Towards a Novel Metadata Information Service for Distributed Data Management

Alexander Woehrer and Peter Brezany
Institute of Scientific Computing, University of Vienna, Vienna, AUSTRIA
Email: {woehrer, brezany}@par.univie.ac.at

Abstract— **The trend in Grid computing towards more data intensive applications, accessing more and more relational databases and requiring advanced integration of second-hand and publicly available data sources, is still upstanding. Rich metadata information about these data sources plays a vital role for efficient distributed data management. There is a lack of service oriented monitoring tools providing a rich set of metadata for data sources, in contrast to the established usage of monitoring and metadata information tools for network and computational Grid resources.**

We allow the improved integration and exploitation of data sources in service-oriented architectures by providing a service oriented monitoring tool, named DSMON, for them – initially focusing on relational databases. Our approach supports coarse and fine grained information about heterogeneous relational databases via a uniform interface and provides a homogeneous view on the available metadata. This paper presents novel usage scenarios needing a monitoring service as well as important requirements to be fulfilled by such a service. Our research prototype is not bound to a specific data access middleware nor necessarily tightly coupled with other components. Therefore it can play a vital role in any service oriented infrastructure targeted towards advanced data access, integration and management. The functionality and performance of the DSMON prototype is demonstrated for commonly used relational databases such as MySQL, PostGRE and Oracle. DSMON is an essential component of a more general metadata information service, called DSMIS, for data sources providing advanced funtionalities like prediction of the future behavior of a data source and external metadata updates. We introduce its overall architecture and discuss its novel functionalities.

Index Terms— **relational database, monitoring service, data management, metadata information service**

I. INTRODUCTION

The way e-Science is working on the Grid with data is different from traditional approaches – from gathering and analyzing your own data towards using second-hand, publicly available data sources and synthetic data [1]. This new approach requires finding relevant data sources for a certain application domain on demand as well as providing efficient integrated access to them [2]. During the data integration the selection of suitable candidates among similar sources and the exclusion of not required ones plays a vital role for efficient data integration on the Grid - due to the high data access and transportation costs [3]. Additionally, the importance of databases - most relational database management systems (RDMBS) - in accessing data on the Grid is constantly increasing. Taking into account the highly volatile Grid environment, adaptive query processing (AQP) [4] plays an important role. The involved process of collecting information regarding the current (sometimes also the past) status of a distributed system like the Grid is known as monitoring [5]. A good overview about available systems to monitor Grid resources, focusing on computational and network resources, can be found in [6]. Metadata [7], associated descriptions to digital resources of diverse kind, is supposed to serve as foundation for advanced services in various application domains, e.g. transparent delivery of information in bioinformatics [8]. The research effort described in this paper fills this monitoring and metadata gap for relational databases and gives an outlook on how to extend its concepts and functionalities for other sources, e.g. XML, as well. Data source related monitoring information is going to have a similar impact for efficient data access and integration on the Grid as the already utilized resource monitoring has, e.g. for fault tolerance and job scheduling (see [6]), and is also crucial for various areas related to distributed data management. We address this information need by providing a uniform interface towards the metadata of heterogeneous relational data sources. Our approach supports a homogeneous view on data access related and data related monitoring information in various granularities. We have evaluated our approach by implementing a research prototype supporting commonly used relational databases (MySQL, PostGRE and Oracle). The remaining part of the paper is organized as follows. Section II describes scenarios in various application areas requiring a service oriented monitoring tool for relational databases on the Grid. Requirements to be fulfilled by such a service are introduced in Section III. The architecture of our service is introduced in Section IV, followed by insights into its implementation in Section V. Performance issues and results are discussed in Section VI. We discuss a future work agenda towards an advanced metadata information service for data sources named DSMIS in Section VII, review related work in Section VIII and close the paper with brief conclusions in Section IX.

This paper is based on "A Monitoring Service for Relational Databases to Support Advanced Data Integration on the Grid," by A. Woehrer and P. Brezany, which appeared in the Proceedings of the First International Conference on Complex, Intelligent and Software Intesive Systems (CISIS'07), Vienna, Austria, April 2007. © 2007 IEEE.

This work is being supported by the research projects *SemDIG* (funded by the Austrian FIT-IT initiative of the Federal Ministry of Transport, Innovation and Technology) and the *Austrian Grid Project* (funded by the Austrian BMBWK).

II. Usage Scenarios

The following scenarios for different areas of advanced data management and integration in a service oriented architecture [9] show the urgent need for a monitoring and metadata information service for data sources.

A. Query Optimization

The first scenario, shown in Figure 1, demonstrates the support of our monitoring component for an initial distributed query optimization phase. A query planner produces the given query for our example data source.

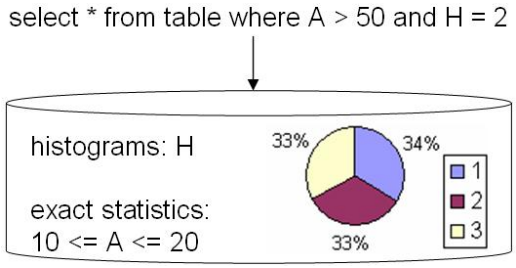

Figure 1. Query optimization scenario. Different kinds of metadata is available in the data source and should be used for efficient access [10].

