(Presented at *Proc. 1st European Across Grids Conference*,
Universidad de Santiago de Compostela, Spain, Feb. 13-14, 2003.)

High-performance GRID Stream Database Manager for Scientific Data

Milena Gateva Koparanova and Tore Risch

Department of Information Technology, Uppsala University,
SE-75105 Uppsala, Sweden,
Milena.Koparanova@it.uu.se, Tore.Risch@it.uu.se

Abstract. In this work we describe a high-performance stream-oriented distributed database manager and query processor under development that allows efficient execution of database queries to streamed data involving numerical and other data. Very high performance is attained by utilizing many object-relational main-memory database engines running on PCs and connected through the GRID.

1 Introduction

We are developing a new kind of database manager utilizing the evolving GRID infrastructure[5] for distributed computations on large data streams. The *GRID Stream Data Manager (GSDM)* will have high performance and support for customizable representation of streamed data in distributed data and computational servers. The target application area is space physics, in particular the LOFAR/LOIS project [9–11], whose purpose is to develop a distributed software space telescope and radar utilizing the GRID. LOFAR/LOIS will produce extremely large raw data streams by sensor networks receiving signals from space. Various numerical selection and transformation algorithms are applied on these streams before the data to be delivered to the client workstations for visualization and other processing in form of streams called *beams*, with rates of several gigabits per second [20].

Our approach to meet the demands of the LOFAR/LOIS (online) applications for very high performance and extensibility is to develop and utilize a distributed, main-memory, object-relational, and stream-oriented DBMS running

on clusters of computers. We are extending an existing main-memory object-relational DBMS engine[16] with capabilities for processing distributed streams.

The remainder of the paper is organized as follows. In the next section we consider the need for and consequences from the stream orientation for the development of the system. Section 3 discusses the GSDM as a new type of application for computational GRIDs. An overview of the GSDM system architecture is presented in section 4, and we summarize in section 5.

2 Stream Database Manager

Regular database management systems(DBMSs) store tables of limited sizes while stream database systems (SDBSs) also deal with on-line streams of unlimited size. The raw data and beams generated by LOFAR/LOIS sensor networks are typical example of streams. Data streams have characteristics that require different handling from the DBMSs. A lot of research in stream data management has been done recently [1–4, 8, 12, 13, 18] and the area offers a number of open research questions. Several important characteristics of data streams make them different than other data: they are infinite, once a data element has arrived, it is processed and either archived or deleted, i.e. only a short history can be stored in the database. It is also preferable to process data elements in the order they arrive, since sorting, even of substreams of a limited size, is a blocking operation.

The specifics of the data streams require to consider a data model that allows streamed data representations and operations over streams to be easily specified. An example of stream-oriented data model is SEQ [17] that is used as a base for modeling of streams due to their sequence nature.

The infinite size imposes a limitation in the stream representation in form of substreams of a limited size, called *windows*. Several operations that are specific for streams must be introduced (e.g. resample and drop in [3]), while some traditional ones need their semantics to be redefined in the context of stream windows (e.g. join of windows and moving window aggregates [18]).

Since stream data elements are arriving over time, a natural technique for querying such data are so called *continuous queries(CQs)* [19]. CQs are installed once and run continuously over incoming data elements until they are stopped explicitly. The presence of long-running CQs increases the importance of DBMS adaptivity which motivates the work presented in [13]. Techniques as approximate query answering, data reduction and multi-query optimization also gain greater importance and applicability in the context of stream data query processing.

SDBS architectures are proposed in [2, 3]. A related area is distributed networks of sensors considered in [1, 12], where a large numbers of small sensors, most likely power limited, generate relatively simple data at rates much slower than in LOFAR/LOIS application. In our application area we have a limited number of scientific data streams with very high generation rate. The expected

CQs will contain User-Defined Functions (UDF) over non-relational data representations of stream windows.

