Secondo-grid: an Infrastructure to Study Spatial Databases in Computational Grids

Wladimir S. Meyer[1], Jano Moreira de Souza[2], Milton Ramos Ramirez[3]

[1] Computer Science Department, Graduate School of Engineering
Federal University of Rio de Janeiro (UFRJ)
PO Box 68.511 - ZIP code: 21945-970 – Rio de Janeiro, RJ - Brazil

[2,3] Computer Science Department, Institute of Mathematics
Federal University of Rio de Janeiro - Brazil

`{wsmeyer,jano}@cos.ufrj.br`   `milton@labma.ufrj.br`

Abstract. *This article describes a framework designed to be used as a platform for experiments of distributed spatial databases in a computer grid. The environment consists of an open database modified to send and receive jobs to others databases using a grid middleware and its services. With this framework resources can be discovered and monitored, for example, to help in the process of constructing a query plan in a distributed environment. A case study based in a geographic database is being used to validate the framework.*

1. Introduction

New communication technologies are allowing the development of computer networks at a very high pace. The increase of speed has improved computing integration of networks all over the world.

Resources like the Internet are present in the routines of people and organizations, enabling new kinds of interactions.

Despite the great amount of computing resources, the level of integration is very weak in most of the cases. Hardware and software are spread around the organizations and their use is locally restricted. Large amounts of redundant data demand much effort from their owners to be produced.

The grid computing paradigm was created to reduce these kinds of problems and to offer support to high performance processing. The analogy with the power grid [Foster and Kesselman 1999], where there are power suppliers and consumers interconnected by an infrastructure, and anyone is capable to use the necessary amount of energy, the computer grid should be used to allow users or organizations to allocate computers' resources on demand to supply their needs. Today many organizations are using grid computing in complex situations, like drugs research, astronomy data processing, genoma research and others.

A particular and very important area within grid structures is Database Systems. Database servers in a grid can share not only data, but also specifics skills that can increase the possibilities of the environment.

This article aims to show some topics related with spatial databases, seen like grid resources, and to propose a framework to study and test new solutions involving distributed spatial databases management systems (DSDBMS) within grid environments.

An extensible [Dieker and Güting 2000] Data Base Management System (DBMS) was selected to increase the range of the possibilities in developing new studies. The Secondo DBMS is an open system capable of supporting many data models, and offering much more flexibility then conventional databases.

The proposed framework consists of a system that uses a modified Secondo and many grid services to allow native distributed query and high speed data transfers between servers. Another functionality is the possibility of registering database resources to be used by others servers in the grid. The status of hardware resources can be monitored to allow for building better query plans.

A case study, based in a geographic database, is being constructed to validate the framework.

2. Computational grids

One of the most recent fields related with high performance processing is grid computing. The grid structure is adequate to reach high performance since it can be formed by hundreds or thousands of computers working in a parallel form. An user does not need to know how or where a resource is: the structure is transparent to him.

Ian Foster wrote a preliminary description of a computational grid [Foster and Kesselman 1999]:

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities"

This definition was complemented in [Foster et al. 2000] with the following:

"Grid computing is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations."

This global definition emphasizes resource sharing and the idea that a grid can involve many organizations. To avoid doubts when classifying systems, Foster made a three-step checklist on a grid system [Foster 2002]:

- <u>It coordinates resources that are not subjected to centralized control</u> – in this environment there is coordination, despite of no central control. So the web can not be seen as a grid system;
- <u>It uses standard, open, general purpose protocols and interfaces</u> – Many systems that classify themselves like grids are not open or do not use a standard to permit a major degree of interoperability with others.
- <u>Its goal is to deliver non-trivial qualities of service</u> – the quality of services delivered by the system results from a synergy, so global quality is greater then the sum of its parts.

The sharing concept in a grid environment is not only related to file interchange, but introduces a new dimension to it as software and hardware can be accessed by all authorized users. Within this perspective, databases, files, memory, disk space and processors can be shared to support a specific demand.

In this context, two roles should be carefully defined: the resource producer and the resource consumer [Foster 2002]. The resource producer can define any kind of resources to share and should define the security policy to them. The resource consumer is an application that needs to access delivered resources.

To permit a high degree of integration in a heterogeneous environment between resource providers and consumers, interface standardization plays an important role. The Global Grid Forum (GGF) was created to standardize grid interfaces [Malaika et al. 2003].

The GGF consists of workgroups that are specialized in some areas of grid computing [GGF Areas/Groups 05]: infrastructure, data, architecture, compute, application, management and security.

The workgroups produce documents like technical specifications to be used by the grid community.

The Open Grid Service Architecture (OGSA) was proposed by GGF to be the backbone of computational grids. The OGSA identifies components needed to build a grid infrastructure and mechanisms that should be delivered as services in a web service manner. OGSA takes an important role in GGF environment since other groups work with its specifications.

In the database field, a specific workgroup should be highlighted:

DAIS (*Data Access and Integration Services*) – The activities of this work group are concerned with all issues necessary to ease access and interoperability among databases in the grid. It does not intend to implement a new DBMS, but to recommend interfaces and services to improve existing products. All interfaces and services are described in terms of XML documents (WSRF) and can support relational and XML databases. The idea is to allow access to data in a computer grid, on demand, that is without any previous knowledge about where they are and in which format they are stored.

The standards are converging to adopt a Web service approach to grid services. However, a typical Web service does not meet an important OGSA requirement: it must be stateful. This requirement means that, for example, properties of a resource must be available between two consecutive service calls. To reach this imposition, WSDL (Web Service Development Language) was extended to support resource properties and the new specification was named WSRF (Web Service Resource Framework) [Sotomayor 2005]. The WSRF was developed by the Oasis group (http://www.oasis-open.org) and is being used by many grid middleware products like Globus. This format was introduced in GGF in 2004 and probably will substitute the OGSI (Open Grid Service Infrastructure), a standard that was not well accepted in the Web community.

