A Service-Oriented System to Support Data Integration on Data Grids

Anastasios Gounaris
University of Manchester
gounaris@cs.man.ac.uk

Carmela Comito
UNICAL
ccomito@deis.unical.it

Rizos Sakellariou
University of Manchester
rizos@cs.man.ac.uk

Domenico Talia
UNICAL
talia@deis.unical.it

Abstract

Data Grids provide transparent access to heterogeneous and autonomous data resources. The main contribution of this paper is the presentation of a data sharing system that (i) is tailored to data grids, (ii) supports well established and widely spread relational DBMSs, and (iii) adopts a hybrid architecture by relying on a peer model for query reformulation for retrieving semantically equivalent expressions, and on a wrapper-mediator integration model for accessing and querying distributed data sources. The system builds upon the infrastructure provided by the OGSA-DQP distributed query processor and the XMAP query reformulation algorithm. The paper discusses the implementation methodology, and also presents empirical evaluation results.

1. Introduction

The Grid has emerged as a promising infrastructure for the coordinated use and sharing of distributed resources in a dynamic manner [12], enabling the temporary pooling of resources to solve specific problems [13]. It does not refer only to resources such as CPU power, storage facilities and memory, but also to data sources; integrating, accessing and thus sharing multiple data sources is deemed as a key point for realizing the Grid vision and bridging the gap between that vision and the current state-of-the-art [13]. A Data Grid can include and provide transparent access to semantically related data resources that are maintained in different syntaxes, managed by different software systems, and are accessible through different protocols and interfaces. Due to this diversity in data resources, one of the most demanding issues in managing and sharing data on Grids is reconciliation of data heterogeneity.

Data Grids that rely on the coordinated sharing of and interaction across multiple autonomous database management systems play a key role not only in scientific initiatives but in many industrial scenarios as well; this fact has led to the production of vendor systems, such as Oracle10g and IBM DB2. At the level of Grid middleware infrastructure, two notable (and correlated) examples are the *OGSA Data Access and Integration* (OGSA-DAI) and the *OGSA Distributed Query Processor* (OGSA-DQP)[1] [3, 4] projects. These projects have moved toward a service-oriented architecture quite early in their lifecycle. OGSA-DAI exposes database management systems (including Oracle, MySQL, SQLServer, DB2, and so on) in a uniform way, whereas OGSA-DQP provides distributed query processing functionalities on top of OGSA-DAI, presenting the multiple databases as a single one. As such, OGSA-DQP can combine and integrate data from multiple data sources. To enhance performance, it employs parallel and adaptive query execution techniques [14]; nevertheless the semantic interpretation of the data rests with the user, and furthermore, OGSA-DQP does not address any schema integration requirements. Hence, to a significant extent, the integration facilities provided by OGSA-DAI and DQP are inadequate to meet the requirements of data grid real cases.

Data integration, in its own right, is one of the most persistent problems that the database and information management community has to deal with. Efficient techniques have been developed for and approaches have been devised to schema mediation languages, query answering algorithms, optimisation strategies, query execution policies, industrial development, and so on [17]. However, effective techniques for the generation and handling of semantic mappings are still in their infancy. The need for semantic correlation of data sources is particularly felt in Grid settings. To date, only few projects (e.g., [7, 6]) actually meet the schema-integration requirements that are necessary for establishing semantic connections among heterogeneous data sources. To this end, the use of the *XMAP* query reformulation algorithm for integrating heterogeneous data sources distributed over a Grid has been proposed [8]. Its aim is to develop a decentralized network of semantically related schemas, so that the formulation of semantically equivalent distributed queries over heterogeneous data sources is enabled. XMAP employs a decentralized point-to-point me-

[1] OGSA-DAI/DQP are publicly available in open source form from www.ogsadai.org.uk.

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

diation approach to connect different data sources based on schema mappings. For instance, if the user submits a query for authors of scientific papers of a certain kind to a certain database containing information about scientific publications, XMAP can return equivalent queries referring to similar databases, provided that the semantic mappings have been defined.

A comprehensive data sharing tool needs to include both distributed query processing and query reformulation functionality. Thus far, data sharing tools in distributed environments tend to build upon non-database management systems (DBMSs) or immature decentralized models, such as peer databases [18]. The main novelty of the work described in this paper is the presentation of a data sharing tool that (i) is tailored to data grids, (ii) supports well-established and widely spread relational DBMSs, and (iii) adopts a hybrid architecture by relying on a peer model for query reformulation for retrieving semantically equivalent expressions, and on a wrapper-mediator integration model for accessing and querying distributed data sources. Since the functionality offered by OGSA-DQP and XMAP is complementary, and provides an efficient basis on top of which comprehensive, Grid-enabled data sharing tools can be built, we discuss the design and implementation of a system combining the two aforementioned artefacts, along with their extended functionality. The result of our work is a unifying service-oriented infrastructure for distributed query processing and query reformulation driven by semantic connections, with a view to providing more complete access and integration services for data grids. An important feature is that, although the system incorporates OGSA-DQP and XMAP, the model it conforms to is generic, and enables the usage of any query reformulation or distributed query processing subsystem.