Without our monitoring component, this would lead in our example, lets assume the table holds 1.000.000 rows and the default selectivity is set to 0.5, to the following row estimate: $rows = (0.5 * 0.5) * 1.000.000 = 250.000$. Figure 1 depicts that the database provides histograms for column H (three values occurring nearly equally often) and exact data statistics for column A (values in it between 10 and 20). If the histogram information for column H inside the database is made available to the external query optimizer via our monitoring component, the row estimate changes to: $rows = (0.33 * 0.5) * 1.000.000 = 165.000$. The example histogram information used is shown in Figure 6. This is significantly lower than before. In [11], it is shown how propagation of errors affects the quality of those estimates, which is very important for Grid environments with its huge and complex query plans. For such plans, the parallelization/distribution costs have to be taken into account as well. These costs are high if ratio between estimated number of rows and its average size is small. Looking at our example, it makes a huge difference if the ratio is $250.000rows/100bytes = 2.500$ (100 bytes default row size value of Oracle 10g) versus $250.000rows/16.000bytes = 15.6$ (real average row size due to a requested BLOB column). Providing also exact data statistics for column A to the query optimizer allows it to refute the data source completely from the query plan, as the statistics state that values in column A are between 10 and 20 and will therefore not return any rows for the given query. The number of sources to be included in a distributed query plan can so be minimized as well as the operator distribution improved. Similar information about candidate data sources as centralized ones have, like histograms [12] and exact data statistics [13], might be used by a distributed query processor, e.g. OGSA-DQP [14]. In [15], the importance of data statistics towards query optimization is described in more detail. Also for non-relational data sources, e.g. XML [16] or files [17], metadata plays an important role for efficient data access. Another interesting aspect of using monitoring components for query optimization is the possibility to combine the data related monitoring information of the main source with the information of its replicas. By this, the workload of gathering histograms and exact data statistics can be divided among them.

B. Adaptive Query Processing

The next scenario shown in Figure 2 represents another usage area, namely adaptive query processing (AQP). A query processing system is defined to be adaptive if it receives information from its environment and determines its behavior according to that information in an iterative manner. The three semantically distinct phases of adaptive query processing are [18]: monitoring, assessment and response. The execution of a plan by a query engine together with its execution environment are monitored, then an assessment is made relating to the progress of the execution and changes in the environment, depending on which a response may be carried out that affects the continuing evaluation of the query. The decisions made in the assessment component are based on the available monitoring information, which is currently focused on the computational/network resources and the progress information of the query execution [18]. In the given example we have a set of two databases, where one is the replica of the other. The adaptive query processing (AQP) component subscribes during the query setup phase to data source index changes of replicas of the currently used master data source (decided by a query optimizer in an earlier step). Changes of available indexes for the replica database are detected by the monitoring component, which triggers some notification mechanism [19] and informs the AQP system, where the current situation is investigated by an assessment component. In our example, the decision is made to switch for a given query from the current data source (no usable index for that query) to the replica data source (usable index) to improve query performance, as the new index will allow to speedup access. During the query processing the data can be analyzed on-the-fly, as done in [20], and

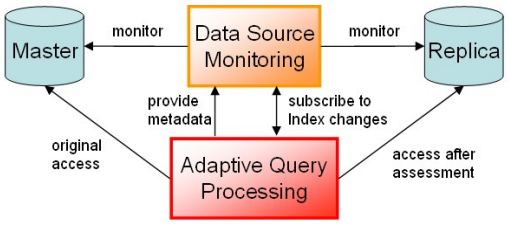

Figure 2. Adaptive query processing scenario. The instrumented query engine can be consumer (index changes) and producer (metadata updates) of monitoring information.

provide additional metadata input towards the data source monitoring infrastructure.

C. Data Management

In the third scenario, pictured in Figure 3, our monitoring component is used for data integration management. All data sources to be accessed by the Data Integration service have to be registered [21] and semantically annotated [22] via a Data Source Registration service for seamless semantic data integration. If the data source changes, for example one column is dropped, the registered information gets corrupted and therefore the semantic integration broken. With a data source monitoring component in place, the Data Source Registration service subscribes to schema changes of registered data sources. When it receives a notification, actions can be taken to cope with the change; either initiating a registration-update procedure or some automatic conflict resolution, e.g. by removal of the annotation for the deleted column.

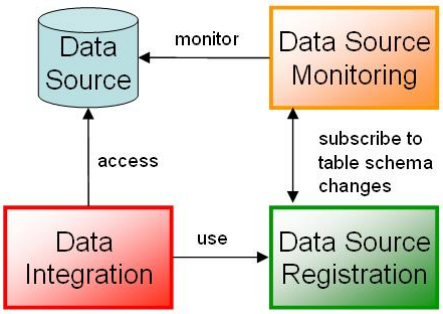

Figure 3. Data integration management scenario [10].

III. Requirements

In order to be applicable to the already introduced scenarios and other data access and integration related areas, a data source monitoring component has to fulfill the following requirements:

Support different monitoring granularities. The various consumers of the monitoring information will need only parts of the whole information provided. A health-status service might just be interested in the connection time to see if a certain data source is online while a data integration service will be interested in the attributes of a certain table or database. The interface has to support coarse grained (database level) as well as fine grained (table and/or column level) monitoring information.

