Design and Implementation of GT4 based Database Access and Integration Service in Grid Environment

Hyuk-Ho Kim*, Ha-Na Lee*, Pil-Woo, Lee**, Yang-Woo Kim*
*Dept. of Information and Communication Engineering, Dongguk University
**Team of CNI Middleware Research, Korea Institute of Science and Technology Information
e-mail : *{hulegea, hana1007, ywkim}@dongguk.edu, **pwlee@kisti.re.kr

그리드 환경에서 글로버스 툴킷 4 기반 데이터베이스 접근 및 통합 서비스 설계 및 구현

김혁호*, 이하나*, 이필우**, 김양우*
*동국대학교 정보통신공학과
** 한국과학기술연구원 CNI 미들웨어 연구팀

Abstract

Data Grid is a kind of Grid computing provides the cooperative environment through the distributed data sharing, and can manage the massive data easily and efficiently. We designed and implemented Globus Toolkit4 (GT4) based database access and integration service (GDAIS). This service was implemented as Grid service for run on the GT4 which is Grid middleware. And it provides functions which are automatic registration of database in virtual organization, distributed query service, and the unified user interface. Also this system can use components which are provided from GT4. Therefore it can improve the efficiency to distribute and manage databases, can easily access and integrate of the distributed heterogeneous data in Grid environments.

1. Introduction

Grid computing [1, 2] provides a network-based infrastructure integrating geographically dispersed computing resources such as high performance computer, storage/database, high-tech experimental equipment, etc. The most common resource in Grid is storage. Data grid can construct a virtual storage environment for sharing the massive data (file system, storage, etc) in virtual organization. Grid database which is the kind of Data computing, is a database management system for processing and use of the distributed data in Grid environment.

The data in Grid environment has heterogeneous features. For access and integration of the data, it n eeds to the following requirements: 1) it needs to the integration interface for distribution and management for data resources[1]; 2) it needs to the distributed query service which can query to all data resource; 3) also it can integrate the distributed query results; 4) it can show the integrated query results to user as the unified mechanism.

The proposed system, Globus Toolkit4 [3] based data access and integration service (GDAIS), provides:

- provides a heterogeneous data resource distribution and management service,
- provides a distributed query service which can query to all data resource,
- provides a unified user interface which can integrate and show the distributed query result.

The rest of this paper is organized as follows. In Section 2, we briefly explain the related work. Section 3 presents the system requirements and the overall system architecture. Then the implementation of GDAIS is shown in Section 4. Finally, we present our conclusion and future works in Section 5.

2. Related Work

2.1 Grid Database System

The Grid database system plays the vital role in each of resource and the services of the Grid. The Grid provides a platform, which can structure a high performance Grid database management system with a security, reliability and the independent computation ability [4].

The Grid database management system will become an important resource in Grid, providing the data management service. The construction method of Grid database management system, at first, provide a middleware, which pack the database management system as the Grid service [5], in order to make the Grid application to access the Grid database. Secondly, expand the database management system existed; let it use the function provided by the Grid to realize the distributional database and the directly correlative Grid service.

2.2 Distributed Data Integration Strategy

The essential target of the Grid is to support cooperative work on the shared resources and the Grid database

[1] Data resource is components which abstract actual databases (or other data resources or anything really).

integration. It is the hot researching field in the Grid data management at present. At present there are three kinds of Grid databases integration strategies [6].

1) Virtual database. The virtual database is a federal database. It only has a federal pattern, and all users are unable to perceive the fact that many independent databases exist. Therefore, when virtual database constructs, you need to consider transparency.

2) Custom-made integration. This way refers to the application procedures completing the database integration by themselves. The Grid database management system should support this kind of integrated way, to reduce the cost, the time consumption and the wrong occurrence.

3) Increase integration. In the increase integration, the development personnel do not need to complete each detail of the integration. The accessing of the senior data and the integrated module can automatically complete some integrating step of later periods.

2.3 Open Grid Service Architecture - Data Access and Integration

OGSA-DAI [7, 8, 9, 10] is one kind of integrated middleware, which realize the data access and the integration in Grid environment. OGSA-DAI is constructed into a tool box with many expansion points, so as to let the developers expand the ability to meet the specific requirements The middleware of OGSA-DAI permits the same data resources to carry on the synthesis in a compatible OGSA system structure. The Grid service in the OGSA-DAI provides the basic operation. The standard module is used for accessing and controlling data information and the resources. OGSA-DAI also provides expansibility mechanism, thus it can increase more activities which the users define and are executable in the OGSA-DAI. The implementation of OGSA-DAI is shown in Figure 1.