The prototype that has been developed realizes the Grid vision with regards to data management in two orthogonal ways. Firstly, it enables both query reformulation and distributed query processing across heterogeneous data sources through extensions to the original work of XMAP and OGSA-DQP, respectively. Second, it constitutes an example of how independent systems that seem incompatible at first glance, exposed as services, can work together. OGSA-DQP accepts queries in a subset of OQL (that is very close to simple SQL) and supports relational databases, whereas XMAP was initially designed for XPath queries over XML databases. In our system, this language mismatch has been addressed through the development of a mapping between a subset of XPath over the XML representation of the relational schemas into OQL. Our approach is limited, but it has proved sufficient for our environment.

This remainder of this paper is structured as follows. Section 2 discusses the motivation for the development of this prototype. Its main independent components, namely OGSA-DQP and XMAP are presented in Section 3. Section 4 deals with the architecture, the design decisions and the implementation details. Section 5 presents some experiments that aim at providing useful insight into the actual behavior of the system, and the overheads incurred by the hybrid architecture. Section 6 discusses the related work and Section 7 summarises the paper.

2. Motivation

Our experience with OGSA-DQP is that, although it provides the key functionality of executing distributed queries transparently to the user [3], its applicability is restricted because of two main reasons. First, the service oriented architecture and the SOAP-based communication protocol incur a high overhead. This drawback has been partly ameliorated through the development of adaptive techniques [14]. Second, as the users typically do not know enough information about the semantics of the data in the autonomous, third-party resources to which they are provided access, they find it difficult to formulate semantically correct queries that combine data from multiple sources. Several solutions may exist for the second problem. Our approach, which is the topic of this paper, is to make the following option available to the users: instead of writing a query across multiple databases to compose a query that refers to a single database, and the system, through the XMAP algorithm that has been integrated, to return equivalent queries that refer to data stored to other databases, and subsequently, to execute them automatically. Without such functionality the user would be forced first to understand the semantics of the data in all data sources, and then to compose a union query, whereas now a query to a single database is sufficient to trigger the query reformulation mechanism, which automatically constructs and evaluates the semantically equivalent queries.

3. Background

Before presenting the architecture of our prototype, we briefly describe the two systems it integrates, namely OGSA-DQP and XMAP.

3.1. The OGSA-DQP System

OGSA-DQP is an open source service-based Distributed Query Processor supporting the evaluation of queries over collections of potentially remote data access and analysis services. OGSA-DQP uses data services provided by OGSA-DAI to hide data source heterogeneities and ensure consistent access to data and metadata from any database resource. An OGSA-DAI data service exposes data service resources, which wrap autonomous DBMSs. The version of

OGSA-DQP used in our prototype builds upon the Globus Toolkit 4 WSRF infrastructure [9].

OGSA-DQP provides two types of services, *Grid Distributed Query Services (GDQSs)* and *Query Evaluation Services (QESs)*. The former are visible to end users, accept queries from them, construct and optimise the corresponding query plans and coordinate the query execution. Query evaluation services are hidden from the users, implement the query engine, interact with other services (such as OGSA-DAI services, ordinary Web Services and other evaluators), and are responsible for the execution of the query plans created by a GDQS. The interactions and functionality of OGSA-DQP services are described in detail in [4], whereas a more complete list of publications is provided at the download site.

The operations a data service resource can perform are called *activities*. For each activity to be called, there needs to be a separate, dedicated *activity element* in the perform document received by the data service resource. Activities are also the extensibility point of OGSA-DAI and DQP, i.e., additional functionalities are implemented as new activities.

3.2. The XMAP framework

The primary design goal of the XMAP framework is to develop a decentralized network of semantically related schemas that enables the formulation of queries over heterogeneous, distributed data sources. The XMAP framework abstracts from the underlying network infrastructure, it is modeled as a number of various autonomous nodes (that can be also referred to as sites, sources, peers, etc) which hold information, and which are linked to other nodes by means of mappings. Therefore, it can be extended at information nodes in any networked environment, and, as thus it can be seen as a set of network nodes connected to the Internet. More precisely, XMAP is composed of a collection $\mathcal{N}$ of *nodes* which are logically bound to XML data sources. That is, each data source D_n is represented by exactly one node n and, conversely, each node has access to a single data source, named *local data source*. Naturally, a *local schema* S_n is associated to this data source D_n. Data sources employ the XML data model and each source defines its own XML Schema. Each node also holds a collection of mappings M_n from its local schema to other foreign schemas. Finally, a node knows a list (also named *partial view* or, simply, *view*) of other nodes (called *neighbors*). These nodes are connected to each other through declarative mappings rules.