Most of the SDBSs described in the literature have centralized query processing and scheduling where a central node has more or less global information about the system and makes optimization and scheduling decisions accordingly. The need for efficient execution of a number of CQs on clusters of GSDMs puts additional difficulties because of the distribution and parallelism. For example, the system design must answer to questions as how to distribute the work among the database nodes, what type of parallelism would give greater advantage for scientific data streams with applied UDFs, how to coordinate the operations from a single pipeline running on different nodes or even clusters. Therefore algorithms for query optimization and scheduling have to be developed that take into account the specifics of the distributed GRID environment.

3 A Computational GRIDs Application

We can consider the GSDM project as a new kind of application for computational GRIDs. Utilization of parallel and distributed GRID environment is motivated by the following reasons: i)The volume of data produced is too large to fit in a single main-memory, and therefore suggests data to be distributed among clusters of main-memories. ii) The very high data flow rate requires very high performance of insert, delete and data processing operations.

Different kinds of parallelism can be utilized to achieve high performance. For very heavy computational operations data partitioning parallelism must be considered, while for cheaper operations a pipelined form of parallelism might be useful. The fragmentation strategies for data parallelism are well investigated for relational databases. Research has been done on data fragmentation problems in Object-Relational DBMSs that discusses how to achieve efficient parallel execution of UDFs while preserving their semantics [7, 15].

Several projects as Globus [6] and Nordugrid [14] provide tools for building computational GRID infrastructure. We consider computational GRIDs as more appropriate for our GSDM than a regular parallel computer because of dynamics and scalability. Computational GRIDs are a natural extension of parallel computers allowing not only greater processing power, but also ability for dynamic resource allocation and incorporation of new nodes when necessary. For instance, if a new CQ with high cost UDFs is installed on the GSDM, this may require staging and starting the database manager on new nodes and scheduling of some query operations on them. The dynamics can also be important characteristic in an environment where different number of data streams are used over time or the source streams have varying incoming rates. The efficient utilization of computing resources requires ability to dynamically incorporate new nodes to the system, and to free them for other users when not needed any more.

Several problems and research questions arise from utilization of the computational GRIDs for an application such a database management system. Resource management of the multiprocessors and clusters of workstations is per-

formed typically by a batch system, that takes care of the job scheduling, queue placement and prioritizing. Most of the applications requiring computational power run as batch jobs. The GSDM can be considered as a persistent job with dynamic resource requirements (increasing and decreasing over time) and interactive user interface. The DB managers and users must be provided with ability to install and stop different CQs during a user session without need to restart the system when a parameter has changed. Therefore, extensions of the job management systems of computers connected through the GRID are needed, such that interactive database query jobs are to be permitted. Working in a GRID environment requires also to consider issues as security and accessibility. For example, if a central scheduler is used for scheduling of the CQs, it must be accessible from all participating nodes, which limits its possible locations and creates a potential bottleneck. The GRID infrastructure should provide a new service by which database servers running on different clusters can communicate with very high performance.

4 GSDM System Overview

Figure 1 illustrates a scenario of applications interacting with GSDM through the *GSDM service manager*. It is a relatively light-weight server that receives GSDM commands and, depending on their kinds, starts or calls other GSDM servers. The GSDM commands can, e.g., be registration of CQs or information about the active CQs running currently in the system.