Customizable to various needs. A relational database management system exposed on the Grid can host multiple databases with a broad variety of schemas not relevant (also for security reasons) to the outside world. Specific user tables are needed, while internal system tables are not. The monitoring component has to support mechanisms to define which database, schema and tables to include or exclude from the monitoring process in a flexible manner.

Little cumbering of the target data source. The monitoring process is an important, additional overhead task for a data source. It shouldn't flood the data source with requests in a way that it affects day-to-day business. This implies to keep the requests as simple as possible to gather the needed monitoring information as well as finding a suitable frequency for this process.

Support push concept rather than polling one. A consumer of the monitoring information shouldn't be forced to continuously poll our service if something has changed. Rather, the client should have the ability to let the service know in what type of changes he is interested (called subscription) and then be notified by the service. The interface has to support some kind of notification mechanisms [19].

Virtualization. Available metadata information about data sources are highly heterogeneous in terms of how to get them and what they contain. The monitoring service has to provide a uniform interface to this information as well as a homogeneous view on them.

Loosly coupled. The component is neither tightly coupled to a specific data access middleware nor necessarily tightly coupled with other components.

Be a sink for monitoring information. In a monitoring infrastructure, some service (the metadata information service named DSMIS in Figure 10) should provide an interface that allows that this service is the sink for monitoring information about a target data source.

IV. Architecture

A *data source monitoring (DSMON)* service provides metadata information of multiple heterogeneous databases via a uniform interface. Supported metadata about a data source is shown in Figure 4. Coarse grained information (in rounded boxes, representing metadata at the database level) and fine grained information (in square boxes, representing metadata on table level) is provided. A client accesses metadata information either ones (pull mechanism) when needed or subscribes to topics of interest and gets continuously informed about changes via notifications (push mechanism).

The current DSMON service, targeted towards relational databases, provides an interface to two kinds of monitoring information, namely *data access related* and *data related*, which are described in more detail in the following paragraphs.

A. Data access related

This category provides information about the access quality of a data source. An important indicator for the current workload of a relational database is the connection time - especially when the monitoring component is near to the source and the current network state does not tamper this value. The available indexes on a table and its logical schema are the second important part of the data access related monitoring information. The logical schema information contains the table name, column names and their types and the primary key information.

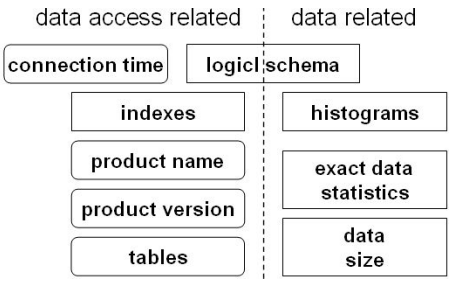

Figure 4. Metadata provided for a database (rounded) and its tables (square).

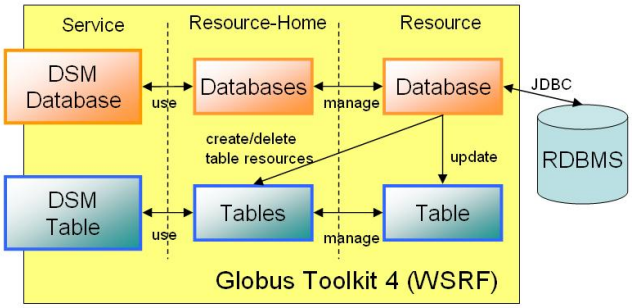

Figure 5. Implementation view of DSMON [10].

The index metadata includes the target columns of the index, the number of distinct keys, the overall size, index type (unique or not) and the sort direction.

B. Data related

This category contains information about the data quality and data size of a data source. As we are dealing mostly with autonomous data sources attached to the Grid, we are (normally) not allowed to change them. Nevertheless, we should expose as much suitable information about a data source as we can get to support advanced data integration and management. Most relational databases provide some kind of data statistics for internal query optimization usage in their system tables. Histograms are a good example for such information available for free. Histograms are especially useful where the data is not evenly distributed and shows a skewed distribution. The cost based optimizer of a relational database uses the approximate histogram information to improve query plans. We expose available histograms of a relational database in a homogeneous way. The monitoring service builds on our earlier work done for the D^{3}G framework [13] to provide exact continuous statistics (minimum and maximum values of a column, missing values, etc.) for certain parts of a database (likely to be the ones used frequently). In distributed and/or parallel environments, data transportation costs are of big importance. These costs are dependent on the number of rows and their individual sizes. To allow improved estimates for data access and integration, we provide the average row length (in bytes) for a table and the average column length (in bytes) for each column.

V. Implementation of DSMON

Our research prototype is built on top of the open-source Web Service Resource Framework (WSRF) [23] implementation of the Globus Toolkit 4 (GT 4) [24], which provides a common base for the development of Web and Grid services. Figure 5 depicts an internal view on our data source monitoring component implemented by two services fulfilling all of the requirements introduced in Section III, except of the sink monitoring information. In order to be generally applicable, the request to gather the needed metadata information by the monitoring component is done via a pull model over JDBC [1]. This means that it connects to the relational database and queries the system tables in a given frequency. The setup of our monitoring component for a database is done via two XML files. One holds the connect information to the database, the other allows to define what monitoring information should be provided and of which schemas and tables of a database to expose metadata. The set of databases to monitor can be changed at runtime via two methods of the service representing a database, namely *addDatabase* and *removeDatabase*. The *addDatabase* method needs the above mentioned XML configuration files as input. A flag indicates if the database should be permanently (making it available again when the service gets restarted) or temporarily monitored. For removal of a database (and its related tables), the database resource key is needed as input.