The XMAP integration [8] model is based on schema mappings to translate queries between different schemas. The goal of a schema mapping is to capture structural as well as terminological correspondences between schemas. Thus, in [8] , we propose a decentralized approach inspired by [18] where the mapping rules are established directly among source schemas without relying on a central mediator or a hierarchy of mediators. The specification of mappings is thus flexible and scalable: each source schema is directly connected to only a small number of other schemas. However, it remains reachable from all other schemas that belong to its transitive closure. In other words, the system supports two different kinds of mapping to connect schemas semantically: point-to-point mappings and transitive mappings. In transitive mappings, data sources are related through one or more "mediator schemas".

We address structural heterogeneity among XML data sources by associating paths in different schemas. Mappings are specified as path expressions that relate a specific element or attribute (together with its path) in the source schema to related elements or attributes in the destination schema. The mapping rules are specified in XML documents called XMAP documents. Each source schema in the framework is associated to an XMAP document containing all the mapping rules related to it.

The key issue of the XMAP framework is the XPath reformulation algorithm: when a query is posed over the schema of a node, the system will utilize data from any node that is transitively connected by semantic mappings, by chaining mappings, and reformulate the given query, expanding and translating it into appropriate queries over semantically related nodes. Every time the reformulation reaches a node that stores no redundant data, the appropriate query is posed on that node, and additional answers may be found. Therefore, beyond basic processing and communication facilities (exchanging messages with other nodes), nodes are supposed to be able to execute the XMAP query reformulation algorithm and to answer locally the queries they receive. We performed an extensive set of experiments to evaluate the performance and effectiveness of our approach. From such experiments we can realize that XMAP addresses the scalability concern scaling well with the number of participating nodes and guaranteeing quick production of reformulations, within few milliseconds even for the most demanding configurations.

In our architecture the reformulation algorithm has been re-engineered as a Web Service, referred to as *XMAPAlgorithm Web Service (XMAP-WS)*.

4 System Architecture

4.1. The hybrid model

A comprehensive data integration architecture needs to combine both the query reformulation and the query processing services. Our system offers a wrapper/mediator-based approach to integrate data sources, and adopts the XMAP decentralized mediator approach to handle seman-

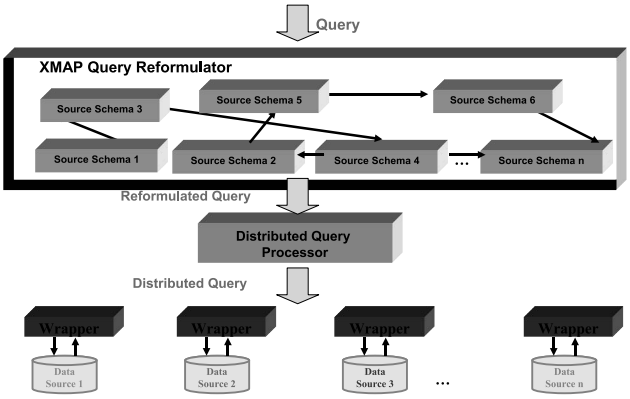

Figure 1. System Model.

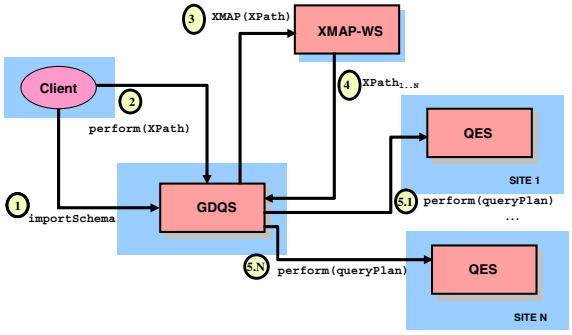

Figure 2. The high-level architecture. The arrows denote (remote) service calls.

tic heterogeneity over data sources, whereas syntactic heterogeneity is hidden behind OGSA-DAI wrappers. More precisely, the proposed framework is characterized by three core components: a *query reformulator engine*, a *distributed query processor*, and a *wrapper module* (see Figure 1).

The very general behavior of the developed system can be briefly outlined as follows. XMAP plays the role of the reformulation engine, OGSA-DQP is the distributed query processor with GDQS playing the role od mediator, whereas wrappers are provided by OGSA-DAI. The user query is handled by the reformulator engine that through the XMAP query reformulation algorithm produces zero, one or more reformulations of the original query. All the obtained reformulations (included the original query) are then processed by the DQP module that partitions each of such queries in several sub-queries to be executed in parallel. Then, each produced sub-query execution plan is processed by OGSA-DQP's QESs that access data sources through OGSA-DAI data service resources and produce the query result.

The model above can be implemented in several ways. Three main options include: (i) to incorporate the reformulation algorithm within the DQP module; (ii) to make the reformulation algorithm a stand-alone module that is called from within DQP; and (iii) to make the reformulation algorithm a stand-alone module that is called from a third module that acts as the bridge between DQP and the former module. The basic advantage of the first option is that the overheads due to inter-module communication are minimized at the expense of generality. In the third option, both DQP and query reformulation algorithms can be replaced and modified easily, i.e., the design is more generic, but the system becomes less efficient. In our system, we followed the second option that is the middle solution, as shown in Figure 2. In our architecture, the query reformulation algorithm is exposed as a completely independent service, so that any such algorithm can be plugged in and out of the system. However, this service is called from within OGSA-DQP, which has been extended accordingly, and as such, if we want to replace OGSA-DQP with another DQP system, we need to re-implement these extensions.

