A COMPREHENSIVE FRAMEWORK FOR GRID DATA MANAGEMENT[1]

Dang Tran Khanh, Phan Thi Thanh Huyen, Vo Hoang Tam
University of Technology, VNU-HCM

1. INTRODUCTION

Grid-based applications frequently manipulate data distributed in wide and heterogeneous environments. Therefore, data management is increasingly becoming significant in Grid computing area. Open Grid Services Architecture - Data Access and Integration (OGSA-DAI) [2] is a basic middleware that allows data in data resources of different kinds (including relational, XML and files) in Grid environment to be queried, updated, transformed and delivered in a consistent, data resource-independent way.

However, data access is just a facet in data management. Some of other requirements in data management need to be fulfilled are data distribution, data integration, data security, etc. To tackle this problem, users can extend OGSA-DAI web services to expose their own data resources and to support application-specific functionality. Although these approaches address various requirements in Grid data management, their data management components can not become a framework for Grid-based applications in managing data. None of previous work provides a general solution for this task.

In this article, we design and implement a comprehensive Grid Data Management (GDM) framework, which enables applications running in Grid environment to manage their data in an easy way with many services that meet requirements of data distribution, data access and integration, data security, performance analysis, monitoring, and billing. The GDM framework comprises of two modules: GDM core and GDM client. The former deals with supplying services of the framework and the latter helps applications interact with the former. This architecture not only hides the complexity of the Grid system from the users, but also improves the management of data resources in Grid environment.

The rest of this article is organized as follows. The next section briefly discusses related work and outlines data management challenges in Grid environment. In section 3, we present the architecture of the GDM framework in detail. Section 4 describes the implementation of the framework. Conclusions and future work are given in section 5.

2. RELATED WORK

In this section, we discuss some existing research work concerning with developing Grid-based applications to motivate the need of a Grid data management framework.

One of the most popular software toolkits for Grid computing is the Globus Toolkit [1, 14]. Globus Toolkit (GT), developed by the Globus Alliance, has been recognized as the de facto standard in building Grid solutions. It is a collection of libraries and software services dealing with common problems that occur when building distributed system services and applications. GT includes fundamental softwares for security, information infrastructure, resource management, data management, communication, fault detection, and portability that can be used either independently or together to develop applications. In data management, GT supports data transfer using Grid FTP (File Transfer Protocol), RFT (Reliable File Transfer), and data replica management using Replica Location Service (RLS), Data Replication Service (DRS) to improve

[1] This research has been financially supported by the Vietnamese Ministry of Science and Technology through EDAGrid project.

the availability of data in Grid system [15]. One component of GT we want to elaborate on is Open Grid Services Architecture - Data Access and Integration (OGSA-DAI) [2].

As presented in [2], OGSA-DAI is a middleware product that allows data resources – including relational, XML and files – to be accessed via web services. These web services allows data in data resources to be queried, updated, transformed and delivered in a consistent, data resource-independent way. Information about these data resources and the functionalities supported by the service can also be accessed. Another important feature of OGSA-DAI is the extendibility. Users can extend web services to expose their own data resources and to support application-specific functionality, in addition to that provided by OGSA-DAI. Therefore, it can be used as a base to develop higher-level services such as data federation, distributed query processing, data mining, data visualization, etc. It provides web services compliant with two popular web services specifications: Web Services Inter-operability (WS-I) compatible with the UK OMII's implementation of WS-I, and Web Services Resource Framework (WSRF) compatible with the GT's implementation of WSRF. User can interact with a data resource exposed by OGSA-DAI by using a corresponding data service. Moreover, it also supplies the OGSA-DAI Client Toolkit, a Java API that provides the basic building blocks for client development. The Client Toolkit minimizes the specialist knowledge required to interact with OGSA-DAI services and provides some protection from future changes to the data service interfaces. With above properties, when deployed within a Grid environment, it thereby provides a means for users to Grid-enable their data resources. Using OGSA-DAI, managing and accessing data resources, especially the data resources in different formats become easier.