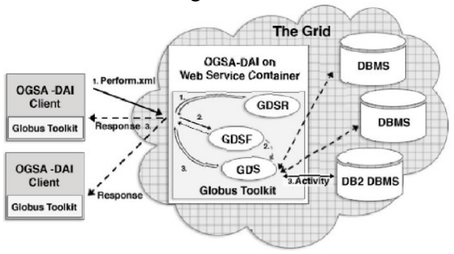


Figure 1. Implementation Process of OGSA-DAI

At present the middleware of OGSA-DAI is in research phase and development continuously. The main flaws of the existing middleware of OGSADAI are: Lacking the database automatic registration service; the capability to tolerate a bad database connection; the data integration of the isomerism database is not good.

3. The System Design

3.1 The System Requirement

In general, many web databases cannot be accessed directly by using local database drivers or be queried using a specific database language, such as SQL. Instead, they must be accessed by using HTTP GET or POST requests via web search interfaces and be queried by using keywords or Boolean conditions. Web database owners usually do not make the metadata of their databases available.

In order to provide the interface for access and integration of data in Grid environment, it has to satisfy the following requirements:

- Query processing using database query language,
- Access to database using the specific driver such as, JDBC and ODBC,
- Monitoring databases in Grid environment,
- Complying with Grid specification, so it can easily be plugged into any Grid environments,
- Support to Grid Security Infrastructure (GSI) for authentication of database.

3.2 System Architecture

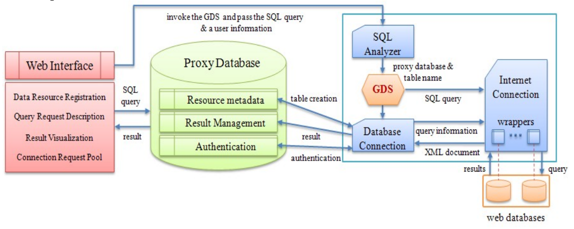


Figure 2. System Architecture

As you can see Figure 2, the overall system is divided into 3 sections. Fist, it is user interface. It provides the function which can conduct the data resource registration. It also can query to all data resource. And the query result can be see web-based user interface. In addition, it provides an efficient connection pool function. It analyzes a query request, and then connects to a suitable data resource. Second, it is proxy database. It manages registered data resources, authentication, and query results. Last, it is the actual query conduction. This portion has SQL analyzer for parsing of SQL which is sent from user interface. Through the parsed result, it can communicate with an applicable Grid database service (GDS). GDS first conducts user authentication process. After user verification, it builds the result table which is for save the result data, and conducts a query. And then it conducts the query result to send the proxy database.

3.2 System Performance Factors and Tools

The factors for the proposed system performance experiment are following:

- SQL Query Parsing. This indicates the total time that it takes to analyze a user SQL query statement in order to conduct on the heterogeneous databases.
- Local Database Driver access. This indicates the time that it takes to access the data resource directly, which is collocated with the client program.
- Remote Database Driver access. This indicates the time that it takes to access the remote data resource, which is collocated with the client program separately.
- Local Database Query Processing. This indicates the time that it takes to process a query on the same machine with a data resource.
- Remote Database Query Processing. This indicates the time that it takes to process a query on the remote machine with a data resource.

- XML Analysis for Integration Query result. This indicates the time that it takes to analyze the result (XML document) for the integration of the query result.
- Query Result Integration. This indicates the time that it takes to integrate the result.
- Convert into Web-based Result page. This indicates the time that it takes to convert into the web page in order to provide the integrated result on the user interface.
- Computing Resource Usage. This indicates CPU and memory occupancy when the query is working.

Excepting the system performance factors we mentioned above, others are following: 1) it is a case that data is inserted into the database; 2) it is a case that a security component is configured; 3) and it is a case that DOM parser and/or SAX parser is used in order to parse XML document (i.e. the query result)

The following tools can be used for analysis of the system performance.

- J2SDK [11] and Apache Ant [12], which are used for the source files' compiling and deploying.
- *System.currentTimeMillis*, which is used to record the current system time.
- Apache Log4j [13], which is used to log the information needed.
- EJ-JProfiler [14] and Borland Optimizeit [15], which are used to monitor CPU and memory occupancy.

4. Implementation

In this section, we will present the implemented modules and the environment configuration for implementation of GDAIS, and will explain the implemented modules.

