

Design and Implementation of OGSA-WebDB

– A service-based system for making existing web databases grid-ready –

Isao Kojima and Said Mirza Pahlevi

Grid Technology Research Center(GTRC),

National Institute of Advanced Industrial Science and Technology(AIST), Japan

E-mail:{kojima.mirza}@ni.aist.go.jp

Abstract

This paper describes the design and implementation of a service-based system called OGSA-WebDB. OGSA-WebDB is a grid database interface which enables grid application to query existing web databases using SQL. The system is composed with proxy database, mediator and wrappers. The proxy database delegates external web databases and the mediator converts SQL query into search conditions to the web databases. Result data is converted by wrappers and is stored into the proxy database. We propose the simple two phase processing scheme to support SQL query for web databases. The mediator also uses simple dynamic query processing scheme to join multiple web databases. Performance evaluation based on the response time shows the efficiency of the method. The system is fully implemented on the top of the Globus Toolkit and OGSA-DAI.

1. Introduction

Distributed and heterogeneous database integration based on grid technology [1,2] is an emerging research topic, especially for scientific databases for the life sciences, or materials and geo-spatial databases. For instance, distributed data mining to explore new scientific knowledge should be possible in a grid environment that integrates many large-scale scientific databases[21].

There is currently an enormous number of databases on the web (web databases), and the problem is how to interface these existing web database resources easily in the grid environment. For example, our survey[4] found over 100 scientific web databases in our institute and over 1,000 in the area surrounding Tsukuba City, Japan. If we could interface these with a grid environment, it would be possible to accelerate research activity.

In order to achieve scientific database integration, we have developed a system called OGSA-WebDB[3], which integrates existing web databases in a grid environme

This paper describes the design and implementation of the system.

Section 2. describes the architecture of OGSA-WebDB. Section 3 discusses the problems and solutions of the system. Related work is reviewed in Section.4 and Section 4 presents the experiment results and implementation issues. The final section gives some conclusions.

2. OGSA-WebDB¹

2.1 Overview

The purpose of OGSA-WebDB is to provide a grid service interface for existing web databases. In principle, each database query issued within the grid is converted into site-specific queries for web databases. Fig. 1 shows the architecture of the system. OGSA-WebDB consists of the following components.

1) Proxy Database: This database delegates external web databases within the grid. Any

¹ We used the word "OGSA" since the system is developed on top of GT3.0(OGSI) and OGSA-DAI/OGSA-DQP.

grid application will access the proxy database instead of the original web databases. In OGSA-WebDB, we use a relational database system as the proxy database, so that each web database is represented as a relational table. Our approach is based on a virtual approach that converts an SQL query into a form query in real time; we manage the materialized proxy tables explicitly. This table can be used as a cache.

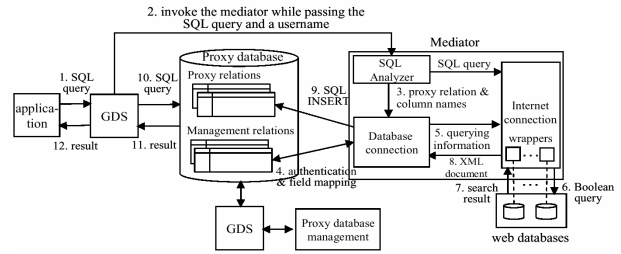


Fig. 2. OGSA-WebDB Process Flow

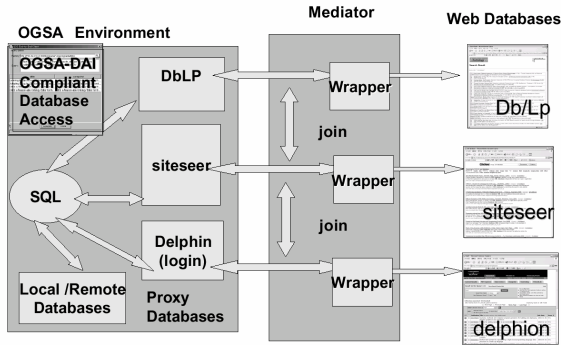


Fig. 1. OGSA-WebDB Architecture

2) Mediator: The mediator bridges the gap between the proxy database and the external web databases. It accepts SQL queries (sent by a grid application) and transforms them into one or more Boolean search conditions sent to the target web databases. The mediator also controls parallel access and join operations over multiple web databases.

3) Wrappers: Separate wrappers are installed for each web database. After being invoked by the mediator, they construct and send an HTTP request to the appropriate web databases. They receive HTML documents from the web databases and convert them into a set of relational tuples to be inserted. These wrapper programs are stored using the proxy database and are managed by the database management service.