OGSA-DAI has been used in many Grid-based applications to manage their data. Most of projects currently use it as a piece of Grid middleware to access data resources such as GridMiner, SIMDAT, Provenance, etc. Other projects extend it to provide a more powerful tool. The GridMiner project [10], as the first Grid research effort, addresses all phases of knowledge discovery in databases and other large data sets integrated into Grid and implements them as an advanced service-oriented Grid infrastructure. For data accessing and integration a Grid Data Mediation Service is used which provides a single virtual data source with a global schema for the user. The developed concepts have been implemented by re-using the OGSA-DAI Grid Data Service as a framework to show their feasibility. This contribution is enabling access to more than one data source over a global schema with a subset of SQL and integrating the results in a standardized way. However, this component is so specific for Data Mining application that the ability to reuse or extend is not high. Having the same purpose with GridMiner is SIMDAT project [11], Provenance project [12], etc. The Secure Data Grid project [13, 3] is to develop a secure system for accessing databases across the Grid. In this project, a Role-Based Access Control (RBAC) mechanism is designed and implemented to enhance the existing security infrastructure of OGSA-DAI by incorporating the Community Authorization Service (CAS) provided by the Globus Toolkit. Similarly to Secure Data Grid, IBM OGSA-DAI Database Replication [4] project presents an opportunity to extend OGSA-DAI to provide it with more powerful capabilities. This IBM funded project aims to combine OGSA-DAI technology with the IBM DB2 data replication application to provide a Grid service interface for managing database replication. Grid-enabling database replication in this way will support more scalable and secure data replication across heterogeneous resources and multiple administrative domains.

In summary, the above projects address many requirements in Grid data management, but their components can not serve as a framework for Grid-based applications in managing data. This leads us to develop a Grid Data Management (GDM) framework which supports services for applications in Grid computing environment to manage their data in an effective and secure manner. These services include distributing, accessing and integrating data, performance monitoring, and billing service. The detail of this framework will be presented in section 3.

3. A COMPREHENSIVE FRAMEWORK FOR GRID DATA MANAGEMENT

In this part, section 3.1 presents an overview of our GDM framework's aims. Architecture and functional descriptions of each component of the framework are shown in Section 3.2. Section 3.3 illustrates a typical interaction between an application and GDM.

3.1.Goals of GDM

Software supporting the management of data on Grid should fulfill demands for distributing, accessing and integrating data in a secure manner as well as monitoring, analyzing performance and billing. The following four basic goals drive the design of our GDM framework.

- ***Data Distribution, Access and Integration***: In Grid environment, data is stored in different locations, leading to the requirements in data distribution, access and integration. Data Distribution supports the distribution of data to data resources while Data Integration is concerned with the integration of data from various data resources. Data Access is responsible for providing access to data in a consistent, data resource-independent way such as querying, updating, transforming and delivering data.
- ***Data Security:*** Data is protected when delivered and stored in different data resources. Another desired requirement is user management with the ability to map Grid users to database role.
- ***Performance Analysis and Monitoring:*** Monitoring the frequency of accessing data and operations on data, response time of the system, job status, etc. Using these parameters, we can improve the performance of the system by many methods such as adjusting the data distribution policy, predicting the response time for a specific operation, etc.
- ***Billing:*** Applications can be billed with respect to many criteria, for instance, the amount of data they have accessed and the complexity of operations on data, etc.