Another project involving OGSA and databases is the OGSA-DAI (Data Access and Integration). This project does not belong to GGF but has a strong bond with GGF-DAIS workgroup and their solutions have influences on each other. The OGSA-DAI implements many standards and is available to both environments: OGSI and WSRF. The databases currently supported by the OGSA-DAI are: Microsoft SQL Server, Oracle, PostgreSQL, MySQL and IBM DB2, apart from Apache Xindice, and binary XML files [OGSA-DAI-WSRF 05].

There are several middleware to implement computational grids. In this work Globus Toolkit 4 was adopted and its major characteristics are related as follows in the next topic.

2.1. Globus Toolkit 4 (GT4): infrastructure to distributed spatial databases

The Globus Alliance's product, Globus Toolkit, is a middleware to make grid infrastructures. The recent version 4 is based on stateful Web services that can increase the possibilities and the integration level of heterogeneous environments. Its tools came from the old versions like GRAM, MDS, RLS and GridFTP, although some of them are not supported by Web services such as in the case of GridFTP, RLS (Replica Location Service) and MyProxy [Foster 2005]. The product displays a high degree of adherence to GGF standards and some solutions found by Globus Alliance were adopted by the Global Grid Forum.

Its powerful set of tools allows the customization of a computer grid environment. Some important components delivered with Globus are presented below.

WS-GRAM (*Grid Resource Allocation and Management Service*) – This is the job execution management of GT4. It allows submitting a job and monitoring its life cycle by means of a handler named EPR (*End Point Reference*). With EPR it is also possible to kill a submitted job execution. This version supports parallel jobs, multi-jobs and job and process *rendezvous*.

WS-RFT (*Reliable File Transfer Service*) - controls and monitors multi-file transfers using GridFTP.

WS-MDS (*Monitoring and Discovery Service*) - Offers means of registering and querying resources besides mechanisms to read its properties. Lifetime of resources can also be set.

GridFTP – This is an enhanced FTP server to support grid operations under high performance throughput. It is a pre-web service component, but services related to WS-RFT can be used to indirectly manage the GridFTP.

RLS (*Replica Location Service*) – It is another pre-web service component and its functions are to register and discover information about replicated files.

All components that support Web services can be deployed in Java, C or Python containers.

Most of its new services are incompatible with their equivalents in previous versions. A recent study made by the UK Task Force [Harmer et al. 2005] underlines the stability and performance of this release and reveals many others positive aspects observed after many tests were performed.

3. Databases and grids

The use of databases in computational grids is a great challenge to developers of grid infrastructures and databases.

The challenge aggregates all the complexities inherent to distributed databases with the needs of integration of heterogeneous resources in a weakly coupled environment.

Security questions assume an important position since there are no boundaries to the several organizations that participate in grid. They form so-called virtual organizations, an abstract entity that could have a huge amount of resources shared by the members.

Despite all the adversities, the capability to share data, information knowledge and database skills over the grid is a strong motivation to establish interface standards to allow access, discovery, monitoring and integration of data.

Either to reach performance or to access a large amount of data stored in widely distributed databases, some research centers have adopted their own architecture to grid databases. This is the case of the ASTRON Astronomical Institute in the Netherlands which used an architecture based on an object-oriented database, in a high performance grid of the LOFAR project [Risch et al. 2002]. This project intends to process a very large amount of data acquired from space by means of thousands antennas spread in a region of about 350 km of diameter. Another research center that took such initiative is the CERN (European Organization for Nuclear Research) [Stockinger 2001], which also adopted a solution based on an object-oriented database to solve problems in High Energy Physics (HEP) where a solution involving grid and databases was needed.

An efficient structure to replicate and promote the transfer of huge amounts of data amongst nodes in a grid must be provided depending on application type.

As explained above, the GGF lists several services related to data access and integration, but these services are not enough to support all kinds of functionalities needed by a DBMS being heavily based on relational and XML databases [Malaika et al. 2003].

4. Issues on Distributed Spatial Database Management Systems

A Distributed Spatial Database Management Systems (DSDBMS) is a structure made by a set of Spatial Database Management System (SDBMS) spread by many sites and connected by a communication medium. There are several complex issues related with these systems that should be underlined: complexity in processing distributed query efficiently, interoperability amongst SDBMS, management of systems with different levels of autonomy, the design of a distributed database, objects that trespass servers and distributed control [Ramirez 2001].

Spatial data are in most of times complex and extensive and the solutions used to solve problems like interoperability and heterogeneously in a relational model are not adequate for a DSDBMS.

A DSDBMS project can be made by two forms: top down or bottom up. The first form occurs when a new project is being created and all SDBMS are tailored to this specific purpose. The second form is used when the project already start harnessing existent SDBMS. With the top down approach problems related with interoperability and heterogeneously are easily solved, while that using a bottom up form, these problems can assume a major importance.

Despite relational database management systems are being widely used to build DSDBMS, they are not the most appropriate. Extensible systems [Carey and Haas 1990], toolkits and object-oriented database were used to these kinds of non-conventional applications.

In this work a different kind of extensible database is being used: Secondo. Secondo is as comfortable to extend as any object-relational system and almost as flexible as a database toolkit: an extensible algebraic term execution engine with support for persistent objects [Dieker and Güting 2000].

A well structured modular architecture offers specifics resources to handle spatial and temporal data. Secondo is being used as a versatile SDBMS in this work.

A short overview of Secondo is presented in next topic.

4.1. Secondo: an environment to build spatial databases

Secondo is a DBMS developed at the Fernuniversität Hagen to allow: building research prototypes, teaching architecture, and implementing database systems [Güting et al. 2004b]. It is an extensible platform and is not based on a specific data model. By means of second-order signature [Dieker and Güting 2000] it is possible to implement support for any data model, such as relational, object-oriented, XML, spatial and temporal. Its basic semantic module is named algebra. Algebra consists of types and rules, and can be built to a specific knowledge domain. Today there are over twenty algebras whereas others can be created to satisfy specifics needs. However, two of them should be emphasized: the spatial algebra and the temporal algebra that can be directly used when Secondo is acting as a SDBMS.

