Design and Development of Database Integration Framework for Distributed Queries in Parallel using Open Grid Service Architecture

Thin Thin Nwe, Myo Hein Zaw
University of Computer Studies, Monywa
thinthinnwe07@gmail.com,mheinzaw@gmail.com

Abstract

The growing number of large knowledge bases necessitates developing techniques that enable the sharing and a cooperative scheme for accessing massively distributed, heterogeneous databases, created and managed independently by several research institutions. However, enabling the sharing of databases without following a predefined common schema and supporting parallel execution of distributed queries are remained as major issues and challenging tasks towards research community. In this paper, a database integration framework that supports execution of distributed queries in parallel has been proposed. Mediator–wrapper architecture has been used to implement the system in which existing services of Open Grid Service Architecture-Data Access and Integration (OGSA-DAI) middleware has been used as wrappers to provide a standard interface to heterogeneous data resources. A distributed query processor (DQP) has been implemented using extensibility points of it to serve as a mediator that provides a global view of wrapped data sources and supports execution of distributed queries in parallel. The frame work has three main parts: data provider layer; Data Grid infrastructure and a user interfacing portal.

Keywords *– OGSA-DAI, Grid Computing, DQP*

1. Introduction

A wide range of scientific applications, such as life and environmental sciences, are highly dependent on database management systems for organizing and storing their data. Since the need to access and analyze datasets from these database management systems are often large and almost always geographically distributed, they have been trying to find a way that allows the sharing of and access to massively distributed and autonomous databases. Data Grids [1] provide efficient solutions to scientific data sharing to collaborate across institutional and geographic boundaries. Most early works on Data Girds focused principally on the provision of infrastructures for managing and providing efficient access to file-based data [2] and proposed solutions only for file-level sharing. We propose a flexible database sharing system to provide highly scalable collaborative environment, while supporting SQL-like content-based queries without a globally shared database schema. This work is based on OGSA-DQP (Open Grid Service Architecture-Distributed Query Processing) [3]. OGSA-DQP is based on the Grid middleware known as OGSA-DAI (Open Grid Service Architecture-Data Access and Integration) [4,5] which is built on OGSA (Open Grid Service Architecture). To enhance information querying over large scale distributed databases, we build OGSA-DQP, which provides distributed query processing functionalities, on top of OGSA-DAI [6,7]. The rest of the paper is structured as follows. Section 2 presents the overview of our proposed system model. Section 3 gives a more detail explanation on the implementation architecture of our system, and Section 4 is Experimental Evaluation and finally, we conclude our paper in Section 5.

2. Overview of System Model

In our proposed system, geographically distributed databases are federated into the Data Grid to provide consistent access to them from Grid applications and to help coordinated use of multiple databases from the Grid middleware. The overview of our proposed Grid-based database sharing system is illustrated in Figure1.

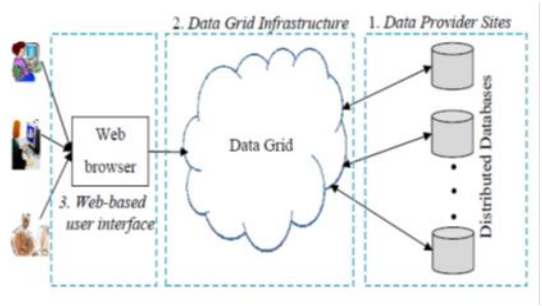

Figure 1. Overview of System Model

Each and every distributed database in the system has a complete database management system itself, with access control, catalog, and query processing etc. These database management systems are integrated into the Data Grid to share their resources. Instead of storing all of the datasets in one place, our proposed system will exploit the capacity of other distributed database management systems' storage and query processing capabilities.

3. Proposed Database Integration Framework

The overview of our proposed Grid-based database integration framework is illustrated in Figure 2. It consists of three main parts: (i) Data Provider Layer (ii) Data Grid Layer and (iii) Grid Portal.

