

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,Tore.Risch}@it.uu.se

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

1 Introduction

We are developing a new kind of database manager utilizing the evolving GRID infrastructure [6] for distributed computations on large data streams. The *GRID Stream Database 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 [10,11,12], 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. The data is delivered to the client workstations for visualization and other processing in form of streams called *beams*, with rates of several gigabits per second [22].

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[17] 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 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.

* This project has been supported by VINNOVA under contract #2001-06074.

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 examples of streams. A lot of research in stream data management has been done recently [1,2,4,5,9,13,14,20] 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 is stored in the database. It is also preferable to process stream data with order-preserving non-blocking algorithms.

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. Examples of stream-oriented data models and query languages are SEQ [19] and Tribeca [18].

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 are introduced (e.g. resample and drop in [4]), while some traditional ones need new semantics in the context of stream windows (e.g. join of windows and moving window aggregates [20]).

Since stream data elements are arriving over time, a natural technique for querying such data are so called *continuous queries (CQs)* [21]. 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 adaptability which motivates the work presented in [14]. 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 [2,4]. A related area is distributed networks of sensors considered in [1,13], where a large numbers of small sensors, most likely power limited, generate relatively simple data at rates much slower than in our LOFAR/LOIS application. The scientific data streams in our application area have high generation rate and contain complex numerical data which suggest for non-relational stream representations by User-defined Data Types (UDTs). The continuous queries over LOFAR/LOIS streams contain application-dependant User-Defined Functions (UDFs) over UDTs.

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. The architectural challenges in the design of a distributed stream processing system are discussed in [3]. In the GSDM system design we have faced and must address similar problems, e.g., how to distribute the work among the database nodes, what type of parallelism would give greater advantage for scientific data streams

with UDFs applied, and 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: 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 [8,16].

Several projects as Globus [7] and Nordugrid [15] provide tools for building computational GRID infrastructures. 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 some query operations on these. The dynamics can also be important characteristics in an environment where different numbers 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.

Utilization of the computational GRIDs for such a database management system opens several problems and research questions. Resource management of the computing nodes is performed typically by a batch system that takes care of job scheduling, queue placement, and prioritizing. Most of the applications for computational GRIDs 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 some 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 also raises new security and accessibility issues.

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 the distributed GSDM system. There are three types of GSDM nodes. The *Metadata Manager* is a relatively light-weight server that stores the metadata of the system and interacts with users and applications. It receives GSDM commands and, depending on their kinds, might start or call other GSDM nodes. The GSDM commands can be, e.g., registration of a new CQ or a meta-query about the active CQs running currently in the system.

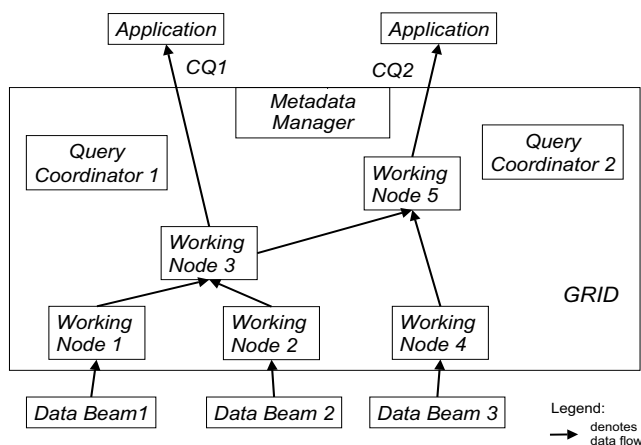


Fig. 1. GRID Stream Data Manager Scenario

In the figure there are two applications that have registered to the GSDM two different continuous queries, CQ1 and CQ2. A CQ can include regular set operators extended with user-defined application-dependent algorithms for stream filtering, transformation, and fusion. The queries in the example specify joining and filtering of data from three different data streams, called Data Beam 1, 2, and 3. When the metadata manager receives command to register new CQ it starts a *GSDM Query Coordinator* node and delegates the service of the received continuous query to this node. For better scalability of the system there can be many coordinators as indicated in the picture where there is one coordinator node per CQ. The execution of CQs is performed by *GSDM Working Nodes*

(WNs). For a given CQ the query compiler of the query coordinator will construct a distributed execution plan to be executed on the working nodes. In the example, the query coordinator 1 has created an execution plan for CQ1 where the WN3 joins data streams from Data Beam 1 and 2 through the intermediate WN1 and WN2.

The arrows on the fig. 1 denote the data flow between stream sources, GSDM nodes and applications. The GSDM nodes also exchange control messages which are not shown to keep the scenario picture simpler.

In the scenario it is assumed that the source streams can access the IP addresses and deliver data directly to the working nodes 1, 2, and 4. This assumption is important since the target LOIS/LOFAR project produces very large streams that cannot be managed through a single node. Many computer resources accessible through the GRID infrastructure do not currently satisfy this requirement. It is desirable to consider such a requirement when new GRID-enabled computer resources are designed.

In the following we will describe in more details the software architecture of different GSDM nodes. The metadata manager interacts with the applications through the *application API* module. It provides primitives to register, start and stop monitoring a CQ. In its local database the metadata manager stores information about the registered and running CQs, coordinators, and working nodes. This allows the applications and users to ask meta-queries about, e.g., the status of CQs running currently in the system.