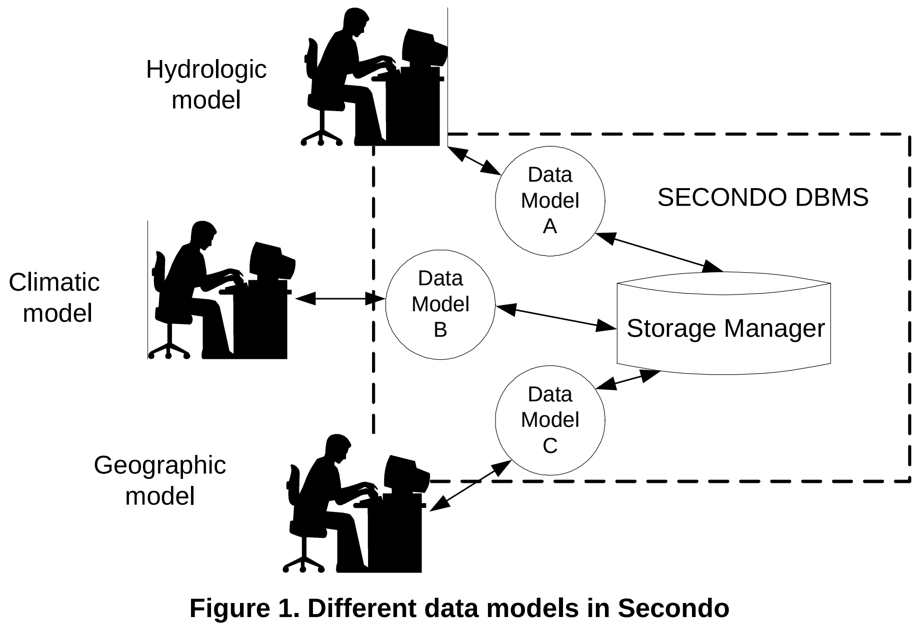


Figure 1. Different data models in Secondo

The DBMS environment is made by three basic modules: optimizer, graphic user interface (GUI) and kernel, each of them able to work independently from the others.

Users can access the DBMS via three paths: via graphic user interface, via optimizer or directly through the kernel (Figure 2).

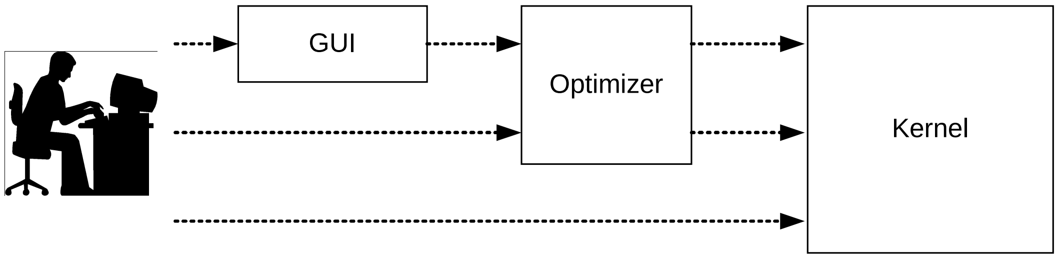
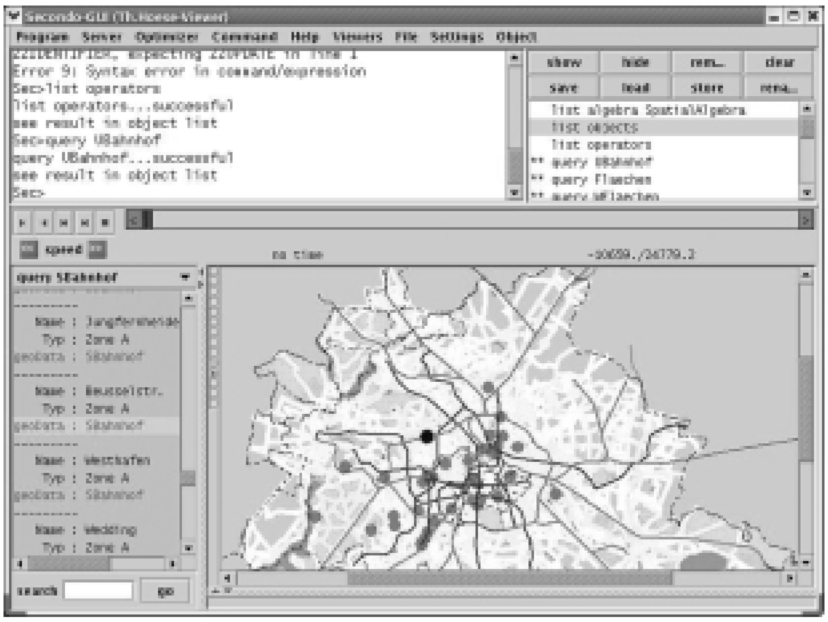


Figure 2. Three access forms to Secondo

Optimizer is responsible to find the best query plan from an user input and to offer it in an adequate format. The optimizer that comes with the full version was developed for relational algebra and a specific one must be developed for each new algebra created. The language to write optimizers is Prolog.

The graphic user interface (GUI) was developed in Java and has the advantage of graphically displaying many kinds of data already defined in native algebras. It can be used separately to browse spatial or spatio-temporal data in external files. It can be extended to support new kinds of data.


Secondo-GUI (Ti-Hoese-Viewer)
Program Server Optimizer Command Help Viewers File Settings Object
show hide rem... clear
save load store rena...
speed
no time
-30659./24779.2
query SBahnhof
search go

```
ZZIDENTIFIER, expecting ZZDFINE in line 1
Error 9: Syntax error in command/expression
Sec>list operators
list operators...successful
see result in object list
Sec>query UBahnhof
query UBahnhof...successful
see result in object list
Sec>
```

```
   list algebra SpatialAlgebra
   list objects
   list operators
** query UBahnhof
** query Flaechen
** query WFlaechen
```

```
---------
    Name : Jungfernheide
     Typ : Zone A
geoData : SBahnhof
---------
    Name : Beusselstr.
     Typ : Zone A
geoData : SBahnhof
---------
    Name : Westhafen
     Typ : Zone A
geoData : SBahnhof
---------
    Name : Wedding
     Typ : Zone A
```

Figure 3. The Secondo GUI

Finally the kernel, that is the core of Secondo, is divided roughly in four modules.

