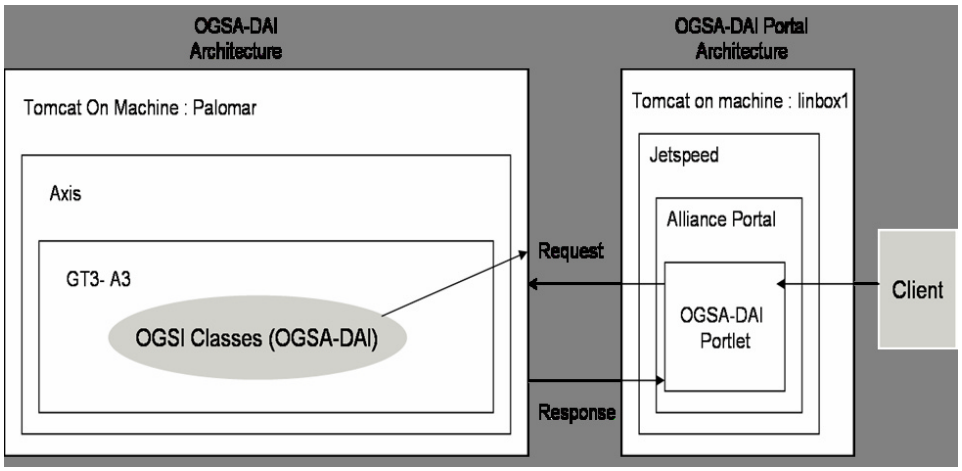

Figure 1. Architecture of Portal and relationship to OGSA-DAI

The architectural components of the Alliance portal and OGSA-DAI portlet, shown in Figure 1, are:

- A servlet container, Tomcat, that serves as a web service hosting container for both OGSA-DAI and the Alliance portal.
- A portal implementation, Jetspeed, that provides an API for developing portlets.
- The Alliance Portal in which the OGSA-DAI portlet resides.
- An implementation of the SOAP protocol, Axis, for messaging.
- Globus Toolkit, GT3-A3 – an alpha version of a reference implementation of the OGSA infrastructure as described in the Open Grid Service specification. The toolkit provides a grid service container that runs within a standard web service container.
- OGSA-DAI - grid-enabled middleware reference implementation enabling access and control to external data resources.

Portal interaction with the database through OGSA-DAI is illustrated in Figure 2. The portlet is provided with the Grid Service Handle (GSH) of the Grid Service Registry (GSR). This is done out of band. The portlet queries the registry to obtain the grid service handle to the factory, and the accompanying service document that describes the grid data service instance that has already been created. From a prior screen, not shown, the user has browsed the local files system to obtain a perform document that describes the query the user wishes to execute. That perform document is shown at the top of the portlet.

The user issues a query by selecting 'Query/Update Database' at the bottom of the page. Shown in the portlet are the results of having executed a query. Shown to the left is the response document in its XML form. On the right is the status of the execution. Partially shown to the bottom right are the results in table format. The response document has been converted to a table format using XSLT before being displayed to the user.

