VIII Brazilian Symposium on GeoInformatics, Campos do Jordão, Brazil, November 19-22, 2006, INPE, p. 91-108.

Spatial Query Broker in a Grid Environment

Wladimir S. Meyer[1], Milton R. Ramirez[2], Jano M. Souza[1]

[1] Computer Science Department of
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 *Grid computing can be seen nowadays as a promising distributed environment to form federated GIS. As supposed, spatial query processing within this distributed environment is a challenge. This paper presents Spatial Query Broker (SQB), an architecture for spatial distributed query processing in a grid computing context. This implementation aims at executing spatial queries in a dynamic manner.*

1. Introduction

Lately, the profusion of equipment related to spatial data generation has been responsible for the production of a lot of spatial data to attend different purposes in many domains. Government agencies, large private corporations and scientific centers are the most common producers and consumers of this kind of data that are employed in decision-making systems, analysis tools and experiments.

Many efforts have been made to create standard architectures capable of offering several levels of relationship among spatial data producers and consumers. The Open Geospatial Consortium (OGC) is one of the most important organizations involved with the standardization and its interfaces and services have been recommended as a solution for interoperability problems over a distributed environment. Even though OGC standards have been adopted by many systems, as a solution for uniform spatial data access, there is a lack of mechanisms capable of leading them towards the high performance direction, as in services related to data transfer and spatial data processing among servers (Câmara and Queiroz 2002). A typical question is "how to dynamically execute spatial operations involving features from different data providers?" In these situations the client application is responsible for taking care of the entire process after accessing the remote spatial data. So all complexities must be embedded in the client application, which doesn't necessarily lead to an efficient result.

The grid computing paradigm gathers many capabilities that could be employed in the aforementioned context. It has emerged with the purpose of sharing free resources beyond organizations' boundaries, aimed at a high performance infrastructure.

This work aims to provide a strategy to execute spatial joins among large spatial databases spread geographically and integrated by a grid environment. This strategy is implemented as a spatial query broker being capable of receiving a spatial query and selecting computer nodes dynamically to process parts of the original query in parallel in order to reduce its overall execution time. The architecture was influenced by

previous ones presented in (Meyer et al. 2005), (Meyer and Souza 2006) and (Smith et al. 2002).

The structure of the paper consists of an explanation of problems related to distributed spatial queries (section 2) and some solutions based on filter/refine strategy, since they play an important role in the GIS domain, followed by an overview of Virtual Organizations in a grid context (section 3) as a metaphor, emphasizing their aspects and functionalities that could be applied to a distributed GIS context. A description of resource brokers (section 4) means to depict important functionalities that could be used in a spatial query broker. Some related works involving distributed spatial query processing are presented in section 5 and the proposed architecture is detailed in section 6. Some preliminary results on the parallel execution of spatial queries are analyzed in section 7 and some important remarks are taken into account in section 8.

2. Distributed Spatial Queries

A filter/refine technique has been adopted in many spatial database management systems (SDBMS) to reduce the exact geometric tests in spatial queries (Hanssen 2005). It consists of the execution of a preliminary query with approximations of the actual features (usually Minimum Bounding Rectangle - MBR) to discard some of those that don't satisfy a specific. The next step consists of executing the exact processing of the involved geometries to determine which of them definitely satisfy the predicates.

An important characteristic of spatial queries is that they are much more expensive than those involving conventional data. The size and number of vertices of geometrics' data are the main factors. The queries normally use large computing resources when executing operations with geometrics and/or topologic predicates, turning indispensable the use of spatial indexes.

This technique was extended in (Brinkhoff et al. 1994) to process spatial joins, including a geometric filter step after a MBR join. This new step introduces a more accurate approximation of spatial objects, to reduce the size of inconclusive pairs.

Some strategies were proposed in (Ramirez 2001), based on Brinkhoff's model to deal with spatial joins in a distributed environment trying to explore an additional parallelism among SDBMS involved during the exact processing step.