After having exposed the XMAP framework as a stand-alone WS, reusing it for other examples and in other settings does not pose significant problems. However, in order to integrate it with OGSA-DQP as in our case, two main technical issues had to be overcome. The first was to adhere to OGSA-DAI design principles, which entailed that the additional data integration functionality must be implemented in the form of an OGSA-DAI activity with all its complexities. The second main technical challenge stemmed from the fact that a solution to the language mismatch problem had to be developed. Both issues are discussed in the next subsection.

4.2. System Functionalities

The main features, that provide added value to stand-alone OGSA-DQP are *query reformulation* and *query transformation* that we have implemented in a new activity, called *XPathMappingActivity*. The schema of the new activity is shown in Figure 3. The *expression* element contains the submitted XPath query, whereas *ServiceLocation* contains the address of the web service to be contacted for the actual query reformulation, according to the architecture discussed previously. As such, each OGSA-DQP data service resource supports two main operations, one for OQL queries and one for XPath queries that are reformulated and translated into OQL (see Figure 4).

As mentioned before, query reformulation is enabled with the help of the XMAP-WS, the descriptor of which is shown in Figure 5. However, the reformulated queries returned by the XMAP-WS cannot be evaluated in their current form by OGSA-DQP, as the latter accepts only OQL queries. Thus a query transformation step is required. The policy for that is as follows. In general, the set of mean-

```
<?xml version="1.0" encoding="UTF-8"?> ...
 <xsd:schema
 <xsd:complexType name="XPathMappingType">
  <xsd:complexContent>
   <xsd:extension base="gds:ActivityType">
     <xsd:sequence>
       <xsd:element name="expression"
            minOccurs="1" maxOccurs="1">
         <xsd:complexType mixed="true">
           <xsd:complexContent>
            <xsd:extension base="gds:ActivityInputType"/>
           </xsd:complexContent>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="ServiceLocation"
            minOccurs="1" maxOccurs="1">
        <xsd:complexType mixed="true">
         <xsd:complexContent>
          <xsd:extension base="gds:ActivityInputType"/>
         </xsd:complexContent>
        </xsd:complexType>
       </xsd:element>
       <xsd:element name="webRowSetStream"
            minOccurs="1" maxOccurs="1">
        <xsd:complexType mixed="true">
         <xsd:complexContent>
          <xsd:extension base="gds:ActivityOutputType"/>
         </xsd:complexContent>
        </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 <xsd:element name="XPathMappingStatement"
              type="gds:XPathMappingType"
              substitutionGroup="gds:activity"/>
</xsd:schema>
```

Figure 3. The schema of the new activity.

```
<activityConfiguration> ...
 <activityMap>
  <activity name="XPathMappingStatement"
   implementation=
   "uk.org.ogsadai.dqp.gdqs.XPathMappingActivity"
   schema="xpath_mapping_statement.xsd"/>

  <activity name="oqlQueryStatement"
   implementation=
   "uk.org.ogsadai.dqp.gdqs.OQLQueryStatementActivity"
   schema="oql_query_statement.xsd"/>
 ...
 </activityMap>
</activityConfiguration>
```

Figure 4. Fragment of the activity configuration document of an OGSA-DQP data service resource.

```
<?xml version="1.0" encoding="UTF-8"?> <deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <globalConfiguration>
   <parameter name="adminPassword" value="admin"/>...
 </globalConfiguration>...
 <service name="XMAPAlgorithmService" provider="java:RPC">
   <parameter name="className" value="xmap.XMAPAlgorithm"/>
   <parameter name="allowedMethods" value="*"/>
 </service>...
</deployment>
```

Figure 5. Fragment of the WSDD of the XMAP-WS.

ingful XPath queries over the XML representation of the schema of relational databases supported by OGSA-DQP fits into the following template:

$$/database\_A[predicate\_A]/table\_A[predicate\_B]/column\_A$$

where

$$predicate\_A ::= table\_pred\_A[column\_pred\_A = value\_pred\_A], \text{ and}$$

$$predicate\_B ::= column\_pred\_B = value\_pred\_B$$

As such, the mapping to the `select, from, where` clauses of OQL is straightforward. *column_A* defines the `select` attribute, whereas *table_A, table_pred_A* populate the `from` clause. If *column_pred_A=value_pred_A, column_pred_B=value_pred_B* exist, they go into the `where` field.

The approach above is simple but effective; nevertheless two important observations are: firstly, it does not benefit from the full expressiveness of the XPath queries supported by the XMAP framework, and secondly, it requires the join conditions between tables *table_A, table_pred_A* to be inserted in a post-processing step. Such a transformation falls in an active research area (e.g., [11, 5]), and is implemented as an additional component within the query compiler.