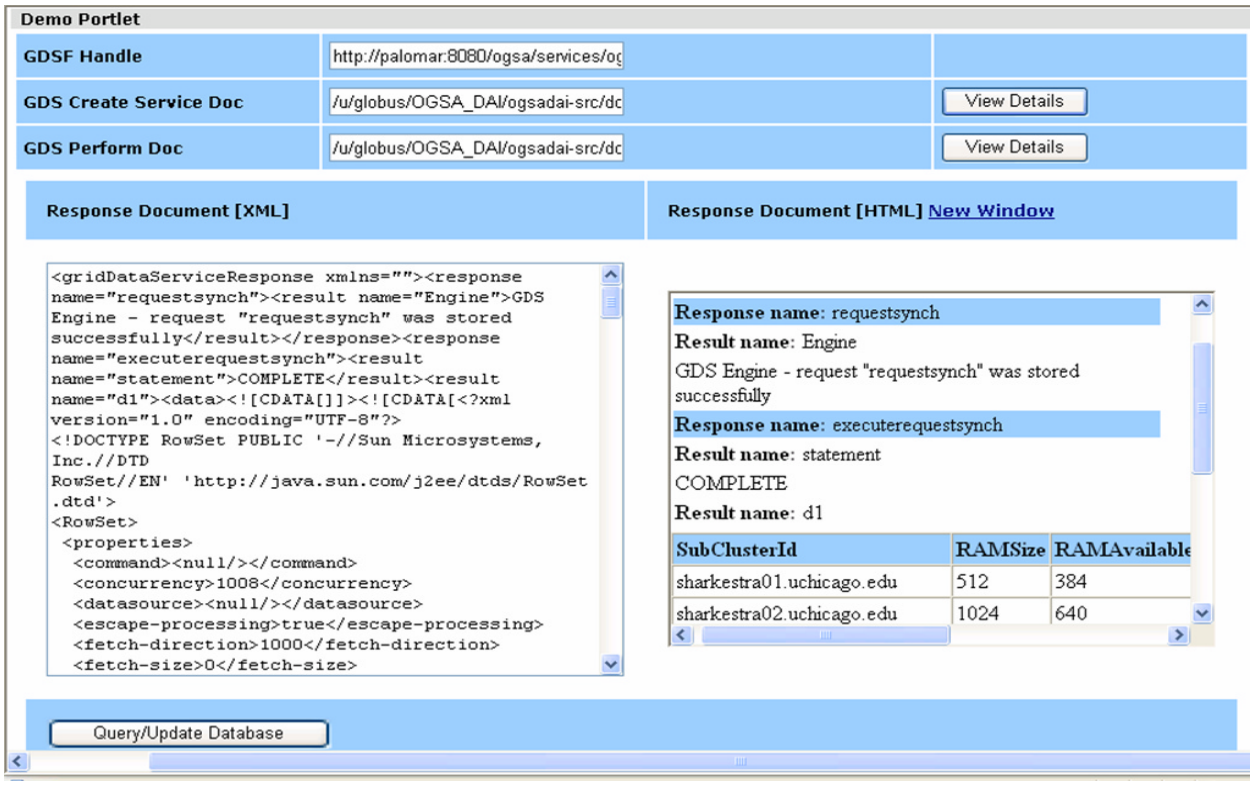

Figure 2. Portal Access to OGSA-DAI

2.2 Discussion of Experience

An obvious issue in development of the portlet is the creation, storage, and transmission of the perform documents. The Grid Data Service runs as user "globus" on a remote server machine (see Figure 1). The portlet must create and manage perform documents while providing document transparency to the user. That is, the user should have the option of being unaware of the fact that communication is through perform documents.

In our first version of the portlet this transparency is not provided. The user explicitly selects a perform document to execute by browsing his/her directory space. Version 2 of the portlet will replace this mechanism with two other access interface mechanisms. First we will use WebDAV to provide less restricted access than the browse feature that now requires that documents be available on local disk or a mounted file system. Second, we will allow the user to enter required information and options in a web form then generate an XML perform document.

The representation of the results in the form of an XML document, that is, the "response document" is not particularly user friendly for relational database users. Thus we transform the results by means of the XSLT stylesheet into a form suitable for human consumption, that is, in a table format of rows and columns. Processing the stylesheet requires the use of an XSLT processor and an XML parser, both of which are available in Jetspeed. XML documents retrieved from an XML-native database such as Xindice are presently left in XML form.

OGSA_DAI also requires the globus certificate of the user to authorize his interaction with the database. The database role-map file has been provided for the users to include their certificates. But, handling of security has to be done by the portlet.

OGSA-DAI provides the capability to include multiple queries/updates within a single perform document. In the case where we need to access to more than one database within a database management system, the possibility of grouping queries into a single perform document is impossible. For instance, the IU RGRbench benchmark queries, discussed below, issue a query repeatedly against the "grid" database. When finished, the average query response time for query *X* is computed and that result stored to the "gridHistory" database. These accesses cannot be done within a single perform document. The reason for this is that a Grid Data Service when created is assigned to a single data resource and a data resource is equated to a database in relational database management system.

Our experience with OGSA-DAI included executing the IU RGRbench benchmark on a relational database that is accessed through a grid services interface, namely GT3alpha and OGSA-DAI v2.0. This is discussed in the next section.

3. Understanding Performance

We undertook a performance assessment as a means to better understand the overheads associated with a grid services interface to a database. Having recently completed the application of the IU RGRbench benchmark/workload to relational, XML, and LDAP platforms (i.e., mySQL 4.0, Xindice 1.1, and MDS2 respectively), we undertook its application to mySQL 4.0 as accessed through the grid services database access implementation of OGSA-DAI 2.0.