2.2 Processing Architecture

The detailed systematic procedure for processing an SQL query is shown in Fig. 2.

The grid application can interact the Grid Data Service(GDS) of OGSA-DAI. Setup file(ActivityMap) is modified so that every SQL(step1) to the GDS is redirected to call the mediator(step2). The *SQL analyzer* within the mediator first make own *database connection*(step3) and retrieve metadata information from the management relations(step4).

This *management relations* is a metadata relations which includes following information.

- 1) *Resource*: wrapper classes and names, URLs for web databases.
- 2) *Authentication* : authentication information for the web databases
- 3) *Mapping*: attribute mapping between local tables and web database fields.
- 4) *User*: user accounts for the proxy database.

The *database connection module* send the query information to the *Internet connection module* which extract search conditions for the web databases(step5). Appropriate wrapper is selected and invoked from this module(step6). The wrapper backs the result data in a form of XML(step8) and the *database connection module* stores the results into the proxy tables(step9).

Finally after all proxy relations are filled with the data from web databases, the control is back to GDS of OGSA-DAI and the original SQL query is performed on the proxy relations(step10). We call this the *two-path processing scheme* since each SQL is performed twice(Mediator and OGSA-DAI).

The main feature of this architecture is that this scheme can be implemented on top of existing grid middleware like OGSA-DAI[14] so that there is no need to modify OGSA-DAI program codes.

3. Problems and Solutions in OGSA-WebDB Architecture

This architecture must deal with the following three problems.

1) SQL Conversion to form a Query:

We assume that each web database supports Boolean conjunctions of keyword and field search conditions. Since most web databases are based on keyword searches, it is difficult to make an exact conversion from SQL to search conditions, such as comparisons.

2) Query Scheduling and Optimization:

An SQL query that includes join operations needs to merge datasets from several web databases. However, an SQL query can include tables without any select conditions. Some query scheduling method is needed.

3) Wrapper Construction:

Since each web database has a different interface, development of the wrapper is an indispensable task. It is very important to provide an easy method to develop the wrapper program.

OGSA-WebDB takes the following approach to solve these problems

1) Approximate Conversion based on Two-path Processing Scheme:

We extract only keyword-based conditions (character matches) from the SQL statement. This is an approximate conversion that will retrieve unnecessary answers from the web. Therefore, we use the two-path processing which is described before. The second SQL processing extracts exact answer since it is performed on the proxy database.

This process includes a procedural overhead since every SQL is processed twice.

However, this overhead is small, as shown in the experiment.

2) Dynamic Query Scheduling and Optimization:

An SQL query that includes join operations needs to merge datasets from several web databases. However, an SQL query can include tables without any select conditions. Some query scheduling method is needed. We assumed that the total operational cost depends on the response time of the web databases and is not a factor of how many results are returned from the web(We will show it in the experiment).

Also, it is very important that the response time of each webdatabases varies depend on the status of the internet. Dynamic processing is necessary.

Therefore, a simple semi-join algorithm over multiple web databases is implemented as follows:

- 1) Perform a conditional search in parallel.
- 2) For each join operation, construct a new search condition using the result of the step1.
- 3) Perform semi-joins in parallel.
- 4) In the step2 and 3, the first-come first-serve algorithm is used to make dynamic scheduling.
- 5) If all relations are queried, merge all the answers.

Figure 3 shows a snapshot of a running grid client. Note that ChemFinder requires a login before sending the query. The SQL is as follows.

```
SELECT distinct m.brand,c.formula,  
c.cas,m.manufacture,f.strength  
FROM fda as f, chemfinder as c, mydruglist  
as m  
WHERE f.name=m.generic  
and c.name=m.generic  
and f.active like "%pril%".
```

In this example, the client sends an SQL query that joins data from two web databases (Drugs@FDA[6] and ChemFinder[5]) and a local mydruglist relation.

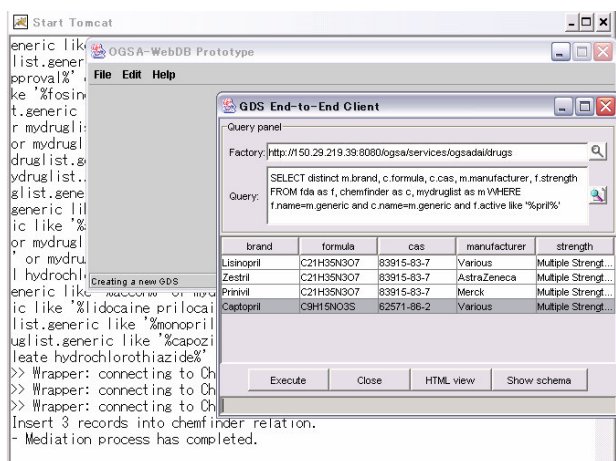


Fig. 3. Screenshot of the SQL Client

As described in (1), the *like* clause “%pril%” is converted into a keyword search and sent to the ChemFinder web. The answer records from ChemFinder are used for semi-joins with FDA, and the attribute values of *name* are converted into another search condition on the FDA web. This is performed in parallel. Finally, the answer from FDA is stored in a proxy relation and the entire SQL, including joins with a local table, is performed again.