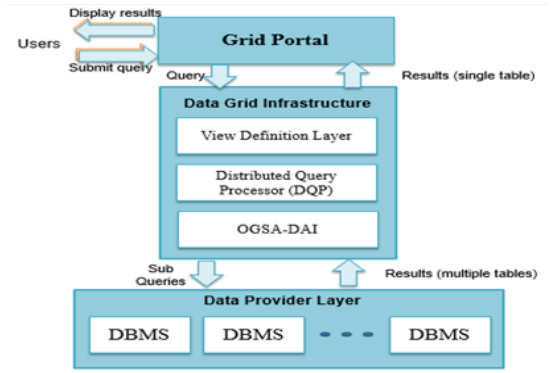

Figure 2. Implementation Architecture of Proposed System

3.1. Data Provider Layer

This includes distributed database management systems which use database servers to handle large amount of records which are stored in relations, where each for row (called tuple) represents a data item and each column is an attribute describing those items. Different databases have been created without following a predetermined common schema. Each system is a complete DBMS in itself with access control, catalogs and query processing etc.

3.2. Data Grid Layer

The Data Grid layer plays a vital role in our proposed system. It aggregates all dispersed data resources into a single and uniform view. It also addresses authentication, secure transfers and mechanisms for searching desired information provided by users.

3.2.1 Integrating Database into the Grid

At the level of Data Grid layer, there are two notable Grid middle wares, OGSA-DQP and OGSA-DAI. The OGSA - Data Access and Integration (OGSA-DAI) project has developed a service-based infrastructure for wrapping and accessing both relational databases and XML repositories in a uniform way as services [5,6]. Although the core OGSA-DAI data service provides a useful abstraction for accessing individual data resources on the Grid, they do not address challenges associated with integrating data from multiple resources. Extending the multiple data source functions of OGSA-DAI, we build OGSA-DQP, which is a service based distributed query processor for planning, scheduling and executing distributed queries in parallel, on top of OGSA-DAI. At a lower level of our Data Grid layer, lies GT4 (Globus Toolkit 4), the OGSA reference implementation which is used both by OGSA-DQP and OGSA-DAI for service instance creation, service state access and life time management of services.

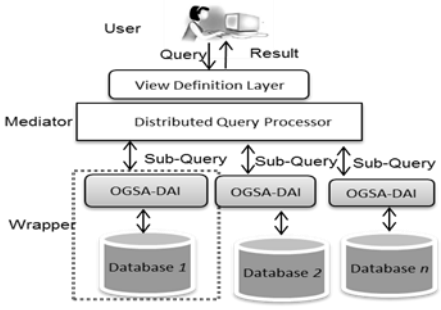

Figure 3. Mediator-Wrapper Vision of OGSA-DQP

Our service based DQP approach described in this paper functions as an integration component allowing queries to be composed over multiple OGSA-DAI wrapped relational data resources. Our proposed system uses OGSA-DAI middleware as the

wrapper and DQP as the mediator to integrate geographically distributed databases into the Grid. Figure 3. illustrates how distributed databases are wrapped by OGSA-DAI services.

3.2.2 Distributed Query Processing in Data Grid Layer

OGSA-DQP provides two services to fulfill its functions: the Grid Distributed Query Service (GDQS) and the Grid Query Evaluation Service (GQES). The GDQS- also known as the OGSA-DQP Coordinator is the main interaction point for the clients. When a coordinator is set up, it obtains the metadata and computational resource information that it needs to compile, optimize, partition and schedule distributed query execution plans over multiple execution nodes in the Grid. The coordinator is currently implemented as a set of OGSA-DAI data resources and activities. The GQES- also known as the DQP evaluator is used by the coordinator to execute query plans generated by the query compiler, optimizer and scheduler. An evaluator is able to play a role in the evaluation of a query by retrieving data from OGSA-DAI-wrapped data resources, invoking analysis services and managing the flow of data between other evaluators. Multiple evaluators are used to provide the benefits of parallelism during query evaluation. Each evaluator evaluates a partition of the query execution plan assigned to it by a coordinator. A set of evaluators participating in a query form a tree through which the data flows from leaf evaluators which interact with Grid data services, up the tree to reach its destination.