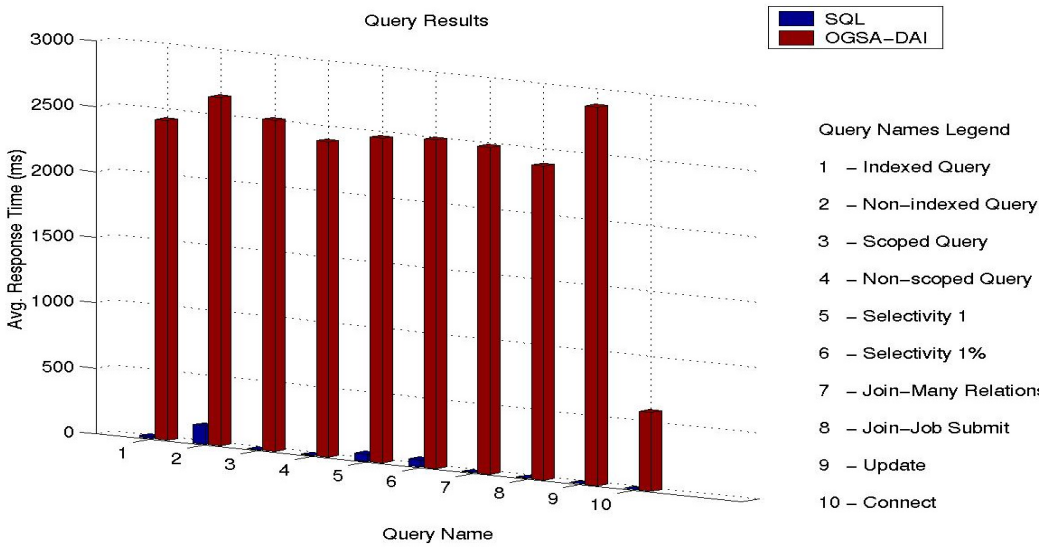

Fig. 3 RGRbench for queries having synchronous delivery

The IU RGRbench benchmark [2] is a set of queries and scenarios defined over a data model of grid resources, such as hosts, clusters, end-to-end connections, users, jobs, and file servers. The data model extends the GLUE schema [6] proposed by the GGF (Global Grid Forum) [5]. The benchmark includes scripts for creation of schemas for three databases: mySQL, Xindice, and MDS2, and for population of those schemas with approximately 80,000 objects (or tuples or documents depending on the terminology one uses.) The queries of the benchmark test a broad range of database functionality through means of realistic grid specific questions. For instance, the job submission query looks for a host having a specific configuration of a particular software

environment exists and a user has an active account. The scenarios are synthetic workloads that run for an interval of time on the order of 10-20 minutes and measure query response time for concurrent queries in the presence of update loads. The benchmark comes with implementations of the query set for SQL, XPath, and LDAP.

The RGRbench benchmark is run against the grid services implementation. Each query script, 13 in all, issues 1000 requests to the database sequentially. There are no other users active. The scripts are submitted by an independent Java client and not through the portal. The query response time is calculated as the average of the runs. Grid Data Service creation time is not included. That is, for purposes of the benchmark, it is assumed that the GDS already exists before testing begins. The data size of the result set varies across the queries from a few bytes to 45KB. Requests having a small result set were issued as synchronous requests. These are shown in Figure 3. Those having a large result set used alternate asynchronous mechanisms as described below. These are shown in Figure 4. The response times for these varied from 3-5 minutes as can be noted from the figure. As part of this effort, we developed a set of perform documents for the benchmark. In the essence of space constraints, results from the "scenarios" are not presented. In Figures 3 and 4, the query names are given to the right and mapped to an integer that appears on the X-axis. Details of the benchmark can be found in [2]; a summary of the queries is as follows:

Indexed (Indexed and NonIndexed) – indexes are well known to enhance retrieval time. This pair of queries tests the index capability by asking similar questions on an indexed attribute and a non-indexed attribute respectively.

Scoped (Scoped, NonScoped, ScopedHost, NonScopedHost) – a scope defines a starting point in a hierarchical search tree. Scoped queries should outperform their non-scoped counterparts in hierarchically oriented databases such as LDAP and Xindice.

Selectivity (Selectivity1, Selectivity1%, Selectivity10%) – the selectivity of a query is the number of objects returned. This query set examines return sets of 1 object, 1% of objects, and 10% of objects.

Join (ManyRelation, JobSubmit) – joins occur when a user requests information that resides in more than one table or collection. These two queries differ in the types of questions asked.

Other (Connect, Update) – "connect" measures the time to connect to a database; no query asked. "Update" measures the time to update a single attribute in an object, row, or document.

It should be noted that the results shown here are gathered against OGSA-DAI version 2.0. Since the release of version 2.0, the OGSA-DAI team has made significant progress in reducing overhead. That being said, Figures 3 and 4 depict the overhead of accessing mySQL4.0 through a grid services interface (GT3.0alpha and OGSA-DAI 2.0). The notably longer query response times for ScopedHost, NonScopedHost, and Selectivty10% are due to the large (> 10KB) result set returned. For the three, we used the "deliverToURL" delivery option because inherent limits in SOAP precluded a synchronous return of a large result set and OGSA-DAI 2.0 predates the option to compress the results. The latter was added with OGSA-DAI 2.4.

