A Grid Based Approach for Dynamic Integration and Access of Distributed and Heterogeneous Information Across an Enterprise

Swapnil S. Bagul, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: swapnil_bagul@infosys.com
Nilesh Ranade, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: nilesh_ranade@infosys.com
Aditya Sharma, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: aditya_sharma01@infosys.com
D. J. Acharya, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: dushyant_acharya@infosys.com
Sumit Kumar Bose, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: sumit_bose@infosys.com
Srikumar Krishnamoorthy, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: srikumar_k@infosys.com
Dheepak RA, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: dheepak_ra@infosys.com
Sweta Mistry, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: sweta_mistry@infosys.com
Shubhashis Sengupta, Software Engineering and Technology Labs, Infosys Technologies Ltd, Bangalore, India; E-mail: shubhashis_sengupta@infosys.com

ABSTRACT

Information within an enterprise is often scattered across various administrative domains and geographical time zones. Moreover, such information is maintained in different databases in heterogeneous formats serving varied needs of different set of people. Need is often felt to integrate this information, spread across the enterprise, for strategic decision-making on a real time basis. In this paper, we introduce GRADIENT – an Enterprise Information Integration solution based on service-oriented architecture for accessing distributed information across an enterprise and for solving integration challenges associated with data heterogeneity and geographical spread. GRADIENT utilizes a grid based approach to distribute computational load of queries and realizes enhanced performance in real-time data integration.

Keywords: Data-Grid, Enterprise Information Integration, Distributed Query Processing

1. INTRODUCTION

Enterprises data is preserved in heterogeneous data sources/formats and dispersed across multiple administrative domains or geographical locations. Need is often felt to integrate such diverse data sources for strategic decision making on a real time basis. However, data integration is a complex and time consuming task due to the heterogeneity and semantic disparity of the underlying data sources. There are varieties of approaches for solving this complex data integration problem and they can be broadly classified into two as: Extract, Transform and Load (ETL) and Enterprise Information Integration (EII). ETL based solutions allow the disparate data sources to be extracted transformed and loaded into data marts or data warehouses for query processing on the integrated data. But, the major drawback of using an ETL solution for integrating disparate data sources is the latency and complexity involved in extracting, cleaning and transforming the data and then moving(also referred as loading) the transformed data into data marts or data warehouses. On the other hand, EII based solutions allow creation of virtualized view of the disparate data sources leveraging the existing infrastructure with little or no movement of data. This data virtualization is achieved in a manner that is transparent to the user. The key challenge in such data virtualization solutions is the complexity involved in integration of the distributed and heterogeneous data sources in real-time.

Grid computing is an ensemble of heterogeneous computing resources for solving complex computation intensive tasks. Data grid is a manifestation of grid technology that helps to achieve virtualization of the data stored in multiple heterogeneous databases stored across multiple locations [1]. Additionally, data grids enable sharing of computational load across different machines. Since EII solutions deal with huge volumes of data during data integration, it may be useful to investigate the use of data grids for queries involving high computational requirements to achieve superior information integration benefits. So, the primary motivation for our work is to combine the EII and data grid technologies to achieve enhanced performance in real-time data integration.

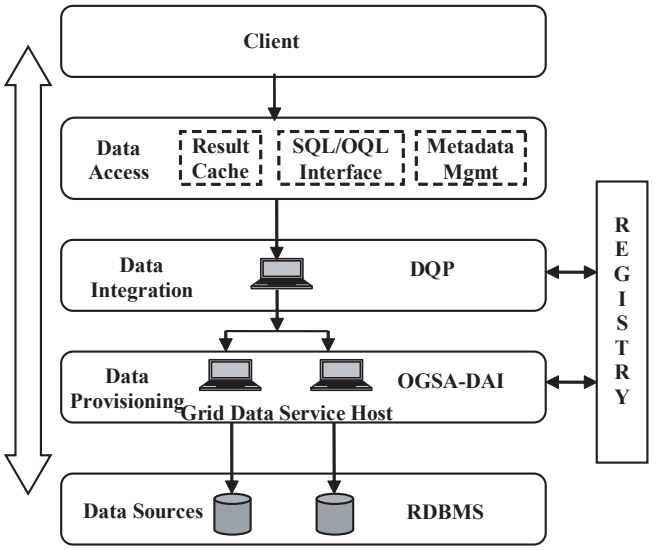
In this paper we present a grid based EII tool for accessing distributed information across an enterprise called GRADIENT (GRid Access of Distributed Information in the ENTerprise). GRADIENT is a service-oriented data grid solution that overcomes the limitations of ETL based data integration technologies and enables real-time data integration using data virtualization. GRADIENT allows the end user to seamlessly query disparate information sources using declarative query languages such as SQL. GRADIENT achieves greater scalability and performance using a distributed query processing engine. GRADIENT also addresses the semantic heterogeneity issues using RDBMS based metadata management system. Some of the salient contributions of this work include metadata management, advanced caching techniques and enhanced distributed query processing engine.

The rest of this paper is organized as follows. In section 2, we describe the proposed GRADIENT architecture. We deliberate on our preliminary experimental results in section 3 and provide concluding remarks in section 4.