The query coordinator produces optimized distributed CQ execution plans by the module *CQ Decomposer and Optimizer*. When deciding how to assign the individual operations in the CQ execution plan to working nodes, the coordinator contacts the metadata manager in order to find out whether there are some other running CQs whose execution can be utilized by the new plan. In the example on fig. 1 the Query Coordinator 2 constructs the execution plan for CQ2 that combines three data beams and finds out that the working node 3 already joins the data beam 1 and 2 in the way as required in CQ2. The *Resource Manager* module in the coordinator can start new GSDM working nodes when necessary and stop the running ones when the user stops monitoring the correspondent CQs. In order to allocate new computer resources the resource manager module needs to cooperate with the resource manager of the cluster.

Figure 2 shows the architecture of a single GSDM working node. The *Continuous Query Manager* registers continuous queries sent by the coordinator and processes them continuously over incoming stream data. The CQs specification is based on regular database queries and the system internally uses a regular database *Query Executor* to execute them on the current stream windows. The stream data sources are received by the *Stream Consumer* module. It is a type of *wrapper* interface that takes care of data encoding into internal stream representation, builds windows of streamed data, controls the data flow, etc. In a data driven stream processing paradigm, the stream consumer must also open listening sockets in order to receive streamed data and have a policy to deal with stream overflow/ underflow. The stream consumer is instantiated for each

accessed data stream of its kind. A particular stream consumer receives inter-GSDM streams. Stream consumer module is also needed by the application in order to receive streamed results. The *Continuous Query Producer* is a module that sends the resulting stream of a continuous (sub)query to the application or to the next working node in the CQ execution plan. The *Plug-ins* contain user-defined application-dependant algorithms for signal transformation, filtering, and combining, implemented as conventional programs and dynamically uploaded into the working nodes that need them.

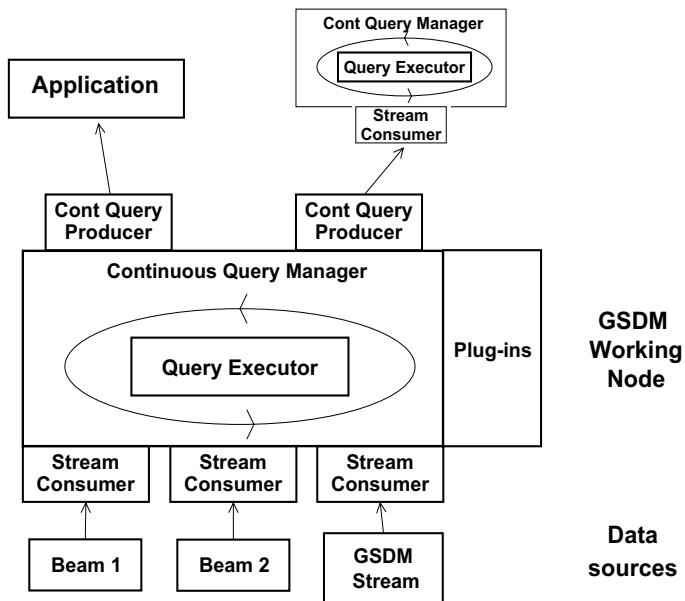


Fig. 2. GSDM Working Node Architecture

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. This achieves flexible and high-performance processing of scientific streams in form of CQs with UDFs. We outlined the distributed architecture of the GSDM prototype system under development at Uppsala Database Lab and discussed some of the problems and open research questions that arose in the development process.

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. M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik: Scalable Distributed Stream Processing. In *Proc. of the 2003 CIDR Conference* (2003)
4. 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
5. J. Chen, D.J. DeWitt, F. Tian, Y. Wang: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. *SIGMOD Conference 2000* (2000) 379-390
6. I. Foster, C. Kesselman (eds.): *The Grid: Blueprint for a new Computing Infrastructure*. Morgan-Kaufmann (1999)
7. The Globus Project. <http://www.globus.org>
8. M. Jaedicke, B. Mitschang: On Parallel Processing of Aggregate and Scalar Functions in Object-Relational DBMS. *ACM SIGMOD Conference*, Seattle, USA, (1998)
9. 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
10. LOFAR: Low Frequency Array. <http://www.lofar.org>
11. LOIS - A LOFAR Outrigger in Scandinavia. <http://www.physics.irfu.se/LOIS>
12. Bo Thide (ed.): First LOFAR/LOIS Workshop, Sweden, June 17-19 (2001) <http://www.physics.irfu.se/LOIS/Workshops/Vaxjo010617-19/index.shtml>
13. S. Madden, M.J. Franklin: Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data. *ICDE 2002* (2002) 555-566
14. S. Madden, M.A. Shah, J.M. Hellerstein, V. Raman: Continuously Adaptive Continuous Queries over Streams. *SIGMOD Conference 2002* (2002) 49-60
15. NORDUGRID: Nordic Testbed for Wide Area Computing and Data Handling. <http://www.nordugrid.org/>
16. K.W. Ng and R. Muntz: Parallelizing User-Defined Functions in Distributed Object-Relational DBMS. *IDEAS 1999* (1999) 442-445
17. 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)
18. M. Sullivan and A. Heybey. Tribeca: A system for managing large databases of network traffic. In *Proc. of the USENIX Annual Technical Conference*, New Orleans, LA, (1998)
19. P. Seshadri, M. Livny and R. Ramakrishnan: SEQ: A Model for Sequence Databases. In *Proc. of the 11th ICDE Conference* (1995) 232-239
20. 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
21. 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
22. 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)