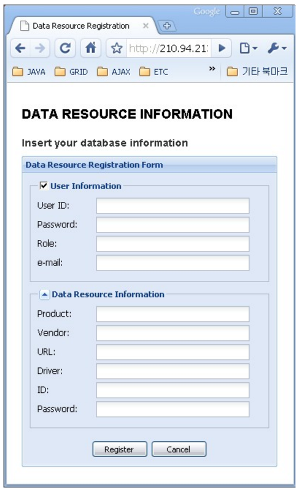
4.1 System Implementation Environment

The proposed system is implemented GDAIS using Java programming language. The system runs on top of Globus Toolkit WS-Core 4.2.0.1 and Apache Tomcat 5.5.26 server. Our system setup consisted of two machines for access the data resource and a machine for other services on the same LAN. A machine for GT was a 2.8GHz Intel Pentium system running Fodora Core 6 Linux with the 2.6.x kernel and J2SE 1.6.0_07. Other machine was a dual 2.40GHz Intel Xeon system running Fodora Core 6 Linux with the 2.6.x kernel and J2SE 1.5.2_08. The machine for Tomcat was a dual 1.60GHz Intel Pentium system running Windows XP with J2SE 1.6.0_07.

The databases used in this system were an open source MySQL 5.0.22 with MySQL Connector/J driver version 3.1.1.0 and PostgreSQL 8.3.3 with PostgreSQL Connecotr Driver version postgresql-8.3-603.jdbc4. And we used Korea postcode for all experiments. MySQL and PostgreSQL database have databases, my1, my2, my3, pos1, pos2, and pos3 with 500 data separately. And each table is 65536 bytes. The rows have following schema: seq int(7) not null, zipcode varchar(7), sido varchar(4), gugun varchar(15), dong varchar(52), bunji varchar(17), PRIMARY KEY(seq), index dong(dong).

4.2 Unified User Interface

A user can use all services using the web-based user interface. As you can see Figure 3, a provider has first to register the database information to the system. When the work which is a database registration is completed, you can see available data resource now.



DATA RESOURCE INFORMATION

Insert your database information

Data Resource Registration Form

User Information

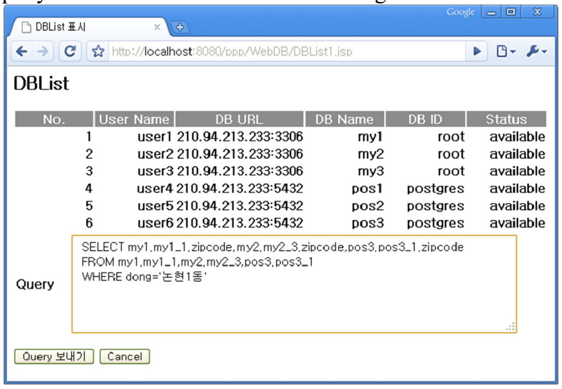
User ID:	
Password:	
Role:	
e-mail:	

Data Resource Information

Product:	
Vendor:	
URL:	
Driver:	
ID:	
Password:	

Figure 3. Registration Form

A user writes a query statement for transaction of a distributed query such as Figure 4. A written query statement is conducted by the distributed query service, and then the query results return to the user such as Figure 5.



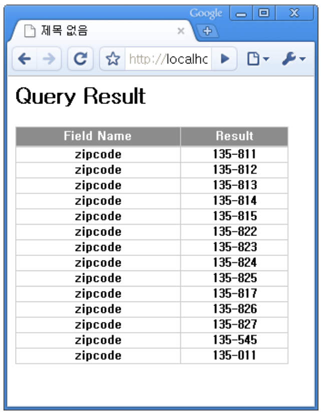
DBList

No.	User Name	DB URL	DB Name	DB ID	Status
1	user1	210.94.213.233:3306	my1	root	available
2	user2	210.94.213.233:3306	my2	root	available
3	user3	210.94.213.233:3306	my3	root	available
4	user4	210.94.213.233:5432	pos1	postgres	available
5	user5	210.94.213.233:5432	pos2	postgres	available
6	user6	210.94.213.233:5432	pos3	postgres	available

Query

```
SELECT my1.my1_1.zipcode,my2.my2_3.zipcode,pos3.pos3_1.zipcode
FROM my1.my1_1,my2.my2_3,pos3.pos3_1
WHERE dong='논현1동'
```

Figure 4. Query Processor Service



Query Result

Field Name	Result
zipcode	135-811
zipcode	135-812
zipcode	135-813
zipcode	135-814
zipcode	135-815
zipcode	135-822
zipcode	135-823
zipcode	135-824
zipcode	135-825
zipcode	135-817
zipcode	135-826
zipcode	135-827
zipcode	135-545
zipcode	135-011

Figure 5. Query Result

The data resource in Data Grid environment has heterogeneous features. When a user conducts a distributed query, the system searches an applicable JDBC driver according to each data resource. If every distributed query has to search the JDBC driver, the system will be overloaded. This system provides a connection pool service which can reduce connection time and disconnection time. So the query process time is faster. Because it is also controlling the connection number at a time, it has an advantage that the system can reduce overload.