3.2.GDM Architecture

3.2.1.GDM in the Whole System

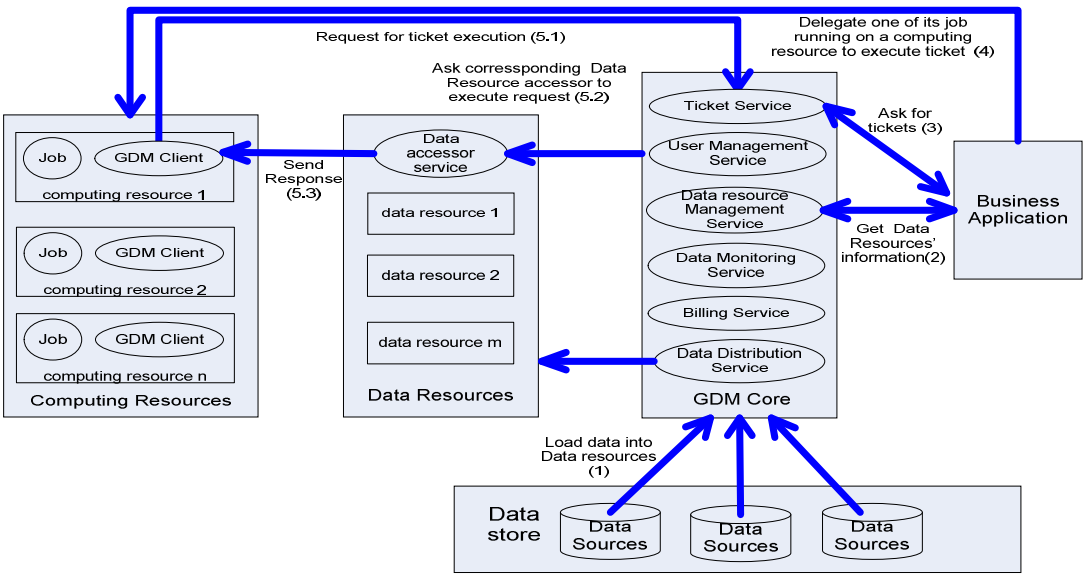

Figure 1. GDM framework architecture.

As illustrated in Figure 1, the main components of the whole system include computing resource, data resource, data store, Grid Data Management core, and business application. We detail them below:

- ***Computing Resource*** is the most common resource allocated to serve the computation in Grid environment and it is assigned a job or a part of job to execute.
- ***Data Resource*** refers to the data storage in Grid environment. Each data resource corresponds to a node in Grid.
- ***Data Store*** represents a data repository where data resides in before it is loaded into Grid environment. Most if not all of the data of Data Store are reflected in data resources. Data Store contains many data sources and a data source is usually distributed into many data resources. Therefore, every operation on data source will be replaced by (translated into equivalent) operations on corresponding data resources in Grid.
- ***Grid Data Management core*** (GDM core) plays the key role in the framework. GDM core is responsible for: storing and updating metadata of data resources in the system; supplying information about the distribution of data sources in data resources as well as data in data resources such as data type, table schema, data resource name, etc; authenticating and issuing roles for users (roles for accessing data about data resources, roles for querying and operating in a specific data resource); managing the issuing and execution of a ticket which represents a specific request on data; monitoring and analyzing the performance of the system.
- ***Business application*** is an application that has a need to work with data in data resources. Application will operate on data through the services supplied by GDM core.

3.2.2.Design of GDM Framework

GDM framework consists of two modules: GDM core and GDM client. The former deals with supplying all services of the framework and the latter helps applications interact with the former. The GDM core module comprises of the following major services:

Data Accessor Service: this service is responsible for interacting with Grid data resources. All requests of accessing and manipulating data in Grid data resources are handled by this service. OGSA-DAI services can be employed to implement Data Assessor Service for its ability to interact with different data resources to query, update, transform and deliver data.

Data Distribution Service: the task of this service is to load data in external data sources into data resources in Grid environment. All queries and subsequent analyses of the data are executed directly in data resources. Information about the distribution of data sources into data resources and information about data resources in Grid are stored in databases of GDM core. The data distribution strategy depends on many factors: requirement of a specific application or job, the number of data resources allocated, capacity and connectivity of every data resource, the frequency of accessing data and operations on data resource, system performance, etc.

The simplest solution here is equal distribution. Data in a data source is distributed equally into all allocated data resources. Besides equal distribution, there are other policies such as Round-Robin Distribution, Gaussian distribution, Random (White Noise) Distribution, Poisson Distribution [7], etc. We can also assign a parameter to each factor which affects the data distribution policy to calculate the priority of data resources. These parameters will be adjusted incrementally by learning from experimental results.

Data Resource Management Service: this service serves all requests for the properties of data resources such as data resource identification, database schema, data size, data distribution, etc. It will ask the User Management Service to authenticate and authorize user before executing

any requests by querying the database of GDM core to get the security related information. To increase the system's response, cache can be used to store frequently used data and it should be updated periodically to synchronize with data in database.

User management service: GDM framework is built on top of Globus Toolkit 4.0. Hence, an application will interact with GDM through a valid Grid user identified by a credential. An application can manipulate on many data sources. To work with a data source, it must be given an account (username, password) granted with some specific role. Information about roles of applications and users can be stored in a database of the GDM core.