3.2.3 Setting up a distributed query service

An OGSA-DQP coordinator consists of two types of OGSA-DAI data service resources: GDQS factory data service resources and GDQS data service resources. GDQS factory data service resource maintains OGSA-DQP system level installation data and supports the execution of the DQPFactory activity, which is introduced to enable configuration. The GDQS data service resource represents a federation of data resources and analysis services over which queries may be composed using DQPQueryStatement activities, and the pool of evaluator services which may be utilized to evaluate such queries. GDQS factory service is deployed on a host machine that plays the role of query coordinator. This data service resource is then used to create GDQS data service resources which can be used by a client to execute queries. In this first step in the interaction between a client and the OGSA-DQP, the client uses a deployed GDQS factory data service resource to create a configured GDQS data service resource. The client interacts with the GDQS factory data service resource by sending OGSA-DAI perform document which specifies that a DQPFactory activity should be executed. The DQPFactory activity is able to interact with a GDQS factory data service resource in order to dynamically deploy a GDQS data service resource. The client interacts with the GDQS factory data service resource by sending an OGSA-DAI perform document [7] which specifies that a DQPFactory activity should be executed. The DQPFactory activity is able to interact with a GDQS factory data service resource in order to dynamically deploy a GDQS data service resource. The DQPFactory activity is parameterized by an XML document which specifies exactly how the deployed GDQS data service resource should be configured. Configuration parameters include the databases and evaluators which can be utilized by the data service resource which is to be created. The result of this interaction is that a GDQS data service resource is created and initialized. The coordinator service now exposes this dynamically deployed GDQS data service resource and it is automatically assigned a resource ID by OGSA- DAI.

3.2.4 Executing distributed queries

The functional flow diagram of distributed query processor is shown in Figure 4

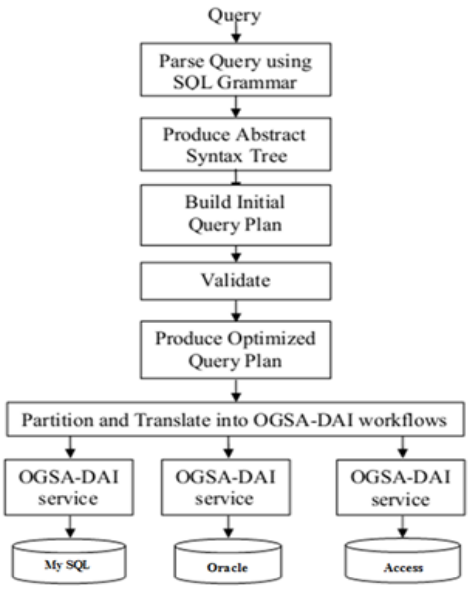

Figure 4. Functional Flow Diagram of Distributed Query Processor

That is used to plan, schedule and execute distributed queries in parallel. Java programming language has been used to implement all the functionalities. When a DQP resource receives a query, the following actions take place:

(1) The query is parsed according to SQL grammar to produce an abstract syntax tree.

(2) The produced abstract syntax tree is used to build an initial query plan.

(3) This initial query plan is then validated and normalized.

(4) The query plan is optimized by a chain of query plan transformers for efficient execution. This includes deciding on which OGSA-DAI server each operation will be executed.

(5) The optimized query plan is split into multiple partitions and converted into multiple OGSA-DAI workflows

(6) Workflows are sent to the associate OGSA-DAI services which in turn perform the activities specified, contact the relational databases and execute the query in parallel.

4. Experimental Evaluation

The main objective of our research is to integrate geographically distributed databases (two MySQL on Ubuntu, Oracle 11g on Window7, Microsoft access 2010 on window 8 with heterogeneous platform) into the Grid and to provide a single point of access to all dispersed data resources to enable effective collaboration and sharing. In this section, we describe experimental evaluations to illustrate the overall performance that has been obtained by using our proposed system architecture and to show the benefits of parallel query processing facilities of DQP. Configuration of DQP is as follow:

```
<?xml version="1.0" encoding="UTF-8" ?>
<DQPResourceConfig>
<dataResources>
<resource url="htp://10.0.0.1:8080/dai/services"
resourceID="MonywaCUResource"isLocal="true" />
<resource url="http://10.0.0.1:8080/dai/services"
resourceID="RemoteKalayCUResource"isLocal="fal
se" />
<resource url="http://10.0.0.1:8080/dai/services"
resourceID="RemoteMagwayCUResource"isLocal="
false" />
<resource url="http://10.0.0.1:8080/dai/services"
resourceID="PhakhoukuCUResource"isLocal="false
" />
</dataResources>
</DQPResourceConfig>
```