3) Wrapper Construction:

Since each web database has a different interface, development of the wrapper is an indispensable task. It is very important to provide an easy method to develop the wrapper program. OGSA-WebDB takes the following approach.

1. Combined general-purpose wrapper development system: We do not provide a specific wrapper generator tool, but interface with public/commercial wrapper generator tools, such as Republica’s XFetch Wrapper[7] and Compaq’s Web language WebL [8]. Any tool that produces a Java jar file can be integrated with the system.

2. Provide a management service: The interface with the wrapper program is also managed using proxy databases. For instance, mapping between the attributes of proxy tables and the search fields of web

databases is also stored as a mapping relation. To facilitate wrapper development, this database provides a management service that is also a grid service. To add new web databases, the user enters the appropriate information and uploads the wrapper jar file. This database and the service are shared, so that users can share these wrappers and web database definitions.

Figure 4 shows a snapshot of the proxy database management tool that installs a new wrapped web database into the system. This tool also helps an administrator to install OGSA-WebDB in a grid environment by updating the grid configuration files. It is implemented as a grid client so that it can work in a shared grid environment.

4. Related Work

The web/database community has long studied data integration and data access across multiple/heterogeneous data sources[9-11]. Some methods, such as the IBM DB2 Information Integrator[12], can be used as grid/web database mediators. OGSA-WebDB has the following properties:

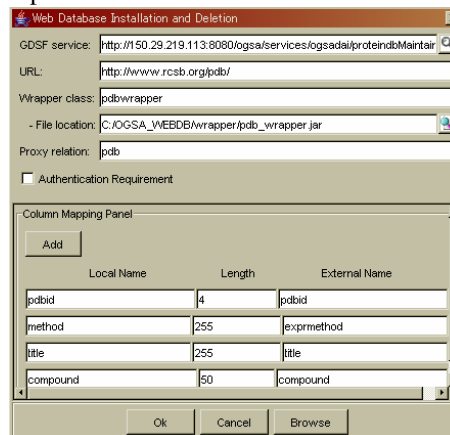


Fig. 4. Management Service Client

1) *Service-based architecture:* All the functions, such as SQL client and management functions, are provided with

a grid service-based architecture.

- 2) *Focus on accessing Web databases*: Our architecture has a proxy database that fills the gap between SQL query language and form queries.
- 3) *Platform independence*: OGSA-WebDB can support any SQL server that OGSA-DAI supports.
- 4) *Support of grid sign-on (GSI)*: GSI is mapped to database accounts and web database accounts.

5. Implementation and Performance

5.1 Implementation

We have fully implemented OGSA-WebDB using Java. Currently, the system runs on top of Globus Toolkit 3[13] and OGSA-DAI 3.02[14]. Although it would be easy to implement using XML databases for query and data conversion, we adopted the OGSA-DAI relational version for stability and extensibility(especially for interoperability with OGSA-DQP[20])

Activity map file entry is currently customized only for the select statement since most web databases support only retrievals.

The proxy databases are stored in an open source MySQL[15] database. Ten sample wrappers have been written, using WebL and XFetch to provide access to several scientific, drug, patent, and bibliography web databases. In the current implementation, each database category has a separate grid service factory, so that it requires a distributed processing service (like OGSA-DQP) to integrate the database service under different factories.

5.2 Performance Evaluation

In order to examine the performance of OGSA-WebDB, we set following measurement criteria.

- 1) *Overhead of the two-path processing scheme*.

Our query processing includes additional SQL processing on the proxy database.

Therefore, we measured the overhead of the 2nd path within the total response time.