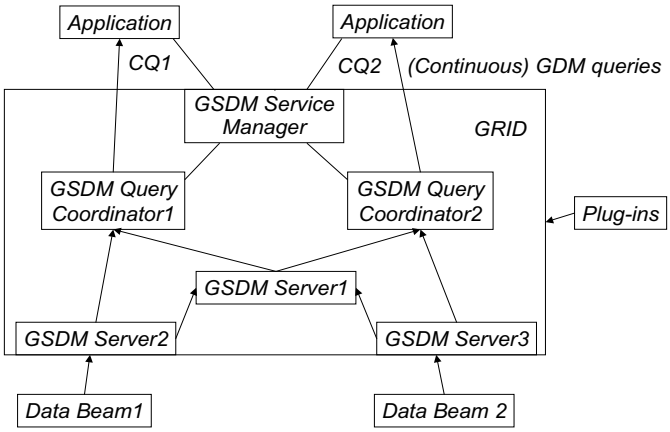

Fig. 1. GRID Stream Data Manager Scenario

In the figure there are two applications that have registered to the GSDM two different continuous queries, CQ1 and CQ2. The queries specify joining and filtering of data from two different data streams, in the example called Data Beam 1 and 2. The GSDM service manager starts a *GSDM query coordinator* node and delegates the service of the received continuous query to this node. The query coordinator delivers the results of the CQ as a stream to the application by receiving data streams from other GSDM nodes or source streams. For a given CQ the query compiler of the query coordinator will construct a distributed execution plan accessing other GSDM servers or source streams. In the example, the GSDM query coordinator 1 has created an execution plan for CQ1 where the coordinator joins data streams from Data Beam 1 and 2 through the intermediate GSDM servers 1 and 2. Analogously the query coordinator 2 joins streams from the sources through GSDM servers 1 and 3. GSDM server 1 produces substreams used in both queries by joining data streams from server 2 and 3. A CQ can include regular set operators extended with user-defined application-dependent algorithms for stream filtering and fusion that are implemented as plug-ins and installed on GSDM nodes.

In the scenario it is assumed that the source streams can deliver data directly to the cluster nodes running GSDM servers 2 and 3. This requires, e.g., the GRID infrastructure to provide a method to access the IP address of individual nodes inside a cluster, which is not always possible presently but is a desired service. The ability to deliver stream data directly to nodes inside a cluster is necessary in our case since the target LOIS/LOFAR applications produce very large streams that cannot be managed through a single node.

Figure 2 shows the architecture of a single GSDM node. An application communicates with a GSDM node acting as query coordinator through the *application API*. It provides primitives to register CQs and to deliver CQ results to the application. The *continuous query manager* registers and processes continuous queries. It produces optimized distributed CQ execution plans. The CQs specification is based on regular database queries and the system internally uses a regular database *query processor* for their optimization.

When a distributed CQ execution plan is produced the *resource manager* of the GSDM node is called. It is a software module that interfaces the GSDM service manager and other GSDM servers in order to decide where the individual operations in the query plan are to be executed. The GSDM service manager needs to be contacted, e.g., in order to find out whether there are some other similar CQs whose execution plans can be utilized by the new plan. The resource manager can start new GSDM servers when necessary. In order to allocate new computer resources the resource manager module of a GSDM node needs to cooperate with the resource manager of the cluster.

Once the execution plan is created the GSDM node needs to communicate with other GSDM servers and data sources through the *wrapper* interfaces. In the scenario the system has to interface mainly stream data sources, but also regular relational databases can be accessed through a *relational data source wrapper*. In the figure the *Global Sky Model* is a relational database storing the most

visible sky objects. Each kind of stream data source needs a source wrapper that accesses the source, builds windows of streamed data, controls the data flow, etc. The wrapper is instantiated for each accessed data stream of its kind. A particular wrapper interfaces inter-GSDM streams. The *plug-ins* contain user defined filtering and fusion algorithms implemented as conventional programs and dynamically uploaded into the GSDM servers needing them.