The interactions of the services are as follows (see also Figure 2):

1. The user contacts the *GDQS* through a client application and requests a view of the schema for each database he/she is interested in. The schema is returned in XML with the elements *database*, *table* and *column* forming an hierarchy. At this point, there is no assumption that the user has an *a priori* knowledge of the semantics of this and the semantically-related databases.

2. Based on the retrieved schema, the user composes an XPath query, which is sent to the *GDQS*, and not directly to the corresponding database service, following the OGSA-DQP approach. An example of an XPath query is *"/database[@dbname=IEEE]/ table[@name=Publications]/column[@name=author]"*, or, in a simpler form, */IEEE/Publications/author*.

3. *GDQS* contacts a *XMAP-WS* service, which encapsulates the XMAP algorithm.

4. The *XMAP-WS* retrieves the locally stored mapping schema, which contains the mapping information that links the paths in the submitted query with paths referring to other databases. It returns a set of *n* queries that all return results that are semantically similar to those of the initial query.

5. For each of the results of the previous step, the *GDQS*

transforms the XPath expression in OQL, parses, optimises, schedules and compiles a query execution plan. The resulting query execution plan is sent to the corresponding *QES*, which returns the results asynchronously, after contacting the local database via an OGSA-DAI service.

In order to make clearer the concepts introduced, in the following we describe an example of use of the overall prototype.

4.3. An Example

Suppose that several publishers (such as IEEE and ACM) make their databases accessible from anywhere, and when a user wants to retrieve the publications of a specific author in a certain year it is sufficient to submit a query only to one of the databases, with the system being responsible for querying the remaining databases. Or, when querying a museum database for artifacts of a specific artist, the system is able to return similar results from other museum databases as well. Supporting scenarios like that, coupled with the emergence of Grid technologies, have motivated the architecture and the system of this paper.

Considering the publisher example, we used a modified real-world data set to validate our prototype, the *DB-Research* data set that has been created for the Piazza system [18][2]. The data set is based on data available on web sites concerning research in the database field. It includes the schemas corresponding to the structure and terminology of 19 such web sites (such as DBLP, CiteSeer, ACM Digital Library, and a few university sites). On the basis of these schemas, we have defined XMAP mappings between the schemas that are semantically similar. More precisely, for each source schema we have defined mapping rules toward, on average, three other source schemas.

Figure 6 shows some of the database schemas in the considered data set. The figure presents two self-explanatory views: one hierarchical (for native XML databases), and one tabular (for object-relational DBMSs). In OGSA-DQP, the table schemas are retrieved and exposed in the form of XML documents. We exploit such representation of database schemas in order to integrate XMAP within the OGSA-DQP. In fact, users can have a view of the XML representation of relational schemas, so instead of translating the tabular view in a XML one, they can directly query the XML representation by using the XPath query language.

Examples of semantic mappings among the databases are illustrated in Figure 7: here, the column "title" of the table "paper" of the database "ACM" is mapped to the column "paper" of the table "proceedings" of the database "DBLP".

[2]The usage of this data set has been kindly authorized by Igor Tatarinov.

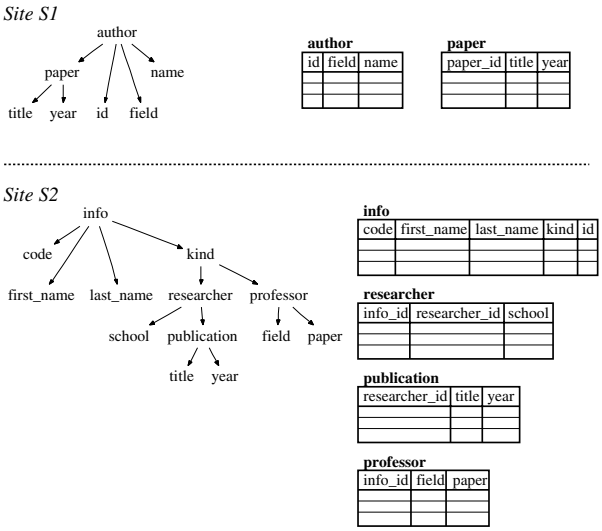

Figure 6. The example schemas.

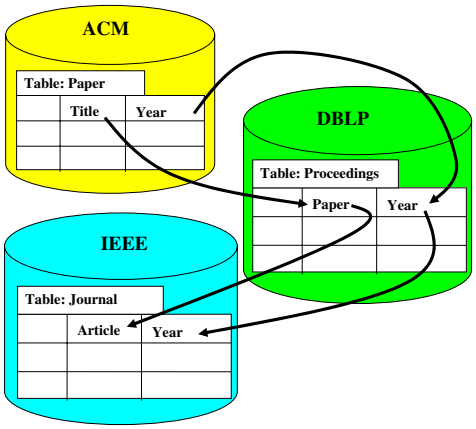

Figure 7. Example of mappings.

The latter is mapped to the column "article" of the "journal" table of "IEEE" data source. This information resides in the mapping document associated with the XMAP Web Service. Note that it is not necessary to map the ACM to the IEEE database directly. The XMAP mappings need to capture the semantic relationships between the data fields in different databases , including the primary and foreign keys. This can be done as shown in Figures 8, 9.