The User management service can be implemented by using role-based access control (RBAC) of OGSA-DAI, which supports RBAC via a role-map file that maps individual Grid users to database roles. In this case, each resource provider has to maintain a role-map file to authorize access to its resources. In [3], the authors propose to use the Community Authorization Service (CAS) provided by the Globus Toolkit to support RBAC for multiple VOs to access Grid databases within the OGSA-DAI framework. The CAS records user groups and their permissions on resources and it targets access control for computational and file-based storage resources. In this method, the CAS grants users memberships on VO roles and then authorizes them in those roles. The resource providers need to maintain only the mapping information from VO roles to local database roles.

We can improve the flexibility of our service by using this solution. We can set some fixed roles on data resources and map Grid user role with these database roles in role-map files supplied by OGSA-DAI. After that, CAS server will be used to grant/revoke Grid users on the Grid user roles. With this method, we do not have to bother about individually adding/removing Grid user information in the role-map files any longer and leverage the features that Grid infrastructure supplies [6].

Ticket Service: in our system, every request for manipulating data in data resources, except for the requests for metadata of data resources, is expressed in a ticket form. Each ticket represents a specific request sent to the system such as data access, data update, etc. A ticket contains main information such as ticket identification, valid time, identification of ticket booking person, ticket type and the number of times in use. The remaining information will depend on the ticket type. There are four types of ticket: Ticket_Data, Ticket_Insert_Record, Ticket_Create_Table, and Ticket_Drop_Table. Ticket_Data is used to access data, so it will contain some additional information to specify the location of data in a table contained in a data resource: table name, data resource name. Ticket_Insert_Record is used to insert data into columns of a table belonging to a data resource. Information pertaining to this ticket are column names, table name, and data resource name. Ticket_Create_Table is used to create a new table in a data resource and hence its specific information is table name, data resource name and table schema. The last ticket type, Ticket_Drop_Table, is the request to delete a table. In this case, it just describes the table name and the data resource name.

An application which desires to manipulate on data must carry out two phases: booking ticket and executing ticket. In the first phase, depending on the type of ticket, application will send appropriate information. If the request is accepted, it will receive the ticket identification (ticket-id). In the second phase, to execute a ticket the application needs to send to Ticket Service a token which contains the ticket-id and the encrypted identification of the delegate. A token is generated by the application booking ticket. Applications can delegate another one to execute the ticket by using its private key to encrypt the delegate's identification and filling the second field of the token with the cipher. This token will be sent to the delegate later. The way to transfer token is out of the scope of GDM. In case the delegate is the application itself, application must still do the same work.

To process a booking-ticket request, the Ticket Service asks the User Management Service to authenticate and authorize user. If every data is valid, it stores ticket in database for later processing and sends the ticket identification to the user.

To execute a ticket, applications send the Ticket Service the ticket's corresponding token and its identification encrypted by its private key. Ticket booking person is identified based on the ticket-id in the token. Ticket Service will use the public key of token sender and ticket booking person to check that the token sender is the delegate for booking ticket person. If this check is positive, the Ticket Service will ask the Data Resource Management Service to execute the ticket. All operations on data distributed in data resources is handled by the Data Assessor Service. The result of this execution will be sent directly to the application by the Data Assessor Service.

Besides, we also have to consider the following problems concerning with booking and executing tickets. First, for the DoS problem, we can define the maximum number of tickets in a session or do not serve application's ticket request if the number of its unused ticket is above the allowed threshold. The value for this threshold can be adjusted depending on the history of using ticket of the application. Another problem is non-repudiation of receipt (NRR). NRR is intended to protect against a recipient's false denial of having received messages. Because applications use GDM client to interact with the Ticket Service, we can ask GDM client to notify the Ticket Service when the result arrives. The other problems such as lost ticket, canceled ticket, expired ticket will be considered in Billing Service.

Billing Service: applications are billed for executed tickets. Fee for an executed ticket depends on the type of that ticket and the complexity of operations required by the ticket. That a ticket will be charged when it has been executed or application has actually received the response depends on the efficiency of solution for NRR. It does not cost much with canceled tickets, lost tickets and expired tickets. The expired tickets will be more expensive than lost tickets and canceled tickets are the cheapest. Charge for these tickets can reduce the DoS problem and load of illegal requests sent to the system. Besides, maximum of the unexecuted tickets of a particular user must be set in order to cope with the DoS problem of ticket request flooding.