To support federation of database without a global schema and solve the problem of schema mismatch problem, views are defined over each single data resource take for example:

```
Information ->SELECT Student_ID as Student_No,
Student_Name as Name, Major, Year,
Eng_Mark,AvgMajor_Mark, Academic_YearFROM
information

Student -> SELECT S_ID AS Student_No, S_Name as
Name, Major, Year, Eng_Mark, AvgMajor_Mark,
Academic_Year FROM student

student_record->SELECT ID as Student_No, Name, Major,
Year, Eng_Mark, AvgMajor_Mark, Academic_Year from
student_record
```

Local mismatch schemas of each data resource are mapped to the virtual schema in these defined Views.

In the proposed system, what the system would like to do is to give answers to questions like: ‘Give me the names and roll numbers of all students who have got greater than 55 marks in English subject’. In other words, users are interested in a single database and single table view of all the data sources. Instead of writing complex queries, the proposed system would allow to simply ask:

```
SELECT * FROM Federation as F
        WHERE F.Eng_Mark>55;
```

To make this possible, the Federation View resource is deployed, which defines a View using the following mapping:

```
Federation SELECT Student_NO, Name FROM
(SELECT* FROM MonywaCUResource_information
UNION ALL SELECT * FROM RemoteKalayCUResource_info
UNION ALL SELECT * FROM RemoteMagwayCUResource_student
UNION ALL SELECT * FROM PhakhouKuCUResource_student_record
) as A
```

Views have powerful data integration capabilities. Views can be used to smooth out differences between table schemas. These mappings exploit the expressiveness of the SQL language and can range from simple column renaming to complex, value-replacing joins. The appearance result of system is shown in Figure 5.

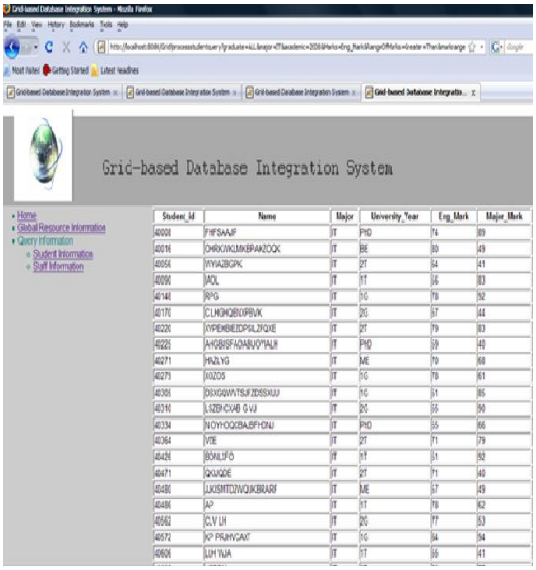

Figure 5. The appearance result of system

4.1 Performance Comparison

Results presented in this section mean what level of performance one may expect when running queries over federated databases.

Four different types of experiments were conducted as follows:

Experiment 1: Queries were sent from the user to the databases involved in the federation using direct Java Database Connectivity (JDBC) connection.

Experiment 2: Queries were sent from the user to the OGSA-DAI wrapped data resources using OGSA-DAI Web service clients.

Response time comparison between accessing each single data resource using direct JDBC and using the proposed system is illustrated in Figure 6. It can be seen clearly from the results that using the proposed system shows longer response time because of the overheads imposed by the Web service layer and serialization. Literature review also indicates that the OGSA-DAI database wrappers are around an order of magnitude slower than JDBC calls for bulk delivery on a local area network.

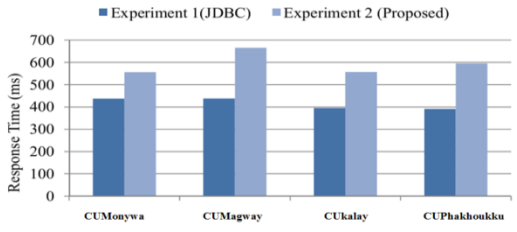

Figure 6. Average Response Time for Accessing Single Data Resource

For next level of research, experiment 3 and 4 are carried out.