Let us suppose a user wants to find the *title* of the *paper* published in the *year* 2000. To this aim the following tasks are performed:

1. The user contacts the *GDQS* and requests a view of the schema of the ACM database, which has a table called "paper", with columns "year" and "title".

2. Based on the retrieved schema, the user composes the following XPath query formulated over the

```
<Mapping>
 <sourceSchema>ACM</sourceSchema>
 <Rule cardinality="Mapping1-1">
  <destinationSchema>dblp</destinationSchema>
  <sourcePath>/acm/paper/title</sourcePath>
  <destinationPath>/dblp/proceedings/paper</destinationPath>
 </Rule>
 <Rule cardinality="Mapping1-1">
  <destinationSchema>dblp</destinationSchema>
  <sourcePath>/acm/paper/year</sourcePath>
  <destinationPath>/dblp/proceedings/year</destinationPath>
 </Rule>...
</Mapping>
```

Figure 8. Fragment of the ACM XMAP document.

```
<Mapping>
 <sourceSchema>DBLP</sourceSchema>
 <Rule cardinality="Mapping1-1">
  <destinationSchema>ieee</destinationSchema>
  <sourcePath>/dblp/proceedings/paper</sourcePath>
  <destinationPath>/ieee/journal/article</destinationPath>
 </Rule>
 <Rule cardinality="Mapping1-1">
  <destinationSchema>ieee</destinationSchema>
  <sourcePath>/dblp/proceedings/year</sourcePath>
  <destinationPath>/ieee/journal/year</destinationPath>
 </Rule>...
</Mapping>
```

Figure 9. Fragment of the $DBLP$ XMAP document.

ACM schema: $Q_{ACM} = /acm/paper[year = \text{"2000"}]/title$, which is sent to the *GDQS*.

3. *GDQS* contacts a *XMAP-WS* service, which encapsulates the XMAP algorithm.

4. The *XMAP-WS* service retrieves the locally stored ACM mapping schema (see Figure 8). More specifically, the schema ACM is connected to the schema $DBLP$ and thus we can rewrite the query Q_{ACM} over the schema $DBLP$ obtaining the query $Q_{DBLP} = /dblp/proceedings[year = \text{"2000"}]/paper$. Then exploiting the semantic mappings concerning the schema $DBLP$ (see Figure 9) we obtain the query $Q_{IEEE} = /ieee/journal[year = \text{"2000"}]/article$.

5. For each of the results of the previous step, the *GDQS* transforms the XPath expressions in OQL producing the following corresponding OQL queries: `select a.title from a in paper where year="2000";` `select d.paper from d in proceedings where year="2000";` and `select i.article from journal where year="200".` OGSA-DQP automatically detects the remote databases for each query; in this case the ACM database for the first one, the $DBLP$ for the second one, and the $IEEE$ for the third.

#T	QRT	XRT	Overall	OH (%)
2	1485(18.6)	2111(123.3)	3688(139.1)	91.8(2.49%)
4	2727(15.5)	1971(5.1)	4785(16.9)	86.2(1.8%)
8	5248(96.8)	2045(15)	7380(105.6)	86.6(1.18%)
16	11256(1185))	2034(1.58)	13377(1185)	86.6(0.64%)

Table 1. Response times (in msecs) and overheads, when the calls to the GDQS and XMAP services are local, and the average size of results per query is 10.

#T	QRT	XRT	Overall	OH (%)
2	1726(102.8)	2056(17.8)	3872(123)	90.2(2.33%)
4	2948(130.8)	1977(2.65)	5010(132.2)	85.4(1.7%)
8	5613(129)	2031 (7.2)	7730 (123.7)	86.6(1.12%)
16	11670(566)	2964(16.5)	13824(559)	89.2(0.65%)

Table 2. Response times (in msecs) and overheads, when the calls to the GDQS and XMAP services are local, and the average size of results per query is 100.

#T	QRT	XRT	Overall	OH (%)
2	9139(398)	6177(91.3)	16390(537)	1074(6.55%)
4	17970(738)	5971(171)	25032(1219)	1092(4.36%)
8	34291(1075)	6133(192)	42233(1111)	1809(4.29%)
16	72300(2288)	6269(103)	79787(2338)	1218(1.53%)

Table 3. Response times (in msecs) and overheads, when the calls to the GDQS and XMAP services are remote, and the average size of results per query is 10.

#T	QRT	XRT	Overall	OH (%)
2	9155(191)	5956(55)	16069(146)	958(5.96%)
4	18308(617)	5947(103)	25344(755)	1089(4.3%)
8	38269(656)	6247(212)	46487(726)	1971(4.24%)
16	75266(1495)	6177(89.3)	82609(1476)	1166(1.41%)

Table 4. Response times (in msecs) and overheads, when the calls to the GDQS and XMAP services are remote, and the average size of results per query is 100.