Data Monitoring Service: this service monitors requests of booking/executing tickets, and analyzes the demands on data from applications. Based on the time-instant when a request is sent and when the ACK of this request is returned from the GDM client, this service can evaluate the performance of the system. The Data Monitoring Service can be developed to visualize these values and status in real time.

3.2.3.A Typical Scenario for Communication between Applications and the GDM Framework

Figure 2 describes a typical interaction between an application and the GDM. This application desires to work with the data sources distributed in some data resources of GDM. Thus it will contact the GDM to get information about these data resources and then requests for some tickets. Because this application consists of many jobs that are scheduled to run on computing resources in Grid, it will grant these jobs the tokens to execute tickets. The interactions between the application and the GDM as well as the interactions between its jobs and the GDM are implemented through the GDM client. For simplicity, in this scenario we just show the GDM client that helps a job of an application communicate with the GDM core.

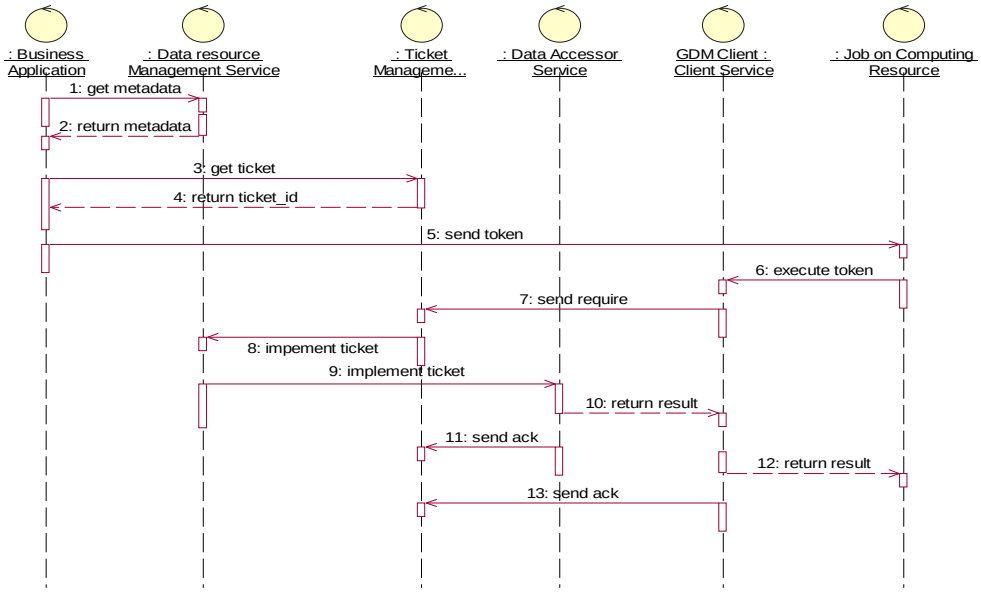

Figure 2. A typical interaction between an application and GDM.

The sample of steps taken to process a data access requirement is as follows:

- Business applications contact the Data Resource Management Service to get meta-data about data sources.
- The Data Resource Management Service returns the corresponding information to the application such as data resource identification, data size, data schema, etc.
- Depending on its business task, the application will request the Ticket Service for some tickets.
- The Ticket Service generates some tickets and stores them before returning ticket-ids to the applications.
- Applications can delegate its jobs to execute tickets by transferring the tokens containing ticket-ids to them (In figure 2, this job is represented by the computing resource where it runs). This transfer is not in the scope of our system.
- The job passes the tokens to the GDM client so that the GDM client can contact the GDM core to execute the associated tickets.
- The GDM client forwards the requests to the Ticket Service.
- The Ticket Service checks the token before sending it to the Data Resource Management Service.
- The Data Resource Management Service sends the ticket's content to the corresponding Data Assessor Service.
- The Data Assessor Service sends the result to the GDM Client.
- The Data Assessor Service also sends an ACK to the Ticket Service to notify of its finished work.
- The GDM client forwards the result to the job on computing resources.
- The GDM client sends an ACK to the Ticket Service to inform this service about the successful transaction.

4. IMPLEMENTATION

After architectural introduction, we now go into more details with a prototype implementation of the framework. The GDM framework, designed and built in this article, comprises of two modules: GDM core and GDM client. The first module is responsible for supplying all services of the framework. The second module provides an interface to interact with the system.