2. GRADIENT ARCHITECTURE

GRADIENT is an EII solution that extends open source projects like the OGSA-DAI (*Open Grid Services Architecture – Data Access Integration*) [2, 3] and OGSA-DQP. Gradient offers a number of advanced features (like metadata management, advanced caching, enhanced distributed query processing engine) that were not supported in open source projects. Using Gradient as a Web Service or API, a client can invoke queries that involve join operations spanning multiple data sources using a single query without having the knowledge of the location of the underlying data sources and their formats. Figure 1, shows a high level architectural view of Gradient having three logical layers, namely, (1) data provisioning layer, (2) data integration layer, and (3) data access layer. We discuss the key features of each of these layers in detail in the following paragraphs.

Figure 1. High level architecture of GRADIENT



2.1 The Provisioning Layer

Provisioning layer provisions the data sources and exposes it as services. Gradient uses OGSA-DAI to expose disparate data sources as Grid Data Services (GDS). GDS accept perform documents (an XML document describing user queries) and parses and validates the query against the metadata extracted by GDS. GDS, then executes the query and constructs response XML document containing the query results. The OGSA-DAI has been extended to support metadata extraction of the data sources like Oracle, SQL Server, DB2 and POSTGRES. In future, we plan to extend it further to support other data sources.

2.2 The Integration Layer

The Integration layer in Gradient provides support for distributed query processing and is built on top of open source OGSA-DQP [3] project. The core distributed query processing engine of Gradient enables optimal sharing of the computational load intensive (e.g. join) queries. This allows for the parallel processing of a query using inter-operator and intra-operator parallelism. Since the database size usually exceeds terabytes, having different portions [4] of a query executed on different physical machines in parallel result in considerable improvement in query response times.

Gradient offers query processing and optimization support for declarative queries over a set of services that includes *databases services* and *computational services*. Database services use emerging standards of Grid Data Services (GDS) to provide a uniform and consistent access to different databases. Computational services are needed for performing query splitting and for executing different portions of the query on different computational nodes for achieving speed-ups. DQP is made up of two services:

I. *Grid Distributed Query Service (GDQS)*: GDQS, also called the co-coordinator, is responsible for (a) Retrieving and storing the metadata of each database (this is done only once during the installation of GDQS on the machine), (b) Creating the single node physical plan and then a parallel plan for a query through successive transformations using relational algebra and calculus, and (c) Scheduling the sub-plans of a parallel plan on the computational nodes. We extend GDQS by building upon the previous work of Polar* distributed query processor for grid [3]. The Polar* is implemented in OPTL-a database optimizer specification language defined by [5]. However, the query operators supported in the original version of Polar* were inadequate to handle the diversity of queries that a user may invoke from enterprise applications. We enhanced the functionality of Polar* to support various query operators such as or, not, in, like, aggregates and non-equijoins. These enhancements will allow the user to make more complex queries than were supported by the original version of DQP.
II. *Grid Query Evaluation Service (GQES)*: GQES, also called the evaluator, is used to execute the query sub-plans. Coordinator schedules query sub-plans on one or more instances of GQES based on decision made by the query optimizer. This allows sharing of the computational load since query processing tasks are often computation and memory intensive. In this context, the GQES is analogous to an idle computational node on a compute grid.

The schematic representation of a typical DQP environment is shown in figure 2. Nodes N_1 and N_2 hosts databases DB_1 and DB_2 respectively. The databases DB_1 and DB_2 are exposed using respective Grid Database Services (GDS). All the nodes also act as evaluators. Nodes N_3, N_4 and N_5 do not host any database and only act as evaluators for performing computation intensive tasks. Any query involving DB_1 and DB_2 will necessarily run on nodes N_1 and N_2 and can additionally employ nodes N_3, N_4 and N_5 for sharing the computational load of the queries.

2.3 The Access Layer

Access layer is the first point for the end user application to access the data exposed by the Gradient using standard SQL queries. In this layer, Gradient uses a metadata management service for input query parsing, query resolution and OQL generation. The GDS – a layer of abstraction that hides the heterogeneity of the underlying databases – provides a service oriented interface for extracting the metadata from the underlying data sources. The metadata which is stored as a relational database in a centralized location maintains all the information necessary for parallel query optimization. This central metadata repository is exposed as