5. Experimental Evaluation

The experiments presented at this section aim at providing useful insights into the behavior of the system, focusing on the overhead added because of the hybrid architecture presented previously. For the behavior of individual components the interested reader may refer to works such as [10, 8].

In more detail, we are interested in measuring the over-

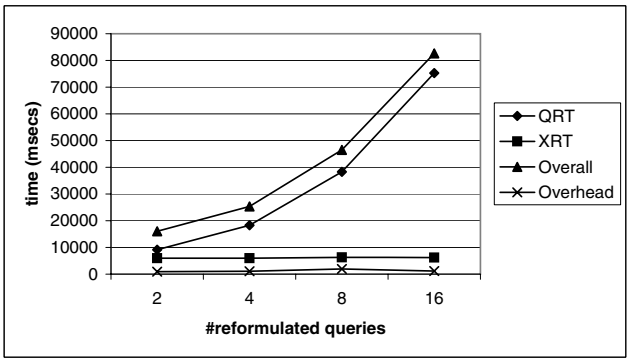

Figure 10. Response times when the service calls are remote and the average size of results per query is 100.

head incurred by the integrated architecture; in establishing whether there is any varying property of the input queries that the overhead depends on; and finally, in establishing whether the overheads still allow query evaluation to benefit from parallelism as it is the case with stand-alone OGSA-DQP [16]. In the experiment, the DQP server was a Fedore Core 4 machine with 1GB memory size and CPU Intel Pentium 4 at 3.20GHz. The metrics that are of interest in the experiments include: (i) the number of reformulated and transformed queries to be executed (#T); (ii) the response time for executing the complete set of these queries (QRT); (iii) the response time to call the XMAP Web Service and retrieve for a given XPath query the set of its equivalent queries along with their OQL transformations (XRT); (iv) the overall response time at the client, which includes QRT and XRT, aling with any overhead incurred by their linking; and (v) the overhead, which is the difference of the overall response time with the sum of QRT and XRT (OH).

Tables 1 and 2 present the measurements taken when all service calls are local (i.e., there is no network communication cost involved) and the average result size of each executed query is 10 and 100 tuples, respectively. The times in columns "QRT", "XRT", and "overall" are the average times of five runs in milliseconds, and, inside the parentheses, the standard deviation is shown. All the queries are simple Select-From-Where expressions scanning a single table and they retrieve from store 10-100 entries each time. The semantic mappings reside on a single XMAP Web Service. The overhead column "OH" shows the absolute overhead in milliseconds. Inside the parentheses is its percentage in the overall response time. From the tables we can see that the QRT cost increases linearly with the number of executed queries (as expected), whereas both the cost to reformulate and transform queries, and the overhead remain stable. The XRT is around 2 seconds, and the overhead is less than a tenth of a second. Moreover, the overhead incurred can be deemed as negligible, as even when there are just two small queries (i.e., retrieving only 10 tuples each), the overhead does not exceed the 2.5 %.

Similar observations can be made when the service calls are remote, as shown in Tables 3 and 4. In this experiment the client resides in continental Europe (and in Greece in particular), where are the other services are on Grid nodes in Manchester in the UK. Due to the network cost, the response times are higher, but the pattern of the previous setting is the same: linear increase of QRT and overall time, and stable XRT and overhead. The former is 6 seconds while the latter is 1 second, approximately. This is better illustrated in Figure 10, as well.

The conclusions that can be drawn are twofold: firstly, there is no single property of the input queries (such as the result size or the number of database entries retrieved) that the overhead depends on. The cost to reformulate and translate queries, and the system overhead are both stable, which entails that if we are able to estimate (with the help of a cost model, or based on previous experience) the execution time of queries, we can find the overall execution time by adding to the query execution cost the flat overhead. Secondly, the overheads still allow query evaluation to benefit from parallelism. We can see from the values of the tables that, apart of the extreme case that there is a very small number of queries and the service calls are local, the dominant cost is the QRT. This is important, as it is the only cost that can be reduced by employing parallel execution techniques. In other words, after the extensions added to OGSA-DQP to integrate the query reformulation functionality, it is still the case that parallelism can yield performance benefits.

6. Related Work

To the best of our knowledge, there are only few works designed to provide schema-integration in Grids. The most notable ones are *Hyper* [7] and *GDMS* [6]. Both systems are based on an approach similar to ours, i.e., to build data integration services by extending the reference implementation of OGSA-DAI. The *Grid Data Mediation Service* (GDMS) is part of the GridMiner project [1] and uses a wrapper/mediator approach based on a global schema. GDMS presents heterogeneous, distributed data sources as one logical virtual data source in the form of an OGSA-DAI service. The main difference from our work is that it relies on the existence of a global schema, which is not that realistic in Grids. *Hyper* is a framework that integrates relational data in P2P systems built on Grid infrastructures. As in other P2P integration systems, the integration is achieved without using any hierarchical structure for establishing mappings among the autonomous peers. In

that framework, the authors use a simple relational language for expressing both the schemas and the mappings. Our integration model follows an approach not based on a hierarchical structure as well, however it focuses on XML data sources and is based on schema-mappings that associate paths in different schemas. Finally, semantic mapping across relational databases coupled with a global-as-view approach is investigated in the context of the SASF project [2]. The notion of peer to peer semantic mappings appears also in a non-grid setting in [18]. An earlier version of this work is described in [15].