Experiment 3: All the datasets in four distributed databases are transferred to a centralized database located on the OGSA-DAI server and the query was submitted to this database using direct JDBC connection.

Experiment 4: Queries were sent from the user to the Federation resource, which federates over distributed data resources. In this scenario, data are being transferred over the network.

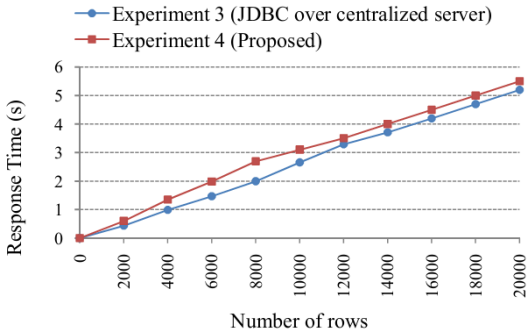

Figure 7. Response Time Comparison between Centralized Server and Proposed System

The second graph presenting the response time comparison for accessing from the proposed database federation and from a centralized database server for different number of rows is illustrated in Figure 7. Since there is a significant performance overhead associated with the creation and unpacking of XML data representations, and for sending in SOAP messages in the proposed system, accessing from a centralized server using direct JDBC shows faster response time than the proposed system. However, all the datasets in the centralized servicer are necessary to follow a common schema and it may require large storage capacity and longer response time for very large amount of datasets.

5. Conclusion

This research attempts to solve this by employing and extending the data access and integration services provided by the OGSA-DAI, allowing geographically distributed, heterogeneous and autonomous database management systems to be exposed in a uniform way as services without a globally shared database schema. The distributed query processor has also been designed and implemented on top of OGSA-DAI as a collection of cooperating services to compile, parse and execute the query in parallel. SQL View functionality has also been integrated in the system to avoid writing

complex queries and to solve schema mismatch problems. A user friendly Web-based interface has been developed to create multiuser environment and provide a single access point to all dispersed data resources. Since the proposed system utilizes advances of Grid technologies to federate geographically distributed databases as a single virtual database, users from different locations can access information from different database servers with no necessary to connect directly to each database and physical location of the data is transparent to them. Location transparency and site autonomy are supported by the system. Lack of collaboration and storage capability problem can also solved by adopting the proposed system where each DBMS still belongs to and is controlled by its original resource provider and the system makes them possible to join and share their resources without losing administrative control. Not like former parallel database designs, the system supports execution of distributed query in parallel across several organizations and administrative domains.

Enabling the sharing of databases without using a global schema and providing highly scalable collaborative environment are the main desirable features of the system, making it a good candidate for Data Grid applications and distributed database management systems. The cost of the proposed system over direct JDBC is offset by the benefit of presenting and querying distributed data sources as if they were a single data source. Whether this trade-off is acceptable or not is for specific users to determine.

References

[1] Ahmar Abbas, Grid Computing: A Practical Guide to Technology and Applications, ISBN:1-58450-276-2

[2] Armey Krause, Gorgi Kakasevski "*OGSA-DAI extensions for executing distributed data mining workflows*", ICT Innovations 2013 Web Proceedings ISSN 1857-7288

[3] Bartosz Doberzelecki, Amrey Krause, Alastair . C. Hume, Alistair Grant, Mario Antonioletti, Tilaye Y. Alemu, Malcolm Atkinson, Mike Jackson and Elias Theocharopoulos, " *Integrating distributed data sources with OGSA-DAI DQP and Views*", Phil. Trans. R. Soc. A (2010) 386, 4133-4145 doi:10.1098/rsta.2010.0166

[4] Gorgi Kakasevski, Anastas Misev and Boro Jakimovski, "*Heterogeneous Distributed Databases and Distributed Query Processing in Grid Computing*", ICT Innovations 2011 Web Proceedings ISSN 1857-7288

[5] The OGSA-DQP project, http://www.ogsadai.org.uk/about/ogsa-dqp

[6] The OGSA-DAI project, http://www.ogsadai.org.uk

[7] Globus Toolkit 4: www.globus.org, http://www.ogsadai.org.uk/documentation/ogsa-dqp_4.0/toolkit.html