[1] a lot of the current available relational database management systems do not support external Web Service calls from within the database

```
<TableName>STATS_O</TableName>
...
<Indexes>
 <index distinct_keys="999687" name="SSI1"
        num_rows="1000000" unique="NO">
   <column descend="NO" name="A" position="1"/>
   <column descend="NO" name="C" position="2"/>
 </index>
</Indexes>
<Histogramms num_rows="1000000"
             avg_row_length_bytes="200">
 ...
 <colHistogramm name="H" num_distinct="3"
             num_nulls="0" sample_size="5909"
             avg_col_length_bytes="8">
  <ValueBHPart number="1998" value="0"/>
  <ValueBHPart number="1946" value="1"/>
  <ValueBHPart number="1965" value="2"/>
 </colHistogramm>
</Histogramms>
<ExactContinuousDS>
 ...
 <ecdsColumnType column_name="A" max_value="20.0"
        mean="15.06" min_value="10.0" missing="0"
        stddev="2.863" total_freq="1000000"/>
</ExactContinuousDS>
```

Figure 6. Resource properties of an Oracle table, including histograms and exact data statistics.

```
<TableName>STATS_M</TableName>
...
<Indexes>
 <index distinct_keys="6"
        name="PRIMARY" unique="YES">
  <column descend="NO" name="id" position="1"/>
 </index>
</Indexes>
<Histogramms num_rows="6"
             avg_row_length_bytes="150"/>
```

Figure 7. Resource properties of a MySQL table, including just a few data statistics.

As shown in Figure 5, the relationship between a database resource and table resources is a master-slave one. For each table to monitor, a database resource creates a table resource via its resource home. Afterwards, the resource properties of the table resource get updated by the worker thread of its master database resource. This worker thread has the only JDBC connection to the data source and shares it with the tables resources to update their resource properties. The resource properties of a table resource, an example can be found in Figure 6, and a database resource are all read only, except the update frequency property. Most interesting (and complex) is the histogram and exact continuous data statistics information. For column *H* detailed value-based histograms are provided by the monitored database. The column is expected to contain just three distinct values, nearly equally often, and no null values. Also exact continuous data statistics are available for column *A*, showing the actual minimum and maximum values for this column as well as standard deviation, total frequency and missing values.

A database resource provides the connection time and the tables property as notification topic [25], while the table resource provides means for a client to subscribe to changes in the logical schema, indexes, histograms and exact continuous data statistics. Our implementation supports a homogeneous representation of all three commonly used histogram types (value based, height based, compressed) for numerical columns. The resource properties of an example MySQL table are shown in Figure 7. No histogram information is available for table

```
<TableName>STATS_P</TableName>
...
<Indexes>
 <index name="pni" num_rows="1025" unique="NO">
  <column name="A" position="1"/>
 </index>
</Indexes>
<Histogramms num_rows="1025"
             avg_row_length_bytes="25">
  <colHistogramm name="X" num_distinct="1000"
        num_nulls="0" avg_col_length_bytes="8">
     <ValueBHPart number="5" value="0"/>
     ...
     <HeightBHPart number="90" value="999"/>
  </colHistogramm>
</Histogramms>
```

Figure 8. Resource properties of a PostGre table, including histograms.

columns in MySQL 5. An example of resource properties for a PostGre table is given above. PostGre 8 provides rich information about table columns, including compressed histograms (value and height based for one column possible) and column sizes.

VI. Performance

Our DSMON component uses one JDBC connection per database (and its related tables), which gets closed again after its usage to collect the monitoring information. This is necessary to get a real connection time value as workload/performance indicator; although we know that opening a connection is a quite resource-expensive step in database transactions requiring multiple separate network round-trips.

A. Database workload reduction

The diagram in Figure 9 shows the number of database requests needed to gather the required metadata sets in correlation to the number of applications and metadata sets. If each application has to examine the data sources themselves, A is the number of applications and M the number of metadata sets (we assume one request to gather one metadata set), the number of requests is defined by $nr_{requests} = A * M$ to get a snapshot of the current data source situation. For continuous metadata needs, e.g. during long running queries or in registry services as introduced in Section II, this formula has to be extended by refresh time T, resulting in $nr_{requests} = A * M * T$.

In contrast, with a monitoring service in place the number of requests to collect the data source metadata is defined by $M * T$. To support continuous metadata needs we have to add the value of $A * M$ for topic subscription, resulting in $nr_{requests} = A * M + M * T$.