4.3 Distributed Query Service

The database integration in Grid environment for users can be accomplished by the query process mechanism which conducts a query to many heterogeneous databases. In addition, it also guarantees database independence and data security. As you can see bottom, our system uses the transformed query statement for query process to the distributed heterogeneous databases.

```
SELECTE  GD_ID1.table_name1.field_name1,
         GD_ID1.table_name2.field_name3,
         GD_ID2.table_name2.field_name1
FROM  GD_ID1.table_name1, GD_ID1.table_name2,
      GD_ID2.table_name2
WHERE  Conditions…
```

The written SQL statement in above is a command which conducts to select the data from table in the specific database. Unlike other SQL statements, this command can query to all registered database. For example, the written SQL statement conducts a query to ‘GD_ID1’ and ‘GD_ID2’ database. Each query result from two databases is returned to user after the JOIN operation. However, if the corresponding field doesn’t exist, the query result will be returned separately. Furthermore, it can also use the SELECT statement with the WHERE condition statement, so is can conduct to query with a specific condition.

4.4 System Security

The Grid Security Infrastructure (GSI) [16, 17] is the portion of the Globus Toolkit that provides the fundamental security services needed to support Grids. Under the Grid environment, to solve the security problem of the process in the access and the integration of database, system transfers the security policy provided by the Grid security infrastructure and base on the news rank, and it combines the lifecycle control characteristic which the Grid service possess. Therefore, it effectively enhances the security of the database access and the data manipulation.

5. Conclusion and future works

In this paper, we designed and implemented the GT4-based database access and the integration service. Moreover, this service was implemented as Grid service for run in any Grid environment. Additionally, system security was configured through GT4’s GSI components. Therefore it can improve the efficiency to distribute and manage databases, and it can easily access and integrate of the distributed heterogeneous data in Grid environments.

We plan an experiment for system stability and excellence in compliance with the above experimental factors. And we also plan to apply the proposed service to the specific Grid application, so that we verify the practical proposed service’s efficiency.

References

[1] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of High Performance Computing Applications, 15(3):200-222, 2001.

[2] I. Foster, C. Kesselman, J. Nick, S. Tuecke. “The Physiology of the Grid-An Open Grid Services Architecture for Distributed Systems Integration”, Global Grid Forum, June 22, 2002.

[3] Globus Toolkit Primer, See http://www.globus.org/toolkit/docs/4.2/4.2.0/key/GT4_Primer_0.6.pdf.

[4] Xiufen Fu, Haishui Xu, Wenguo Hou, Yangseng Lu, and Boxing Chen, “Research of the Access and Integration of Grid Database”, Proc. of the 10th International Conference on Computer Supported Cooperative Work in Design, 2006.

[5] I. Foster, C. Kesselman, J. Nick, S. Tuecke. “Grid Services for Distributed System Integration”, Computer, 35(6): 37-46, 2002.

[6] I. Foster, J. Voeckler, M. Wilde, et al., A Virtual Data System for Representing, “Querying and Automating Data Derivation”. Proceedings of the 14th Conference on Scientific and Statistical Database Management, Edinburgh, Scotland, July 2002.

[7] OGSA-DAI. Available from URL: http://www.ogsadai.org/.

[8] Xiufen Fu, Haishui Xu, Wenguo Hou, Yangseng Lu, and Boxing Chen, “Research of the Access and Integration of Grid Database”, Proc. of the 10th International Conference on Computer Supported Cooperative Work in Design, 2006.

[9] Database Access and Integration Services Working Group, DAIS-WG. Available from URL: http://www.ogf.org/gf/group_info/view.php?group=dais-wg.

[10] N. Hardman, A. Borley, J. Magowan, OGSA-DAI: A look under the hood: Part 1: Architecture and database access, IBM United Kingdom Limited, January 2004.

[11] Java Development Kit (J2SDK), See http://java.sun.com/j2se/.

[12] Apache Ant. See http://ant.apache.org/.

[13] Apache Log4J. See http://jakarta.apache.org/log4j/.

[14] Borland Optimizeit. See http://www.borland.com/.

[15] EJ-Enterprises JProfiler. See http://www.ejtechnologies.com/products/jprofiler/overview.html/.

[16] Globus Toolkit 4.0 Security. Available from URL: http://www.globus.org/toolkit/docs/4.0/securit y/.

[17] Grid Security Infrastructure (GSI). Available from URL: http://www-unix.globus.org/toolkit/docs/3.2/index.html#core.