4. Summary and Future Work

Developing grid service access to databases is an important step in developing Grid infrastructure that adds value to the day-to-day operations of its users. For instance, if the Grid infrastructure can enable access to scientific databases heretofore inaccessible to remotely distributed scientists, the infrastructure will have proved its worth. We have described in this paper our experiences in creating a portlet for OGSA-DAI and also in testing the grid services interface to a relational database by means of a synthetic database/workload we developed to better understand grid resource repositories (repositories of information about hosts, clusters, services, etc.)

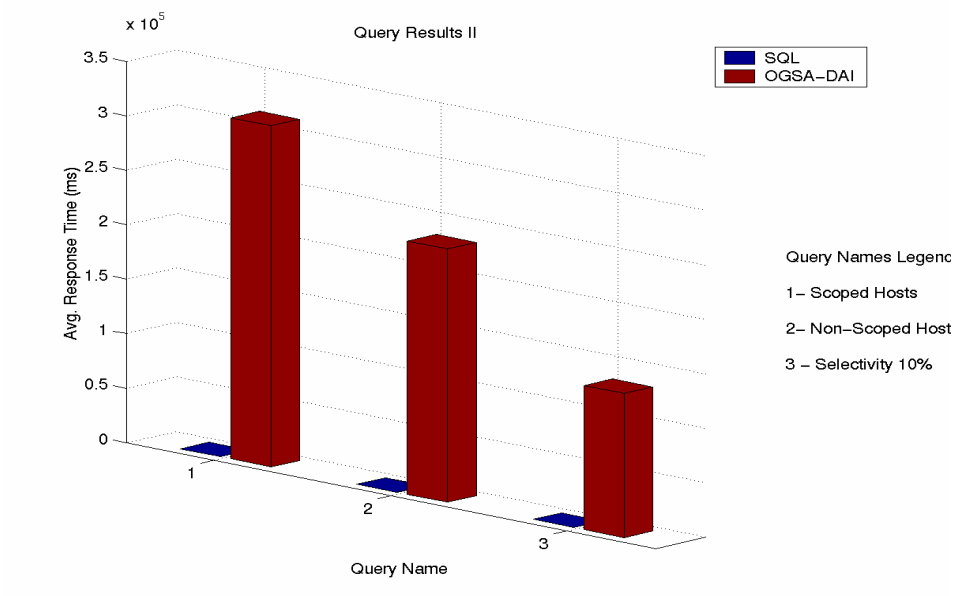

Fig. 4 RGRbench for queries having asynchronous delivery

The perform documents developed as part of the assessment described herein will be made available with the release of the IU RGRbench expected mid-October 2003. Most importantly in our ongoing work is to evaluate the benchmark against the latest release of OGSA-DAI, version 3.0. We are also refining the query response time measurements in order to understand the latencies at a finer granularity.

Issues regarding authentication and access control, dynamic creation of perform documents, refining the portal interface. Dynamic creation of a Grid Data Service in response to multiple users is also being explored. In the absence of a factory that can create the requested data resource, the portlet will have to administer the creation of a factory tailored to provide access to the requested resources. This future work would further enhance the portlet for broader usage.

5. References

[1] OGSA-DAI Open Grid Services Architecture Data-Access and Integration http://ogsadai.org.uk

[2] Beth Plale, Craig Jacobs, Ying Liu, Charlie Moad, Rupali Parab, and Prajakta Vaidya. Understanding Grid Resource Information Management through a Synthetic Database Benchmark/Workload: Benchmark Details. *Indiana University Technical Report TR583*, 2003.

[3] GGF DAIS WG, Grid Data Service Specification, July 2003 http://www.cs.man.ac.uk/grid-db/documents.html

[4] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D.Snelling, P.Vanderbilt. Open Grid Services Infrastructure (OGSI). *Global Grid Forum Draft Recommendation*, 6/27/2003 http://www.isi.edu/~karlcz/papers/draft-ggf-ogsi-gridservice-29_2003-04-05.pdf

[5] Global Grid Forum. http://gridforum.org 2003

[6] DataTAG. Glue schema: common conceptual model for grid resources monitoring and discovery, 2003 http://www.cnaf.infn.it/sergio/datatag/glue

[7] I. Foster, C. Kesselman, J. Nick, S. Tuecke. Grid Services for Distributed System Integration. *Computer*, 35(6), 2002.

[8] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets, *Journal of Network and Computer Applications*, 23:187-200, 2001.

[9] S. Krishnan, R. Bramley, D. Gannon, M. Govindaraju, R. Indurkar, A. Slominski, B. Temko, R. Alkire, T. Drews, E. Webb, and J. Alameda, "The XCAT Science Portal," *Proceedings of Supercomputing*, 2001.

[10] Apache. Axis - http://apache.axis.org

[11] Alliance Portal – http://www.extreme.indiana.edu/alliance