Figure 2. Environment with DQP services

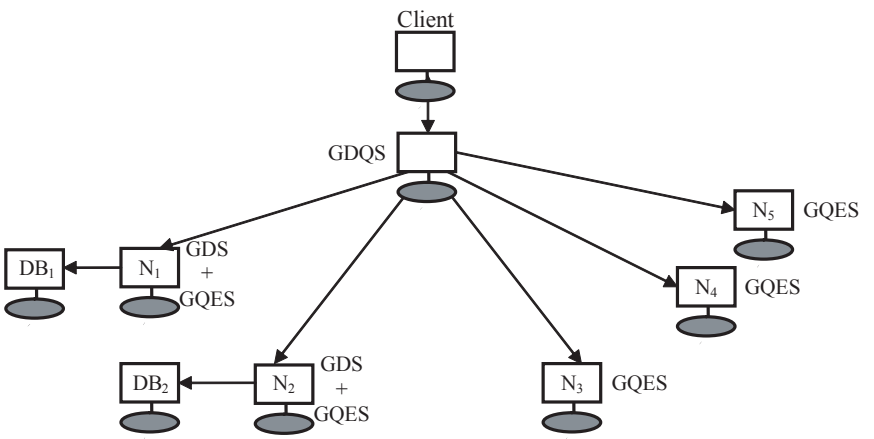
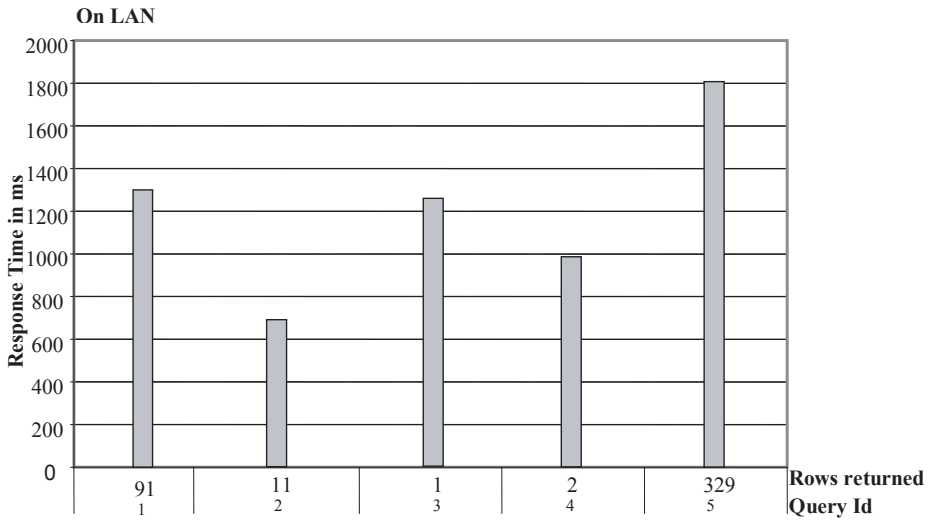


Table 1. Query types executed on GRADIENT

Query ID	Type of Query	Records Returned	No of records	Cross DB Join
1	Full Table Scan	91	91	DB2
2	Range Table Scan	11	34480	MYSQL
3	DB Equijoin	1	13278 * 91	DB2
4	Equijoin	2	34480 * 77	DB2 * ORACLE
5	Equijoin	329	34480 * 77	MYSQL * DB2

Figure 3. Performance Results for GRADIENT



a web service called the Global Metadata Service (GMS) and is responsible for much of the information virtualization talked about earlier. Gradient also employs a number of caching techniques such as data cache, query cache and metadata cache for improving the query response times.

3. EXPERIMENTAL RESULTS

We carry out extensive experimentation, with a Distributed Order Management System (DOMS) hosted on MYSQL, Oracle and DB2 for studying the performance of the Gradient system. GDQS is installed on a machine with Pentium 4, 2.8GHz processor and 1GB RAM. All the DQP evaluators were run on Pentium 4, 2.8GHz processor with 512MB RAM machines. We executed different types of queries ranging from a full table scan on a single database to cross database equi-joins. A complete list of queries used for the experimentation is provided in table 1.

Figure 3 shows the response time of the results for different queries executed in a LAN environment using Gradient.

4. CONCLUSION

The heterogeneity of the databases and the geographic dispersion of the data make it difficult to integrate the data and provide a transparent way of accessing this data by the user. In this paper, we presented an Enterprise Information Integration tool based on the service-oriented concepts and data grids, called GRADIENT. In particular, we elaborated on the metadata management, distributed query-processing and caching techniques implemented as a part of the solution. The Gradient is planned to support Metadata Synchronization, Adaptive Query Processing and Distributed Caching in the future.

REFERENCES

1. R.W. Moore and C. Baru, Virtualization service for data Grids, a book chapter in Grid computing: making Global infrastructure a reality, John Wiley and sons, 2003
2. K. Karasavvas, M. Antonioletti, M.P. Atkinson, N.P. Chue Hong, T. Sugden, A.C. Hume, M. Jackson, A. Krause, and C. Palansuriya. Introduction to OGSA-DAI Services. Pages: 1-12, Springer, Lecture Notes in Computer Science 3458, 2005.
3. M. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. Fernandes, A. Gounaris, and J. Smith. OGSA-DQP: A service-based distributed query processor for the Grid. In Simon J. Cox, editor, Proceedings of UK e-Science All Hands Meeting Nottingham. EPSRC, 24, 2003.
4. 2M. Tamer Özsu, Patrick Valduriez, Distributed and Parallel Database Systems, ACMComputing Surveys, 1996
5. L. Fegaras, and D. Maier, Optimizing Object Queries Using an Effective Calculus, ACM Transactions on Database Systems, Volume 25 , Issue 4, Pages: 457 – 516, 2000.