- 2) *Performance of simple dynamic parallel processing*. In order to show the effectiveness, we compared our parallel processing scheme with the serial one.

	webdb retrieval (msecs)	mediator (msecs)	ratio(%)
DBLP	4738.8	32	0.675276441
CSB	5530	24	0.433996383
CSB	11296.2	20	0.17705069
Chemfinder	5077.4	20	0.393902391
FDA	2363.2	20.2	0.854773189
PDB	6429.4	34	0.52882073

Table 1. Query Processing Time for various Web Databases

	Parallel Execution (msecs)	Serial Execution (msecs)	ratio
Q1 (2tables)	23149.8	53803.6	2.324149669
Q2-a	10667.2	31170.8	2.922116394
Q2-b	31511.8	78002.4	2.475339397
Q3 (3tables)	48399.8	157460.8	3.253335758

Table 2. Serial and Parallel Processing Time

Table 1. and Table 2. show the query processing time for various web databases. All measurement is based on the response time to get the final result. Six web databases were used in the experiment: DBLP[16], the Collection of Computer Science Bibliographies(CSB)[17], CiteSeer[18], ChemFinder, Drugs@FDA, and Protein Data Bank (PDB)[19]. We constructed five SQL queries for each web database and determined the average query processing time.

For Table 2, several join queries are examined. Q1,Q2-a and Q2-b joins 2 tables. Q3 joins 3 tables. Q2-a and Q2-b is the same query for different web databases.

As the result shows,

- 1) The retrieval response time depends heavily on the web database, while the mediator processing time is similar for

all databases. This shows that the mediator processing time is negligible given our simple query execution procedure.

- 2) Parallel processing shows scalability. It is clear that parallel processing can reduce the retrieval time. There is some scalability since the number of tables increases, the ratio also increases.

5. Conclusions and Ongoing Work

This paper examined an OGSA-WebDB system that incorporates existing web databases into an OGSA-based service environment. The software was demonstrated at the SC2003 exhibition and should be released soon. In addition, some experimental query services will be provided on the internet from our website.

Current ongoing work includes:

- 1) Combining with OGSA-DQP to support a distributed environment.
- 2) Devising efficient data-caching and integrity management algorithms for web databases using proxy tables.
- 3) Providing support for the standard web service architecture.
- 4) Providing an efficient database discovery method as a grid database resource information service.

Discussion on Grid Tools:

Based on our development, we have several feedback on following grid tools.

1) *Globus Toolkit:*

In our current implementation, we did not utilize any functions specific to GT3.0/OGSI, such as notification and factory. Also, GSI-enabled version is very slow. Thus, it will be useful to have a subset which supports lightweight web service functions and GSI.

2) *OGSA-DAI:*

As described before, most wrapper returns XML format and it is natural to extend to support XML database. Some relational DBMS product supports XML databases also

with SQL[22]. However, current OGSA-DAI/DAIS specification of SQL and XML database is separated. We think it will be useful if relational and XML implementations are merged into the same single specification.

From this point of view, we are evaluating recently proposed WS-RF[23] specification, especially to have a proper subset of service functions with lightweight implementation.

References

- [1] GGF: <http://www.ggf.org/>
- [2] DAIS-WG: <https://forge.gridforum.org/projects/dais-wg/>
<http://www.cs.man.ac.uk/grid-db/>
- [3] S.Mirza and I.Kojima, OGSA-WebDB, An OGSA-Based System for Bringing Web Databases into the Grid, To appear ITCC'04, 2004.04.
- [4] TsukubaWAN Research Database Portal (prototype) :
<http://www.aist.go.jp/infobase/tsukubawan/>
- [5] ChemFinder,
<http://chemfinder.cambridgesoft.com/>
- [6] Drugs@FDA, <http://www.accessdata.fda.gov/>
- [7] Xfetch Wrapper, <http://www.x-fetch.com/>
- [8] WebL, <http://research.compaq.com/SRC/WebL>
- [9] D.Florsecu et al. Database techniques for the world wide web: A survey. SIGMOD Record,27(3):1998.
- [10] G.Zhou et al. Generating data integratin mediators that use materialization, J. of Intelligent Information Systems. 6(2/3):1996
- [11] Y.Zhuge et al. View maintenance in a warehousing environment Proc.ACM SIGMOD,1995.
- [12] L.Haas et al., Data integration through database federation, IBM System.J. 41(4):2002.
- [13] Globus Toolkit, <http://www.globus.org/>
- [14] OGSA-DAI, <http://www.ogsadai.org.uk/>
- [15] MySQL, <http://www.mysql.com>
- [16] DBLP,
<http://www.informatik.uni-trier.de/ley/db>
- [17] CSB, <http://linwww.ira.uka.de/bibliography>
- [18] Research Index, <http://citeseer.nj.nec.com/cs>
- [19] Protein Data Bank, <http://www.rcsb.org/pdb>
- [20] OGSA-DQP, <http://www.ogsadai.org.uk/docs/dqp>
- [21] Database Grid,
<http://www.gtrc.aist.go.jp/dbgrid/>
- [22] SQL/XML, ISO-IEC 9075-14, 2003
- [23] I.Foster et al. From OGSI to WS-Resource Framework:Refactoring and Evolution, 2004.02, <http://www.globus.org/wsrfs>