Command manager – is a procedural module and data model independent. After processing, a command redirects the data flow to the query processor, to the storage manager or to the optimizer;

Query processor – Works together with functions from algebra modules when executing a query;

Algebra – Made up by several independent modules based on SOS rules and can use tools like data structures and some functions to manipulate data that belong to the next block. An Algebra has a precise data model and a correspondent query language;

Storage Manager and Tools – This module aggregates many auxiliary tools like interfaces for the storage manager, parsers, catalog, etc.

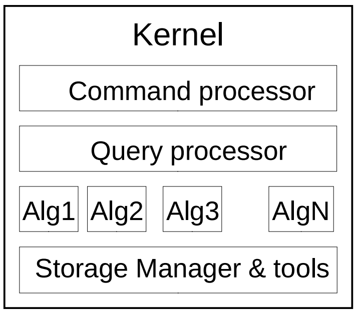
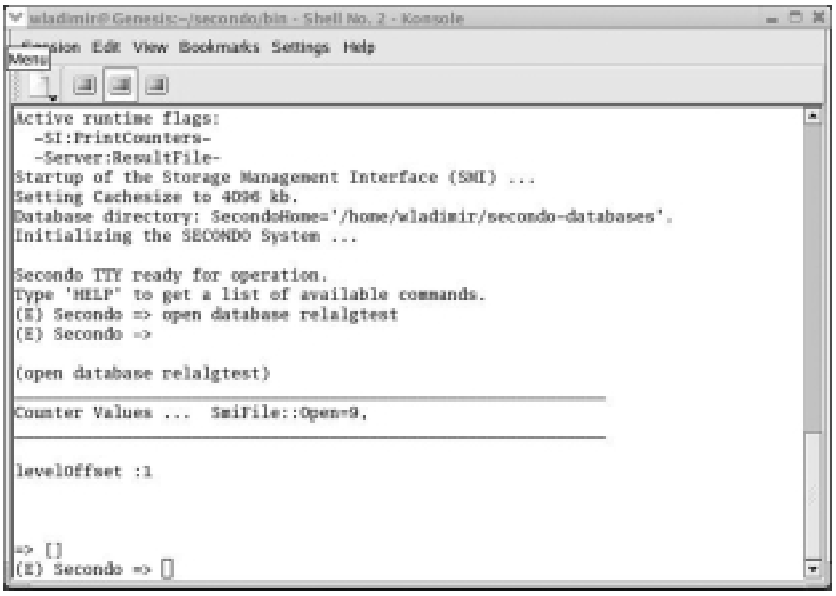


Figure 4. Rough kernel structure [Güting et al. 2004a]

The kernel was written in C++ and has Berkeley DB on a role of storage manager.

It is possible to work with Secondo in a stand-alone or client-server manner and its native operational system is Linux.


wladimir@Genesis:~/secondo/bin - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
Menu

```
Active runtime flags:
  -SI:PrintCounters-
  -Server:ResultFile-
Startup of the Storage Management Interface (SMI) ...
Setting Cachesize to 4096 kb.
Database directory: SecondoHome='/home/wladimir/secondo-databases'.
Initializing the SECONDO System ...

Secondo TTY ready for operation.
Type 'HELP' to get a list of available commands.
(E) Secondo => open database relalgtest
(E) Secondo ->

(open database relalgtest)
_____________________________________________________________
Counter Values ...  SmiFile::Open=9,
_____________________________________________________________

levelOffset :1



=> []
(E) Secondo => 
```

Figure 5. Textual interface of Secondo

When optimizer is being used, users can input SQL directly in the interface; otherwise a SQL like language, inherent to the Secondo environment, should be used.

Secondo has many mechanisms that can be used to study databases behavior under many circumstances. The second-order signature (SOS) is the main concept behind Secondo, because it can supply all needed formalism to support the whole system. SOS is made by two coupled signatures as follows:

The first signature is made by *kinds* and *type constructors*. *Kinds* are sets of types that make up a data model of a Secondo DBMS. *Type constructors* are operators over the *kinds*. With this structure *type expressions* can be created to generate types as results. The goal of the first signature is to specify a data model;

The second signature is formed by types created by the first signature and operators that are used to create value expressions that offer values as results. The goal of this signature is make up a query algebra.

Type constructors and operators in an algebra have data structures associated to help optimization and statistical analysis.

A Secondo DBMS has a standard set of commands independent from the data model that is used for common operations [Güting et al. 2004a].

With the modularity of Secondo's architecture and with SOS formalism, a wide range of application areas can take advantage of its extensibility.

5. Related Work

Efforts have been made to integrate the power of grid computing to databases. With this association, the early computing centered approach was expanded with a data-centric vision. *Distributed Query Processing on the Grid* (DQP) [Smith et al. 2002] is a project involving grid infrastructure based on the Globus middleware, relational databases and parallel-object database servers. Its architecture is centered on a coordinator node, responsible for the construction of query plans and storage of a global schema of a federated environment. Information on grid resources is also stored in this node. Communications amongst nodes runs over Globus using MPICH-G and its infrastructure is based on the OGSI (*Open Grid Service Infrastructure*) standard. The DQP project has some likeness with the current proposed framework. Both differ in the sense that DQP used a central node in an asymmetric architecture while in this job is assumed a symmetric structure where all nodes are equal. The DQP is based on the OGSI standard and on relational databases while here the WSRF is used in its place and an environment based in a non-conventional database (Secondo)

CoDIMS-G: a Data and Program Integration Service for the Grid is a solution proposed in [Dutra et al. 2004], to promote data and program integration, in an environment where this data is distributed over relational databases. The system is also based in OGSI standards and was created to offer support to preprocessing stages of scientific applications where the processing power of a grid can reduce the time spent in these phases.