Lets assume the following scenario for illustrating the data request savings via our monitoring service DSMON. Let there be 5 applications, needing 10 different metadata sets with 9 further checks if something changed. Without a monitoring service this will lead us to $5 * 10 * 10 = 500$ requests against the target data source. Using our

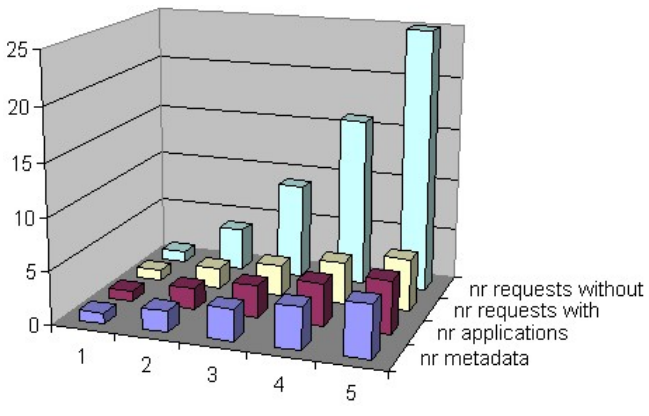

Figure 9. Number of database requests with and without monitoring service – dependent on required metadata sets and number of applications.

	MySQL	PostGre	Oracle
logical schema			
first time	40	30	200
avg time	20	20	30
indexes			
number of SQL queries	2	2	2
number of joins	0	3	0
first time	60	30	200
avg time	35	20	30
histograms			
number of SQL queries	1	2	3
number of joins	0	0	0
first time	90	180	200
avg time	15	25	35
exact statistics			
number of SQL queries			1
number of joins	N/A	N/A	0
first time			100
avg time			20

TABLE I.
MONITORING OVERHEAD (IN MS) TO GATHER TABLE INFORMATION INCLUDING AVERAGE DATA SIZES OF ROWS/COLUMNS

monitoring service will reduce the number of requests against the data source to $10 * 10 = 100$, adding $5 * 10 = 50$ subscription costs results in $5 * 10 + 10 * 10 = 150$ requests.

B. Monitoring component performance

Table I shows the time needed[2] to gather the monitoring information of one database table in one of the relational target databases (MySQL 5, PostGRE 8, Oracle 10g). For each part of the monitoring information the required individual structured query language (SQL) queries (select statements), the number of joins needed in all of them combined and the time for the overall process is given. All databases were installed with default setup on dedicated machines and were accessible via LAN to the monitoring components. The monitored table in each database had the same structure (primary keys, indexes, columns/types) and statistics were gathered for Oracle/PostGre via DBMS default options. The times were measured by the monitoring components (service side) via the standard Java function *System.currentTimeMillis()*. As the extraction of the logical schema was done via standard JDBC methods, these values are dependent on the different ways they have been implemented in the used JDBC drivers. Measurements for exact continuous data statistics via D^3G functionality are just done for Oracle, as it is currently our only supported relational database management system. Measurements to gather database related information (like product name and version) are not necessary, as they occur just once per database and are extracted via standard JDBC methods.

VII. TOWARDS A DATA SOURCE METADATA INFORMATION SERVICE

Our future research agenda for the further development of the basic DSMON component includes the following topics: investigate which security concept is needed for data related metadata information; provide information relevant for sink databases (like available tablespace of a table); further performance testing, especially the number of databases and tables one service is able to monitor, and other scalability issues.

The future data source metadata information service (DSMIS) will use several of the current data source monitoring services (DSMON) as lower level data collection system close to the target data source. It will act as client interface and support advanced functionalities by persisting retrieved monitoring information and allowing foreign metadata providers to sink their findings for further usage.

A. Motivation and Objectives

One of the key issues in data management and integration over multiple administrative domains is to handle the autonomy of data sources [26]. DSMON collects and exposes several available in-system statistics of relational databases in a uniform way. This allows it to support the usage scenarios described in Section II. With DSMIS, we are aiming at the following objectives:

Overcome metadata shortcomings of a target data source. This can be necessary when there are no/few statistics available inside certain data sources, e.g. MySQL.

Reuse workload on it by other data accessors. Each data access middleware request puts workload on the data source. Why not examine results of it to learn more about it? A good example for a possible application using a sink interface is OGSA-DQP. When its table scan operator can not forward any conditions to the target relational database and therefore gets the whole data of the table via a 'SELECT * FROM table' query and performs other operations itself (projection, joins, etc.).

Support non-relational data sources. Structured files and simpler XML data sources are often mapped towards an intermediate relational schema [27] (this is also done for Web databases [28]) within a mediator-wrapper approach [29]. Complex XML datasources are becoming ubiquitous with XML based data integration systems, e.g. TUKWILA [30], consequentially arising. Advanced metadata is needed for their efficient data managment as well.

Allow predictions about the future behaviour of a data source. Such predictions might be used within query optimization frameworks as proposed in [31], where wrappers (in our case the metadata information service as external and non-intrusive observer) supply information on the costs and cardinalities of their portions of a query plan. Improved metadata and prediction capabilities are also needed for advanced Quality of Service [32] support when accessing distributed data sources.