Trying to improve response time a proposal is presented in (Kang and Choy 2002) where the authors aim at executing the two steps of the filter/refine technique in parallel, but details about preliminary considerations limit the scope of the proposal: high bandwidth and thematic data partitioning.

In spite of the good results presented by previous solutions, a small number of servers is involved during the exact geometry processing, normally those storing the themes, which can lead to inadequate response time when dealing with very large data sets.

3. Virtual Organizations in a Grid context

When organizations, computational resources, services and people with similar interests are put together and the sharing rules is defined, they originate a so called *Virtual Organization (VO)* (Foster et al. 2001). The VO is an abstraction that can involve many actual organizations interested in collaborative work, i.e., they can share services, data

and computational resources to reach some common goal. This abstraction could be adopted, for example, by organizations interested in sharing spatial databases or making up a distributed GIS.

A VO often presents some basic functionality items that may be accessed by any member nodes. Some of the most important are:

- File catalog – It consists of a service specialized in registering and monitoring file replicas spread among nodes and that may have distinct logic names;
- Job manager – Entity that receives jobs execution requests and verifies all necessary resources for their execution;
- Resource broker – A service capable of checking all necessary resources described in a job specification and invoke all those resources to perform its execution;
- Information service – A suite of services that gathers information about resources and are often used by applications and other services (like the resource broker).

The profile of a VO can change as a result of its purpose. Services, resources and topology may assume different configurations depending on the nodes' capabilities, the nature of the tasks to be performed and the interests of the members.

Despite the flexibility and the computing-on-demand offered by the VO abstraction, some difficulties like heterogeneity and security, are great challenges for the research carried out (Porto et al. 2005; Foster and Kesselman 1999).

The adoption of standards like OGSI (Open Grid Service Infrastructure) and, more recently, WSRF (Web Service Resource Framework) has brought a new perspective to the VO environment since now it can deliver services with a high degree of standardization for the use of Web services technology. Security problems related to the data and binary transmission may be avoided with the use of binaries encapsulated into services that offer only a standard interface, preserving code detail. In a similar way, the heterogeneity problem inherent to a grid environment can benefit from these standard interfaces since the details related to service implementation are omitted from the consumers of the resources, improving both integration and scalability as a whole.

So, as depicted above, a VO takes an important role in a grid environment and should provide many high level services in order to permit the increase of collaboration among its members. Resource brokers take an important place in this context, especially when the application's nature involves hundreds or thousands processors nodes and resources.

4. Resource Brokers (RB)

In an environment, with thousands of resources dynamically changing, it is almost impossible for a user to choose the best resource to execute a job without any previous information. The RB acts as a middle tier between users and resources (Afgan 2004) and assumes the responsibility of finding the best resources for them and, complementarily, passing the job specification to the resources found (Figure 1).

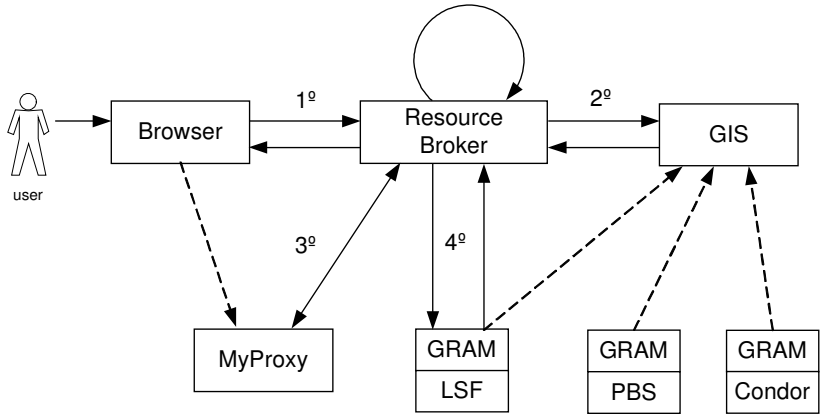

Figure 1. Resource Broker acting as a middle tier in a grid - adapted from (Afgan 2004)

In this figure, the GIS module means "Grid Information Service", the component responsible for advice on resources available. The resource broker - RB - uses this information to locate the best resources without explicit users' participation.