7. Summary

The contribution of this work is the proposal of a unifying architecture and of an approach that combines a semantic data integration methodology with existing services for querying grid-enabled distributed databases. This architecture is used for providing an enhanced, data integration-enabled service middleware for data grids. The architecture employs a wrapper-mediator approach for distributed query processing across autonomous databases exposed as Grid resources, and a decentralized model for establishing semantic connections between such databases. The instantiation of this architecture is service-based and builds upon two existing artefacts, namely OGSA-DQP for distributed query processing, and XMAP for semantic query reformulation. The paper, apart from describing the generic model and the system developed, presents evaluation results that provide insights into the actual behavior of the prototype. The results show that the additional cost is both stable and relatively low, which renders the practical application of the proposal appealing.

This work can be extended in various ways. Directions for future work include larger scale experiments both in terms of the data volume size and the number of the machines used, experiments with simultaneous queries, and evaluation of queries in XPath, if this functionality becomes available from OGSA-DAI/DQP.

Acknowledgement: This work has been supported by the EU-funded CoreGrid Network of Excellence project through grant FP6-004265.

References

[1] GridMiner, http://www.gridminer.org/.

[2] SASF: service-based approach to schema federation, http://sasf.grid.leena34.com/.

[3] M. N. Alpdemir, A. Mukherjee, A. Gounaris, N. W. Paton, P. Watson, A. A. A. Fernandes, and D. J. Fitzgerald. OGSA-DQP: A service for distributed querying on the grid. In *Proc. of EDBT 2004*, pages 858–861, 2004.

[4] M. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. A. Fernandes, A. Gounaris, and J. Smith. Service-based distributed querying on the grid. In *Proc. of ICSOC 2003*, pages 467–482. Springer, 2003.

[5] K. S. Beyer, R. Cochrane, V. Josifovski, J. Kleewein, G. Lapis, G. M. Lohman, B. Lyle, F. Ozcan, H. Pirahesh, N. Seemann, T. C. Truong, B. V. der Linden, B. Vickery, and C. Zhang. System rx: One part relational, one part xml. In *SIGMOD Conference 2005*, pages 347–358, 2005.

[6] P. Brezany, A. Woehrer, and A. M. Tjoa. Novel mediator architectures for grid information systems. *Journal for Future Generation Computer Systems - Grid Computing: Theory, Methods and Applications.*, 21(1):107–114, 2005.

[7] D. Calvanese, G. D. Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. Hyper: A framework for peer-to-peer data integration on grids. In *Proc. of the Int. Conference on Semantics of a Networked World: Semantics for Grid Databases (ICSNW 2004)*, volume 3226 of *Lecture Notes in Computer Science*, pages 144–157, 2004.

[8] C. Comito and D. Talia. XML Data Integration in OGSA Grids. In *1st Int. Workshop on Data Management in Grids (DMG)*, pages 4–15, 2005.

[9] K. Czajkowski et al. The WS-resource framework version 1.0. The Globus Alliance, Draft, Mar. 2004. http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

[10] B. Dobrzelecki, M. Antonioletti, J. Schopf, A. Hume, M. Atkinson, N. C. Hong, M. Jackson, K. Karasavvas, A. Krause, M. Parsons, T. Sugden, and E. Theocharopoulos. Profiling OGSA-DAI Performance for Common Use Patterns. In *Proceedings of the UK e-Science All Hands Meeting*, 2006.

[11] W. Fan, J. X. Yu, H. Lu, and J. Lu. Query Translation from XPath to SQL in the Presence of Recursive DTDs. In *VLDB Conference 2005*, 2005.

[12] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Intl. J. Supercomputer Applications*, 15(3), 2001.

[13] C. A. Goble and D. D. Roure. The semantic grid: Myth busting and bridge building. In *ECAI*, pages 1129–1135, 2004.

[14] A. Gounaris, N. W. Paton, R. Sakellariou, A. A. A. Fernandes, J. Smith, and P. Watson. Practical adaptation to changing resources in grid query processing. In *ICDE*, page 165, 2006.

[15] A. Gounaris, R. Sakellariou, C. Comito, and D. Talia. Service choreography for data integration on the grid. In *Knowledge and Data Management in Grids, CoreGrid Springer Series, Vol. 1*, 2007.

[16] A. Gounaris, R. Sakellariou, N. W. Paton, and A. A. A. Fernandes. A novel approach to resource scheduling for parallel query processing on computational grids. *Distributed and Parallel Databases*, 19(2-3):87–106, 2006.

[17] A. Y. Halevy. Data integration: A status report. In *BTW*, pages 24–29, 2003.

[18] Y. Halevy, G. Ives, D. Suciu, and I. Tatarinov. Schema mediation for large-scale semantic data sharing. *The VLDB Journal*, 14(1):68–83, 2005.