B. Methodology

We elaborate on the following issues to complete our overall architecture and provide the advanced features of

[2]first time value given was minimum (plus 50 per cent possible), avg time given was maximum (minus 75 per cent possible)

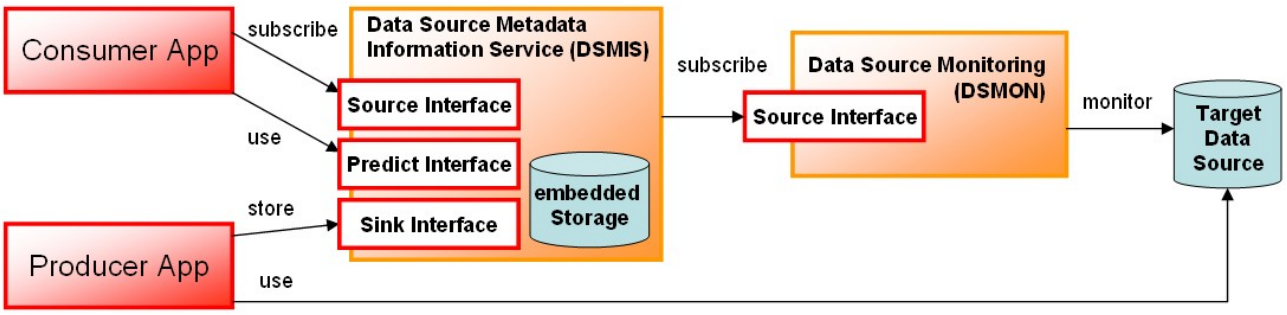

Figure 10. General architecture to support predictions, persisting and external updates of monitoring information. The current data source monitoring service (DSMON) acts as lower level data collection system close to the target data source, while the future advanced metadata information service (DSMIS) acts as client interface with additional functionality.

the DSMIS service:

Sink monitoring information. For *'second-hand'* monitoring information an appropriate inclusion mechanism (e.g. just fall-back option if no other information is available) and scope (e.g. just considered for notifying application) rules have to be defined in a flexible way, e.g. via policy based approaches [33]. Using the technique described in [20] to pass on monitoring information throughout a operator tree of query processing, on-the-fly calculated histograms can be propagated to our metadata information service DSMIS. For maintaining histograms more generally from query (including conditions) feedback the techniques of self-tuning histograms [34] can be applied.

Non-relational data sources. Efficient query processing over XML datasources requires accurate estimation of the selectivities of the path expressions contained in a query. Structural relationship information [16], the XML equivialent to traditional histograms, is needed. Workload driven query feedback based histogram learner, e.g. XPathLearner [35], have already been proposed and can be used to increase the number of available metadata sets for XML sources.

Predictions. The accuracy and application potential of predictions [36] of the future behavior of resources is heavily dependent on the quality of the historical observations about them. A first starting point for a data source is to store and examine the connection times as workload indicator. Patterns of typical working (higher average connection times) and maintenance hours (no connection possible for similar time period at certain days) can so be revealed. A thereon constitutive and broad applicable research direction is finding the correlation between the connection time and its influence on a conrete query response time. The following data table might be used initially, gathered on-the-fly during data integration as described above, for that purpose: query time, hash code of query statement, response time in ms, response size in rows, response size in kilobytes, average connection time during query processing. One idea is to infer from historical data with certain workload on current workload and response time. Another possibility to see trends in the development of a data source or table, e.g. growing 10% a week or getting smaller but also slower.

C. Architecture and Technical Realization

Figure 10 pictures the general architecture of a metadata infrastructure for data sources. DSMON acts as lower level data collection system close to the target data source. It exploits the metadata capabilities of the data source and provides therefore a read-only source interface for monitoring information. The data source metadata information service (DSMIS) provides more advanced funtionalities, based on the collected data from DSMON and metadata updates by producer applications (e.g. query processing engine in Figure 2) retrieved through the sink interface. To persist the data from DSMON and other metadata producers, a storage facility is required. A tailored HSQLDB [37] installation can be used for various application and deployment scenarios. It has a JDBC driver, offers a database engine with small memory footprint and supports various tables (in-memory and disk-based) and modes (embedded and server).

D. Execution Scenario

The following use case explains in more detail how the components in Figure 10 can play together. Imagine a MySQL database (not providing any histograms) is monitored by our DSMON component. A DSMIS component subscribes to several DSMON components, gathers the exposed low level metadata information and stores it locally. An application uses an OGSA-DQP service, needing metadata for its query optimization (consumer role), for a data integration task involving images from this MySQL database. During the query execution the sink interface of DSMIS is used by OGSA-DQP (producer role) to store on-the-fly calculated histograms for this table as well as the deails about the query response described above to be used for future predictions by DSMIS.

VIII. Related Work

Basically, the work presented here addresses monitoring and virtualization of metadata about various heterogeneous data sources over the Grid. Moore and Baru [38]

present an overview of ongoing research in the area of virtualization services for data intensive Grid systems. The Storage Resource Broker [17] is a middleware that provides metadata based file access via a logical name space for identifiying distributed data. It supports location transparency by accessing data sets and resources based on their attributes rather than their names or physical locations. OGSA-DAI [39] is the de facto standard for accessing databases on the Grid, providing logical schema information. OGSA-DQP [14] uses some basic physical information, e.g. average row size, and few data related information, e.g. number of rows in a table, for query optimization by using physical metadata extractors installed on all OGSA-DAI data resources which have to be accessed.

Monitoring of distributed systems [5] for particular purposes such as accounting and fault-tolerance is vital for volatile environments like the Grid. In [6], a comprehensive overview of monitoring components for Grid computing and network resources is provided, but to the best of our knowledge no service oriented monitoring tool for Grid-enabled relational databases has been reported yet. Available database monitoring tools are mostly focusing on the health status of a database and often provide some kind of internal performance tuning and profiling functionality for non-autonomous databases.