Monitoring Data Archives for Grid Environments [Lee et al. 2002] is a work developed for the use of relational databases in a computer grid to store information that could be used to analyze grid performance and its bottlenecks. It uses *Grid Monitoring Architecture* (GMA) as defined by the Global Grid Forum's Grid Performance Working Group. The system consists of applications that were modified to produce monitoring data (information about context) to be sent to a specific storage module that act as a consumer for the data received. Those applications can have these extra skills enabled or disabled when necessary to monitor data production. The information stored can be used to analyze grid behavior, and to find critical points in the structure. The Secondo-grid framework has the capacity to implement several of the mechanisms addressed in the previous work, but the emphasis should be in the use of stateful Web services and WS notifications resources.

6. The Framework proposed: Secondo-grid

An SDBMS with the native capability to interoperate with a grid environment is the main goal of this framework.

To reach some of the desirable qualities of a distributed system like interoperability, performance, support for cooperative work groups and expansibility [Özsu and Valduriez 1999], a framework was proposed to integrate the best properties of computational grids and the flexibility of a extensible database.

With this proposed environment it is possible to implement several experiments involving distributed databases in a computational grid and, particularly, experiments with Distributed Spatial Database Management Systems (DSDBMS). Different kinds of distributed database architecture like integrated, federated and multidatabase can be explored too.

Some of the several possibilities are enumerated here:

- Performance evaluation of the grid environment under distinct architectures, using classical mechanisms [Lee et al. 2002] or specialized algebras developed to reach this goal;
- Tests using Secondo-grid as a gateway amongst several kinds of databases, exploring its different data models;
- Access specialized Secondo servers to take advantage from their personalized set of algebras, either to import some algebra or to move data to be processed remotely;
- Make tests with distributed query in parallel or in cascaded form;
- Use the awareness of context, based on resources status, to plan the queries. In this case, information like CPU load, available amount of memory, performance of communications channels and the number of users accessing a database, could be used to many purposes.

The standards recommended by GGF-DAIS do not support data models apart from relational and XML, and in this case, a full implementation of DAIS services in Secondo can not support non-conventional data models implemented in this DBMS.

With the implementation of GGF-DAIS services, Secondo could be used as a gateway for other applications based on relational databases that run on a grid, in a transparent manner, but this is not intended to be implemented yet in this framework.

Short overviews of Secondo DBMS and Globus Toolkit 4 are needed before explaining framework architecture. Further information on these two environments can be found in the references.

6.1. Secondo-grid description

To permit Secondo act as a SDBMS in a grid environment, a new operation mode was created: the grid mode. This innovation enables access to a computational grid by means of new methods that use web services' clients to request information or to submit a job to the grid (Figure 6).

The functionalities introduced are capable of proceeding with three kinds of services: global query submission, algebra discovery and remote resource monitoring. They were added to Secondo's tools layer and have the following meaning (Figure 7):

globalQueryPlanProcessor() – this is the main method used when submitting a query to the DSDBMS. It is subdivided in two blocks: query plan maker and query execution monitor. In the first block a global query is transformed in sub queries after consult the global schema, the fragments' locations map and the resources status of nodes involved with the query. Second block is responsible by create a job description file, monitor the status of query execution and by integrate the returned results;

requestGlobalSchema() – a global schema of the DSDBMS in XML format, registered previously with MDS, can be requested with this method;

modifyGlobalSchema() – this method permit modify a global schema after a local change in its structure;

requestFragmentLocation() – request a copy from the fragments' locations map;

updateFragmentLocation() – this method permit update changes made in fragments locations.

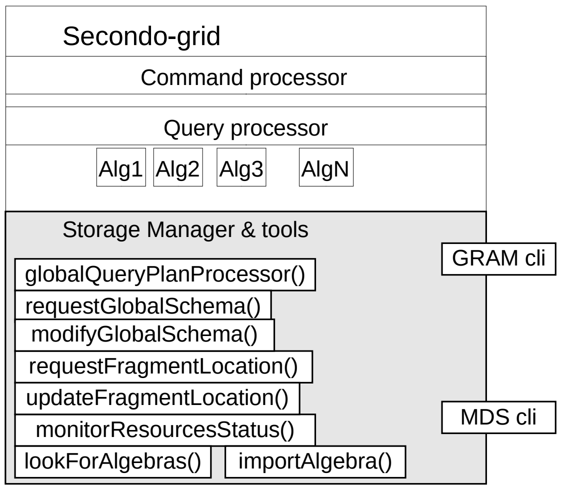


Figure 6. New functionalities in Secondo kernel

All these methods use only GRAM and MDS services from GT4 by means of their respective clients, improving the flexibility of the solution, since none additional web services must be deployed in GT4. Under this perspective a Secondo-grid server can be installed anywhere in the web and can immediately access other Secondo-grids.

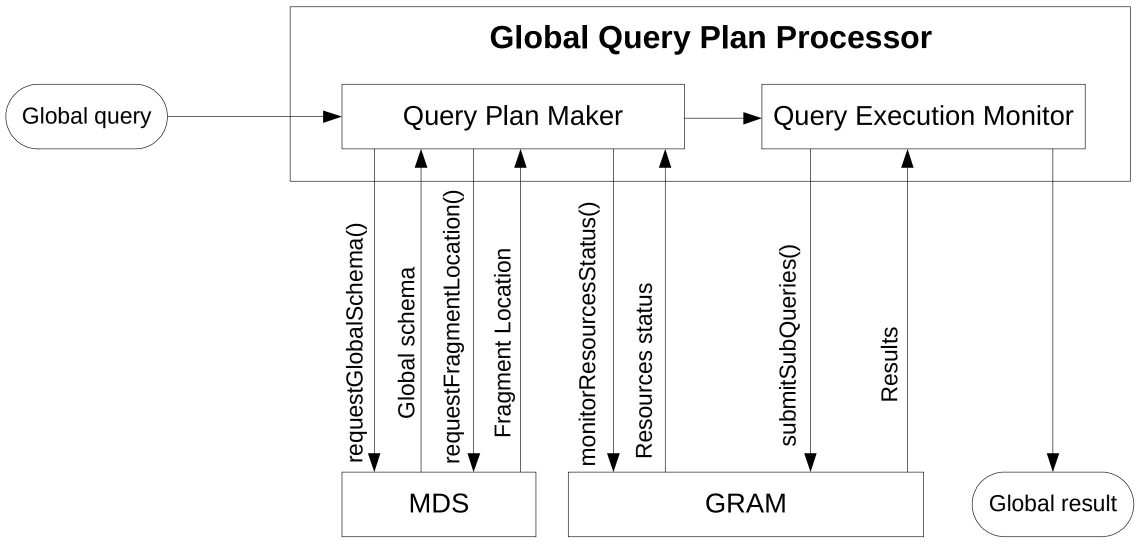


Figure 7. Global Query Plan Processor architecture based (ref. [Ramirez 2001])

When submitting a job of a query processing type to a set of Secondo DBMSs in the grid, a job manager should be used to receive this submission request. The WS GRAM (Globus Resource Allocate and Management supported by Web Services) is a native job manager distributed with GT4 and was used in this framework for its simplicity, although the use of an alternative solution like Condor-G could be the case, especially in complex jobs.