As illustrated in Figure 3, the GDM framework has been implemented on top of OGSA-DAI. To build this framework, we use Java language for the main programming language, and MySQL for DBMS.

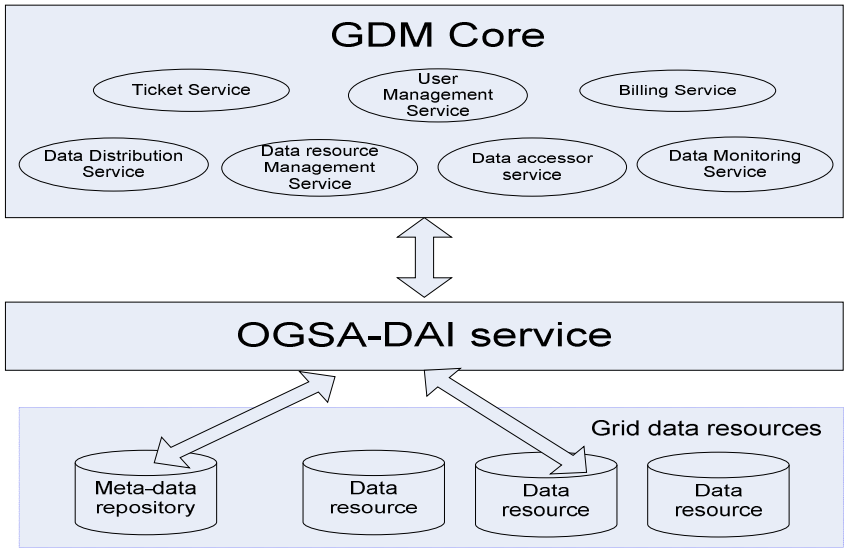

Figure 3. Implementation architecture of GDM framework.

Figure 4 shows the class diagram of GDM implementation. We have developed the GDM core as a Grid service [8] serving all the data manipulation requests from the users. To remove the burden of programming in Grid environment from the programmers, we have also defined Grid Data Management Java API, a standard interface to the system that includes data manipulation operations. This interface is provided and implemented through the GDM client module.

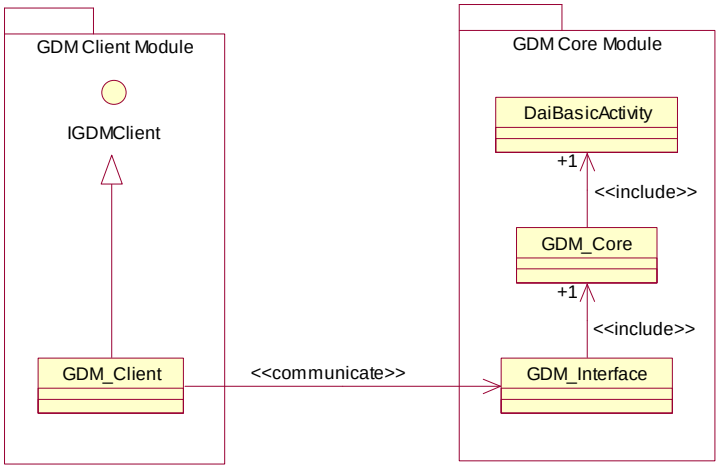

Figure 4. Class diagram of GDM implementation.

The functionality descriptions of classes and interfaces depicted in the above class diagram of GDM implementation are listed below in table 1 and table 2. Class GDM_Core is the major class in GDM core module. It contains full implementation (source code) of GDM core. Class GDM_Interface embodies an object of class GDM_Core. It is responsible for exhibiting the public functionalities of GDM core. Class DaiBasicActivity provides a programming interface for users to work with OGSA_DAI. Class GDM_Core communicate with OGSA_DAI through this class.

Table 1. Descriptions of classes in GDM Core module.

Class name	Description
GDM_Core	Major class in GDM core module. It contains full implementation of GDM core.
GDM_Interface	Embody an object of class GDM_Core. It is responsible for exhibiting the public functionalities of GDM core.
DaiBasicActivity	Provide a programming interface for users to work with OGSA_DAI. Class GDM_Core communicate with OGSA_DAI through this class.