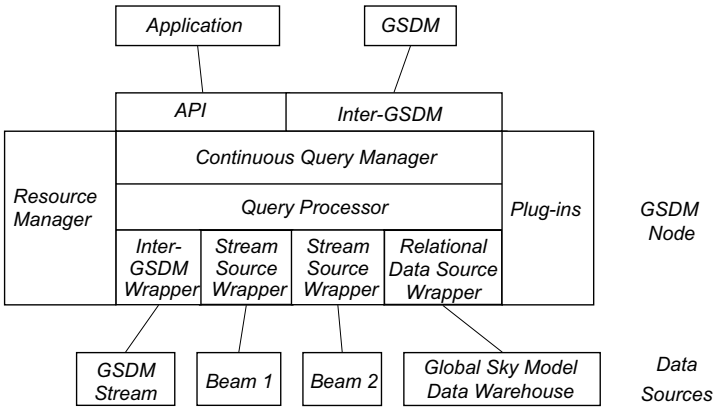

Fig. 2. GSDM System Architecture

5 Summary

We described the GSDM project - a GRID-enabled, main-memory, object-relational and stream DBMS. Main-memory, distribution, and query processing provide high performance, while object-relational functionality provides extensibility capabilities. We utilize extensibility of an existing main-memory object-relational DBMS engine to add stream orientation and customized data representations of scientific data to achieve flexible and high-performance processing of scientific streams. A prototype of the GSDM system is under development at Uppsala Database Lab and we outlined its distributed architecture.

References

1. P. Bonnet, J. Gehrke, P. Seshadri: Towards Sensor Database Systems. In *Proc. of the 2nd Intl. Conf. on Mobile Data Management*, Hong Kong (2001)
2. B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom: Models and Issues in Data Stream Systems. *ACM PODS* (2002) 1-16
3. D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul and S. Zdonik: Monitoring streams - a New Class of Data Management Applications. In *Proc. of the 28th VLDB Conference* (2002) 469-477
4. J. Chen, D.J. DeWitt, F. Tian, Y. Wang: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. *SIGMOD Conference 2000* (2000) 379-390
5. I. Foster, C. Kesselman (eds.): *The Grid: Blueprint for a new Computing Infrastructure*. Morgan-Kaufmann (1999)
6. The Globus Project. `http://www.globus.org`
7. M. Jaedicke, B. Mitschang: On Parallel Processing of Aggregate and Scalar Functions in Object-Relational DBMS. *ACM SIGMOD Conference*, Seattle, USA, (1998)
8. L. Liu, C. Pu, and W. Tang: Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Trans. on Knowledge and Data Engineering*, 11(14) (1999) 610-628
9. LOFAR: Low Frequency Array. `http://www.lofar.org`
10. LOIS - A LOFAR Outrigger in Scandinavia. `http://www.physics.irfu.se/LOIS`
11. Bo Thide (ed.): First LOFAR/LOIS Workshop, Sweden, June 17-19 (2001) `http://www.physics.irfu.se/LOIS/Workshops/Vaxjo010617-19/index.shtml`
12. S. Madden, M.J. Franklin: Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data. *ICDE 2002* (2002) 555-566
13. S. Madden, M.A. Shah, J.M. Hellerstein, V. Raman: Continuously Adaptive Continuous Queries over Streams. *SIGMOD Conference 2002* (2002) 49-60
14. NORDUGRID: Nordic Testbed for Wide Area Computing and Data Handling. `http://www.nordugrid.org/`
15. K.W. Ng and R. Muntz: Parallelizing User-Defined Functions in Distributed Object-Relational DBMS. *IDEAS 1999* (1999) 442-445
16. T. Risch, V. Josifovski: Distributed Data Integration by Object-Oriented Mediator Servers. *Concurrency and Computation: Practice and Experience J.*, 13(11), John Wiley & Sons (2001)
17. P. Seshadri, M. Livny and R. Ramakrishnan: SEQ: A Model for Sequence Databases. In *Proc. of the 11th ICDE Conference* (1995) 232-239
18. P. Seshadri, M. Livny and R. Ramakrishnan: The Design and Implementation of a Sequence Database System. In *Proc. of the 22nd VLDB Conference* (1996) 99-110
19. D. Terry, D. Goldberg, D. Nichols, and B. Oki: Continuous Queries over Append-Only Databases. In *Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data* (1992) 321-330
20. C.M. De Vos, K. van der Schaaf, and J. Bregman: Cluster Computers and Grid Processing in the First Radio-Telescope of a New Generation. In *Proc. of the First IEEE/ACM International Symposium on Cluster Computing and the Grid- CCGrid'2001* (2001)