Performance predictions already play an important role to manage workload on parallel and distributed computing systems [36], several tools for computational Grid resources exist [40].

Our work focuses on the exposure of a homogeneous view of data related and data access related metadata about heterogeneous data sources. This is done via a uniform service oriented interface, currently providing metadata via DSMON for relational databases. The extended DSMIS functionality will allow to sink monitoring information and obtain predictions about the future behaviour of various kinds of data sources. This is crucial for various areas related to advanced data access, integration and management on the Grid.

IX. Conclusions

There is a widespread need for the interconnection of pre-existing and independently operated databases, as second-hand and publicly available data sources in e-Science. This paradigm shift in Grid computing towards more data-intensive applications goes hand in hand with the urgent need for a metadata information service for various kinds of data sources to support advanced distributed data managment. The process of obtaining, collecting, and presenting information required by an application about a target resource, called monitoring, for various particular purposes is viable for a distributed system like the Grid. Our current research prototype of DSMON, based on WSRF, comprises two services: one representing the interface to database resources and the other one to its related table resources. DSMON currently supports MySQL, PostGRE and Oracle databases. The exposed monitoring information for relational databases include detailed access related information (like connection time, logical schema and available indexes) and data related information (like available internal histograms, data sizes and exact data statistics). Our database monitoring service provides a homogeneous view on this metadata for heterogeneous relational databases via a uniform interface. The introduced usage scenarios indicate the versatile application areas for this component and point out the great need for advanced metadata about all kinds of data sources. Our monitoring component is not bound towards some specific data access and integration middleware, e.g. OGSA-DAI, which allows a broad user group.

The research effort described in this paper is an important step towards advanced distributed data management on the Grid. We allow to include rich metadata about relational databases into the decision processes of distributed data management tasks. Our next step is to ellaborate in more detail the architecture for our proposed metadata information service DSMIS, building upon DSMON but providing advanced functionalities such as prediction of data source behavior and external metadata updates. Some of the proposed methods, as well as the overall architecture, can also be applied towards non-relational data sources, which are also gaining on importance in data integration and management on the Grid.

In the end, it is all about making as much metadata and behavioural knowledge about a data source as easy accessible as possible to higher level services for their particular purposes at low cost. This has already proven useful for other Grid resources - because a little more metadata can go a long way for better efficiency.

Acknowledgments

We want to thank the attendees of CISIS'07 in Vienna (Austria) for valuable comments and discussions.

References

[1] A. T. Tony Hey, "The data deluge: An e-science perspective," in *Grid Computing - making the global infrastructure a reality*. John Wiley & Sons, 2003, pp. 809–824.

[2] D. Kossmann, "The state of the art in distributed query processing," *ACM Computing Surveys (CSUR)*, vol. 32, no. 4, 2000.

[3] J. Gray, "Distributed computing economics," MSR-TR-2003-24, 2003. [Online]. Available: http://arxiv.org/pdf/cs/0403019

[4] A. Gounaris, N. W. Paton, A. Fernandes, and R. Sakellariou, "Adaptive query processing: A survey," 2002. [Online]. Available: http://citeseer.nj.nec.com/gounaris02adaptive.html

[5] M. Mansouri-Samani and M. Sloman, "Monitoring Distributed Systems (A Survey)," Imperial College London, Tech. Rep. DOC92/23, 1992. [Online]. Available: http://citeseer.ist.psu.edu/25451.html

[6] S. Zanikolas and R. Sakellariou, "A taxonomy of grid monitoring systems," *Future Generation Computer Systems*, vol. 21, pp. 163–188, 2005.

[7] M. Sicilia, "Metadata, semantics, and ontology: providing meaning to information resources," *Int. J. Metadata, Semantics and Ontologies*, vol. 1, no. 1, pp. 83–86, 2006.