The WS-GRAM, before a job is submitted, should receive a complete job description that consists of a XML file with a great deal of information on the job. The functionalities added to the kernel permit Secondo to automatically generate a job description file.

On the other hand, a query string with all the necessary steps that a remote server should execute must be supplied also in a distinct file named *command file*. When submitting the job (a query to be executed) its description file is read by WS-GRAM which is then informed on the destinations of the command file (Figures 8 and 9).

Due to the size of generated command files, which can be large, a dual channel approach [Lee et al. 2002] was adopted, instead of the use Web services, to exchange data and control altogether. WS-RFT (Reliable File Transfer supported by Web Services) can manage an efficient data transfer via a specialized tool: the GridFTP.

In a remote node, a job resource manager is responsible for promoting job execution. In this framework the *Condor pool* scheduler was adopted and works in conjunction with GRAM.

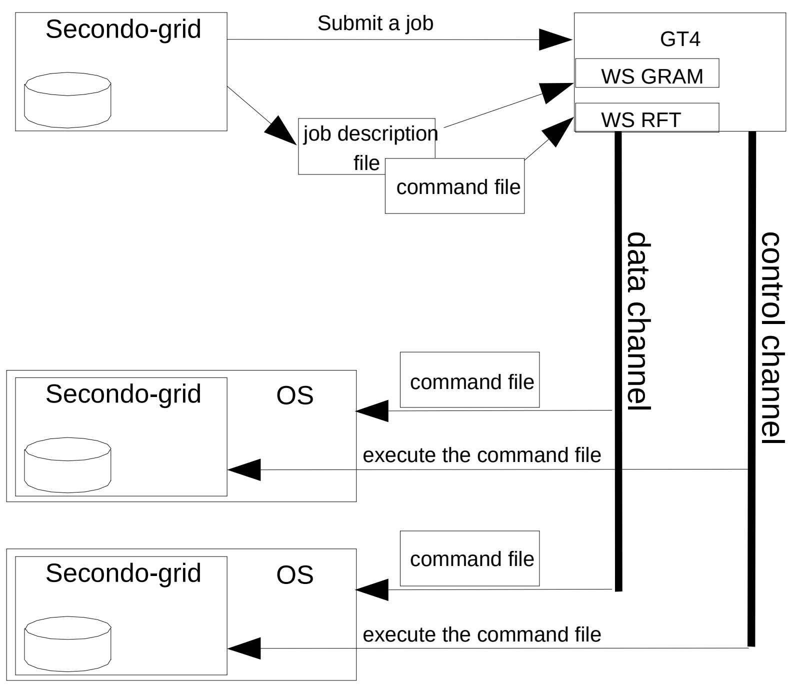


Figure 8. Query processing (phase 1)

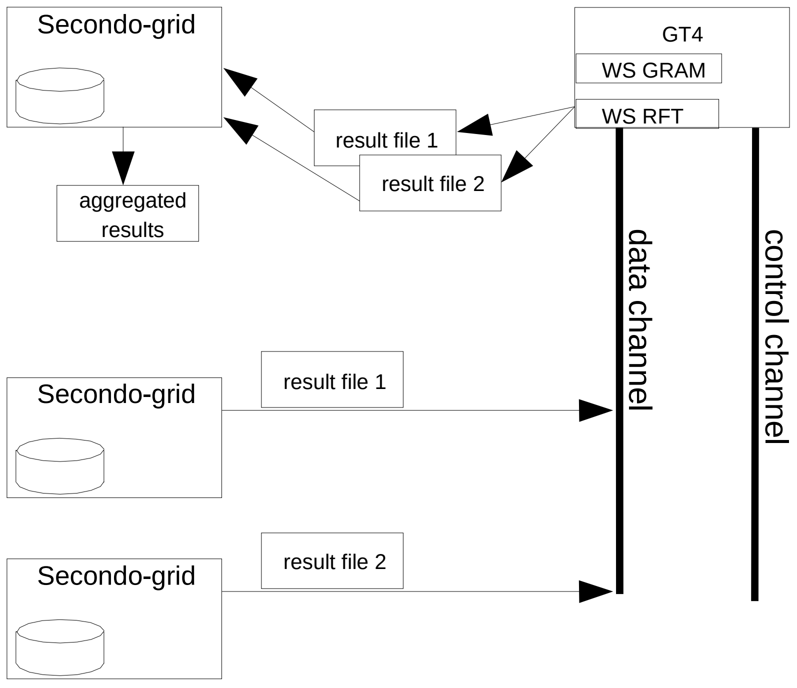


Figure 9. Query processing (phase 2)

It is important to emphasize that when a job is submitted, a handler is created and job status can be monitored. This property is delivered by stateful Web services used by GT4 and is very helpful. This kind of handler is known as an *end point reference* (EPR).

The requester node can submit a query to run in single or multiple Secondo servers at the same time and when submitting multijobs, each one of the jobs generating an EPR.

The framework also includes some specific services to allow:

- Search for new algebras implemented in other Secondo servers;
- Import a specific algebra from a remote Secondo server;
- Monitoring resource status

These services were incorporated in the structure proposed to explore the richness of platforms Secondo and GT4. Following is a short description for the methods that implement these services.