In GDM Client module, the Java interface IGDMClient provides a Java API for users to work with GDM framework. The functionalities provided in IGDMClient interface can be grouped into two categories. The first group is reserved for clients who desire to manipulate data on grid through GDM framework. The administrator of GDM can also perform his management operations through this IGDMClient interface. Class GDM_Client, an implementation of IGDMClient interface, takes the role of communicating with GDM core module.

Table 2. Descriptions of classes and interfaces in GDM Client module.

Class name	Description
IGDMClient	Provide a Java API for users to work with GDM framework.
GDM_Client	Implementation of IGDMClient interface. It takes the role of communicating with class GDM_Interface in GDM core module.

The GDM framework implementation has been tested thoroughly and applied successfully in a grid data mining application [9]. The volume of data feeding into data mining algorithms might be very large. They are usually divided into partitions stored separately on distributed data resources. Also, the format and the structure of them might be completely different, for example one partition is a table stored in a DBMS while the others are XML files. For the fact that data mining usually processes business data, issues like security and billing for data access are also concerned problems. The GDM framework has fulfilled all these requirements. Instead of building a grid data management from scratch, in [9] the authors used our GDM framework as a data management layer in its proposed software architecture for data mining in grid environment.

5.CONCLUSION AND FUTURE WORK

In this article, we have proposed a comprehensive framework for Grid Data Management (GDM) supporting indispensable services for applications in Grid computing environment to manage their data in an effective and secure manner. These services include distributing, accessing and integrating data in Grid environment, performance monitoring, and billing service. We have also presented the main modules of our framework, which are GDM core and GDM client. The first module is responsible for supplying all services of the framework. The second module provides an interface to interact with the system. A prototype of the framework has been developed on top of OGSA-DAI to demonstrate the features and functions of the system. The framework not only hides the complexity of the Grid system from the users, but also improves the management of data resources in Grid environment.

In the future, we plan to intensively carry out evaluations and develop more real-world applications on the system to establish the practical value of the proposed framework [6]. Grid data mining is one of these applications [9]. Further research is also needed to enhance the efficiency of individual modules of the framework and the friendliness of the system interface. Moreover, to strengthen the system and bring end-users convenient and flexible tools, additional add-on features such as visualization of job status and system performance monitoring are also of our great interest in the future.

REFERENCES

[1]. The Globus Toolkit: www.globus.org/toolkit, (2006).

[2]. The OGSA-DAI project: www.ogsadai.org.uk, (2006).

[3]. Anil L. Pereira, Vineela Muppavarapu, and Soon M. Chung, *Role-Based Access Control for Grid Database Services Using the Community Authorization Service*, IEEE Transactions on Dependable and Secure Computing, Vol. 3, No. 2, pp. 156-166, (2006).

[4]. IBM OGSA-DAI Data Replication website: www.aiai.ed.ac.uk/~ychen/ibm_ogsadai/ibm-ogsadai-index.html, (2006).

[5]. Philip A. Bernstein, *Middleware: A Model for Distributed System Services*, Communication of ACM, Vol. 39, No. 2, pp. 86-98, (1996).

[6]. The EDAGrid Project website: www.edagrid.hcmut.edu.vn, (2006).

[7]. Michael Di Stefano, *Distributed Data Management for Grid Computing*, John Wiley & Sons, Inc., (2005).

[8]. Borja Sotomayor, Lisa Childers, *Globus Toolkit 4: Programming Java Services*, Morgan Kaufmann Publishers, (2006).

[9]. Hai Ly-Hoang, et al, *Proposal of software architecture for data mining in grid environment,* Proceedings of International Workshop on Advanced Computing and Applications, ACOMP'2007, Ho Chi Minh City, Vietnam, (2007).

[10]. A Min Tjoa, Ivan Janciak, Alexander Woehrer and Peter Brezany. *Providing an Integrated Framework for Knowledge Discovery on Computational Grids,* 5th International Conference on Knowledge Management, Graz, (2005).

[11]. SIMDAT website: www.simdat.org, (2006).

[12]. The EU Provenance project: www.gridprovenance.org, (2006).

[13]. Secure Data Grid website: www.cs.wright.edu/%7Eschung/SDG/sdg.htm, (2006).

[14]. Luis Ferreira, et al, *Introduction to Grid Computing with Globus*, IBM Corp, (2002).

[15]. Ian Foster, *Globus Toolkit Version 4: Software for Service-Oriented Systems*, IFIP International Federation for Information Processing, pp. 2-13, (2005).