[8] K. Kumpf, A. Woehrer, S. Benkner, G. Engelbrecht, and J. Fingberg, "A semantic mediation architecture for a clinical data grid," in *Grids Computing for Bioinformatics and Computational Biology*. Wiley & Sons, 2007.
[9] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.
[10] A. Woehrer and P. Brezany, "A monitoring service for relational databases to support advanced data integration on the grid," in *Proceedings of the First International Conference on Complex, Intelligent and Software Intesive Systems*, 2007.
[11] Y. Ioannidis and S. Christodoulakis, "On the propagation of errors in the size of join results," in *Proceedings of the ACM SIGMOD*, 1991.
[12] Y. E. Ioannidis, "The history of histograms (abridged)." in *VLDB*, 2003, pp. 19–30. [Online]. Available: http://www.vldb.org/conf/2003/papers/S02P01.pdf
[13] A. Woehrer, L. Novakova, P. Brezany, and A. M. Tjoa, "D^{3}G: Novel Approaches to Data Statistics, Understanding and Preprocessing on the Grid," in *Proceedings of the IEEE 20th Intl. Conf. on Advanced Information Networking and Applications*, 2006.
[14] 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 *Proceedings of UK e-Science All Hands Meeting Nottingham*, 2003.
[15] Y. E. Ioannidis, "Query optimization," *ACM Computing Surveys*, vol. 28, no. 1, 1996.
[16] Y. Wu, J. Patel, and H. Jagadish, "Estimating Answer Sizes for XML Queries," in *Proceedings of EDBT*, 2002. [Online]. Available: http://citeseer.ist.psu.edu/754158.html
[17] A. Rajasekar, M. Wan, and R. Moore, "MySRB and SRB - components of a Data Grid," in *Proceedings of the 11th International Conference on High Performance Distributed Computing*, 2002.
[18] A. Gounaris, N. W. Paton, A. Fernandes, and R. Sakellariou, "Adaptive query processing and the grid: Opportunities and challenges," 2004. [Online]. Available: http://citeseer.ist.psu.edu/743963.html
[19] S. Pallickara and G. Fox, "An analysis of notification related specifications for web/grid applications," in *Proceedings of the IEEE ITCC Conference on Information Technology*, 2005.
[20] A. Gounaris, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou, "Self-monitoring query execution for adaptive query processing," *Data Knowl. Eng.*, vol. 51, no. 3, pp. 325–348, 2004.
[21] S. Bowers, K. Lin, and B. Luedascher, "On integrating scientific resources through semantic registration," in *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, 2004.
[22] S. Staab, A. Maedche, and S. Handschuh, "An annotation framework for the semantic web," 2001. [Online]. Available: http://citeseer.ist.psu.edu/385363.html
[23] The Web Service Resource Framework, http://www.globus.org/wsrf/.
[24] The Globus toolkit, http://www.globus.org/toolkit/.
[25] Web Service Topics, http://docs.oasis-open.org/wsn/.
[26] S. Busse, R. Kutsche, U. Leser, and H. Weber, "Federated information systems: concepts, terminology and architectures," http://cis.cs.tu-berlin.de/Dokumente/Papers/1999/BKLW99.ps.gz, 1999.
[27] A. Woehrer, P. Brezany, and A. M. Tjoa, "Novel mediator architectures for grid information systems," *FGCS*, vol. 21, no. 1, pp. 107–114, January 2005.
[28] I. Kojima and S. M. Pahlevi, "Design and implementation of OGSA-WebDB - a service based system for making existing web databases grid-ready," 2004.
[29] G. Wiederhold, "Mediators in the architecture of future information systems," 1992. [Online]. Available: http://www-db.stanford.edu/pub/gio/1991/afis.ps
[30] Z. G. Ives, A. Y. Halevy, and D. S. Weld, "Adapting to source properties in processing data integration queries," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press, 2004, pp. 395–406. [Online]. Available: http://www.cis.upenn.edu/ zives/research/adp.pdf
[31] M. T. Roth, F. Oezcan, and L. M. Haas, "Cost Models DO Matter: Providing Cost Information for Diverse Data Sources in a Federated System," in *Proceedings of VLDB*, 1999. [Online]. Available: http://www.vldb.org/conf/1999/P56.pdf
[32] R. Braumandl, A. Kemper, and D. Kossmann, "Quality of service in an information economy," *ACM Trans. Inter. Tech.*, vol. 3, no. 4, pp. 291–333, 2003.
[33] N. Damianou, A. Bandara, M. Sloman, and E. Lupu, "A survey of policy specification approaches," 2002. [Online]. Available: http://www.doc.ic.ac.uk/ mss/Papers/PolicySurvey.pdf
[34] A. Aboulnaga and S. Chaudhuri, "Self-tuning Histograms: Building Histograms Without Looking at Data," in *Proceedings of SIGMOD*, 1999. [Online]. Available: http://citeseer.ist.psu.edu/aboulnaga99selftuning.html
[35] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Parr, "XPathLearner: An On-line Self-Tuning Markov Histogram for XML Path Selectivity Estimation," in *Proceedings of VLDB*, 2002. [Online]. Available: http://citeseer.ist.psu.edu/525032.html
[36] S. A. Jarvis, D. P. Spooner, H. N. L. C. Keung, J. Cao, S. Saini, and G. R. Nudd, "Performance prediction and its use in parallel and distributed computing systems," *Future Gener. Comput. Syst.*, vol. 22, no. 7, pp. 745–754, 2006.
[37] HSQLDB, http://hsqldb.org/.
[38] R. W. Moore and C. Baru, "Virtualization services for data grids," in *Grid Computing - making the global infrastructure a reality*, 2003.
[39] OGSA-DAI, http://www.ogsadai.org.
[40] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 4, pp. 41–49, March 2003.

Alexander Woehrer is a PhD student and RA at the Institute of Scientific Computing, University of Vienna, Austria. He received his degree in Business Informatics in 2004 from the University of Vienna. He participated in research visitor programs at CERN, Switzerland and the National e-Science Center in Edinburgh, Scotland. His research interests include distributed data access, integration and management on the Grid, including its semantic and adaptive aspects.

Mag. Woehrer is a member of the ACM, the Open Grid Forum and the Austrian Computer Society.

Peter Brezany is a professor at the Institute of Scientific Computing, University of Vienna, Austria. He received his PhD in Computer Science in 1980 from the Slovak Technical University Bratislava, Slovakia. He began research in 1976 on the design of parallel programming languages and their compilers. Since 1990 he has worked on automatic parallelization of scientific applications for distributed-memory systems, parallel I/O support for HPC, and large-scale parallel/distributed data mining. His current research focus is knowledge discovery and data management on Grids.

Prof. Brezany is a member of the IEEE Computer Society and the Austrian Computer Society.