LookForAlgebras() – This method is used to search implemented algebras in Secondo servers of the grid. To reach this goal, all servers must register their algebras with their MDS services. This service returns a list with servers' names and their associated algebras' names. The answer format is:

(*server-name   alg1   alg2   alg3 …*)

ImportAlgebra(*server-name, algebra-name*) – This method should be used after a specific algebra has been found by the *LookForAlgebras()* method. *ImportAlgebra* passes all needed information to its local RLS (*Replica Location Service*) so that the files related with that algebra could be imported from a Secondo server. The sequence needed to import the files is: local RLS request to the addressed LRC (*Local Replica Catalog*) all files locations assigned to that algebra. With this list, the RFT service proceeds with the transfer via a GridFTP session. At the end, the files belonging to the algebra are positioned in their correct places and the Secondo must be recompiled to activate them.

MonitorResourcesStatus(*monitor-program, server-list*) – With this service the status of previously selected resources can be monitored and sent back to the client. Such information can be used when making a more complex query plan. This service also uses the GRAM and the RFT services (Figure 10). An autonomous and short program should be done to inquire, in each addressed server, information such as the amount of available memory, amount of free disk space and CPU load. Any kind of resource can be monitored (hardware and software), but specific programs or script files should be built. Once the monitoring application is defined, it is sent to the servers via RFT and the results returned, after acquired, to the client node.

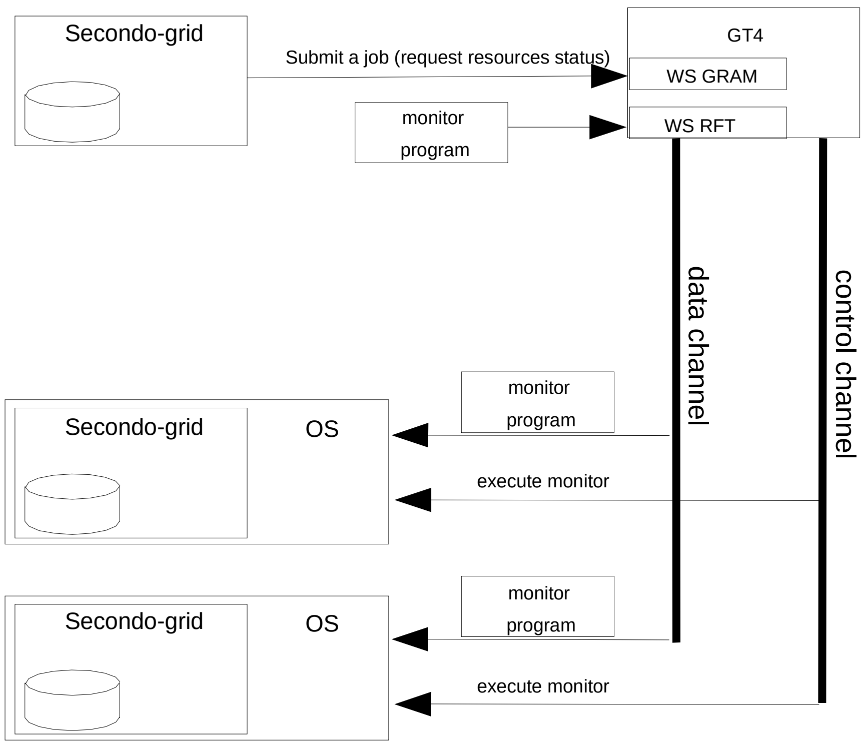


Figure 10. Monitoring resources

7. The study case description

A study case is needed to test the service-based framework and to analyze its performance when working with a distributed geographic database. To reach the desired goals, it is being built with the following characteristics:

Composition – It is composed by a computational grid with four computers running GT4 and Secondo-grid over fedora Linux core 3. GT4 was installed with GRAM, MDS, RFT and GridFTP services. Condor pool is being used as scheduler on each node.

Distributed spatial database design – With the purpose of test some kinds of queries, the database was developed in a top-down form within a federated architecture, where a global schema is coded in a XML file and registered as a grid resource to be easily accessed by any database node of the grid. It was adopted a thematic fragmentation schema and all themes belongs to a common region. However it is possible to have a specific theme replicated in more than one server (Figure 11). This was done for tests purpose only.

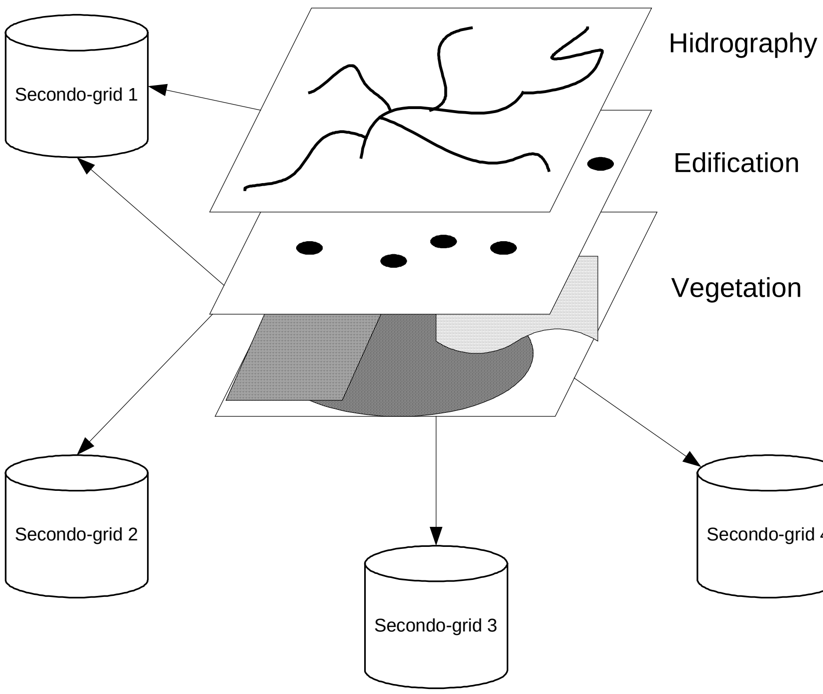


Figure 11. Locations of thematic' fragments

A precise fragments' locations map must be kept in the DSDBMS environment and must be registered as a grid resource like the global schema.

Autonomy – All SDBMSs (Secondo-grids) have a moderate degree of autonomy, but the cost of this freedom is to keep updated both global schema and fragments' locations map.

Nature of data used with the study case – The data being used consist of cartographic data obtained from Directory of Geographic Service/Brazilian Army. The themes being used are: hydrograph, edification and vegetation. They were adopted because of the nature of their geometries that were considered of three types: lines, points and regions, respectively.

Queries to be implemented – To validate the framework, two kinds of simple tests are being built: a spatial select and a spatial join based in a first instance by an ingenuous strategy. An overview of both strategies can be seen bellow:

Select

Read the global schema

Read the fragments' locations map

Read resources status from nodes with fragments involved in the query

Select the nodes with best conditions in case of a replicated fragment

Break the global query in sub-queries

Generate a job description file and submit the job to GRAM

Receive and integrate results to generate a global result

```
Join
Read the global schema
Read the fragments' locations map
Read resources status from nodes with fragments involved in the query
Select the nodes with best conditions in case of a replicated fragment
Break the global query in sub-queries
Estimate cardinality of sub-queries
Build a job description file that determines sub-queries execution in an adequate
   order: sub-queries with smaller cardinality at first
Submit the job to GRAM
Transfer the results of these first sub-queries to nodes where the last stage of the
   queries should be executed as a local query in a SDBMS environment
     (ingenuous approach).
Transfer the final results to the original node and delete all temporary files.
```

The program used to monitor the status of resources can be built of several manners. For this study case is being used a script file with Linux commands to detect: CPU load, amount of free memory, amount of free space in hard disk.

With this arrangement, join and selection of operations will be executed and information on their performance will be analyzed. It should be emphasized that the state of some resources will be monitored with the purpose of making a better choice about where and how to run a job (query in this case).

The expansibility of this structure is high, since modifications to global schema and fragments' map are made in an unique place and at any time a new Secondo server can request a copy of this schema to interact with the DSDBMS. An interesting resource delivered by the MDS is the possibility of assigning a lifetime to registered resources. With this it is possible to destroy all copies of an obsolete schema after a stipulated time, forcing a refresh after a new one is registered.

8. Conclusion

The framework Secondo-grid is being built by over three months and there are not numeric results to be analyzed yet. However we expect to have some relevant results until the end of 2005. The proposed environment can be used to implement new solutions involving distributed spatial database management systems as for example those related with spatial join.

The flexibility offered by Secondo in sense of permit the use of specifics data models, like temporal, can open a new range of possibilities, enabling for examples researches with mobile objects in DSDBMS environments.

On other hand the power of GT4 middleware, increased by the use of stateful web services, can permit an easily access and integration with servers around the web. Its high degree of compliance with the GGF's standards, can ease future integration with other resources in the grid. An implementation of GGF-DAIS structure, as a layer, can be stated as important goals to future works.

The main contribution of this work is the proposal of a symmetric architecture based on grid services built with an extensible database and a configurable grid middleware with purpose of study DSDBMS in a computational grid. The non-conventional nature of the Secondo permit the development of specific algebras to support specifics needs. When registered, these algebras can be shred by all SDBMS from the grid, permitting

9. References

OGSA-DAI-WSRF, http://www.ogsadai.org.uk/docs/WSRF1.0/doc/background/supported.html, accessed on 6-9-2005

GGF Areas/Groups, http://www.ggf.org/ggf_areasgrps_overview.htm, accessed on 6-9-2005

Carey, M., and Haas, L. (1990) "Extensible Database Management Systems", volume 19 nº4. SIGMOD.

Dieker, S., and Güting, R. H. (2000). "Plug and Play with Query Algebras: SECONDO A Generic DBMS Development Environment". In *International Database Engineering and Application Symposium*, Hagen, Germany, 380-392.

Dutra, M., Porto, F., Barbosa, A., Fontes, V., and Schulze, B. (2004). "CoDIMS-G: a Data and Program Integration Service for the Grid." ACM, Toronto, Canada.

"What is the grid? A three point checklist."(July 22). *Grid Today*, July 22.

Foster, I. (2005) "A Globus Toolkit Primer Describing Globus Toolkit Version 4 - Draft".

Foster, I., and Kesselman, C. (1999). "Computational grids". In *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman.

Foster, I., Kesselman, C., and Tuecke, S. (2000) "The Anatomy of the Grid Enabling Scalable Virtual Organizations".

Güting, R. H., Ansorge, D., Behr, T., and Spiekermann, M. (2004a) "Secondo User Manual". Hagen, Germany, Fernuniversität Hagen.

Güting, R. H., Behr, T., Almeida, V., Ding, Z., Hoffmann, F., and Spiekermann, M. (2004b) "Secondo: An Extensible DBMS Architecture and Prototype". Hagen, Germany, Fernuniversität Hagen.

Harmer, T., Stell, A., and McBride, D. (2005). "UK Engineering Task Force Globus Toolkit Version 4 Middleware Evaluation.".

Lee, J., Gunter, D., Stoufer Martin, and Tierney, B. (2002) "Monitoring Data Archives for Grid Environment". IEEE.

Malaika, S., Eisenberg, A., and Meltron, J. (2003). "Standards for Databases on the Grid.".

Özsu, M. T., and Valduriez, P. (1999). "Principles of Distributed Database Systems." Prentice-Hall.

Ramirez, M. R. (2001) "Spatial Distributed Query Processing". Rio de Janeiro, RJ, COPPE/UFRJ.

Risch, T., Koparanova, M., and Thidé, B. (2002). "High-performance GRID Database Manager for Scientific Data." Paris, France, 99-106.

Smith, J., Gounaris, A., Watson, P., Paton, N. W., Fernandes, A. A. A., and Sakellariou, R. (2002) "Distributed Query Processing on the Grid".

Sotomayor, B. (2005) "The Globus Toolkit 4 Programmer's Tutorial".

Stockinger, H. (2001). "Distributed Database Management Systems and the Data Grid." CERN (European Organization for Nuclear Research), San Diego.