Resource brokers are not present natively in all grid middleware, but many initiatives have generated products that turn the grid user's life easy. GridBus (Venegupal et al. 2004), Emperor (Adzigogov et al. 2005) and GridWay (GridWay Team 2006) are some of these efforts. They can be thought as high level services that locate resources that agree with a job specification.

The role of scheduler may also be taken up by the RB, who makes execution plans and defines the order in which tasks are to be carried out based on complementary information like resource capabilities, communication costs and other statistic metrics. For this ability, RBs are also known in grid platforms as "meta-schedulers".

When assuming a scheduler's role, problems concerning the multi-organizational nature of the grid take place: difficulty to pre-allocate, load balancing and different performance for communication channels, besides the natural resources' heterogeneity (Foster and Kesselman 1999; Foster et al. 2001).

There are two basic strategies adopted by resource brokers to create an execution plan when acting as a scheduler: static and dynamic strategies. The first one defines the execution plan before runtime and any context changes occurred during runtime are not taken into account; an example of this class of RB is the WMS (Workload Management System) (Andretto 2004) which integrates the gLite grid middleware. The second one starts with a predefined plan that can change during runtime, in an adaptive manner, chasing the best results as a whole. GridWay RB (GridWay Team 2006) lies in this class.

Scientific applications, the greatest users and motivators of the grid paradigm, are strongly based on file as the lowest data unit (Di et al. 2003). This implies that most of the RBs are tailored to this data approach.

This paper proposes an similar service to deal with scheduling of spatial queries among spatial databases systems in a grid, similar to the job/file approach of conventional RB. The *Spatial Query Broker (SQB)* being proposed is presented in section 6.

5. Related Works

During last years many efforts have been made to permit executions of spatial queries in a distributed environment.

An extension of Brinkhoff solution, for the distributed spatial query processing was proposed by (Ramirez 2001) with the purpose of exploring the parallelism in the most critical phase of a spatial join process: exact geometry processing. In that work, after the elimination of the geometries that definitely don't participate from the final result, by means of approximate filtering, the inconclusive geometries are simultaneously processed by the SDBMS that participate of the query. This strategy was called MR2 and uses spatial indexes (R*-Tree), from themes involved in the operation, during the filtering step. In this proposal only execution of a query is taken into account. This adaptation of the three-step approach is followed by the present proposal since the filtering phase may lead to a strong reduction of inconclusive pairs and provide some true hits without the need of actual geometry processing.

The use of *Web services* as seen in (Ilya et al. 2003) was a powerful solution the authors found to reach scalability in an infrastructure designed to optimize query processing in distributed spatial databases. The scenario presented consists of independent organizations that produce data which may be overlapped geographically. This data is not intentionally replicated over the member nodes. A global index, based either on R-tree or Quadtree, is maintained in each node during all the time and if some localized data modification causes a change in its minimum bounding rectangle (MBR), all index replicas are updated. A query submitted to a node is then forwarded to other nodes that have data involved with it. The main goal was to reduce the traffic among nodes, improving queries response. The use of a global index distributed over the nodes had an important role in this context. The Web service approach, to easily turn the integration among nodes, and the involvement of several servers to deal with a query are relevant to our proposal.

The *Grid Greedy Node Scheduling Algorithm* (G2N) presented in (Porto et al. 2005) assigns sub-queries to grid nodes as a result of their throughputs expressed in tuples/second. If during runtime the actual throughput of a node differs too much from the previous known one, this new value is used to re-schedule the remaining subqueries. A dynamic load balance, based on processors' throughput, is used in a similar manner on the SQB.

In (Mondal et al. 2003) a dynamic load balancing strategy was adopted to address queries to grid clusters, adopting either a migration or a replication policy for the data, in order to explore the reliable nodes. The knowledge on the historical behavior of the clusters is a good heuristic to be explored when planning the query in the SQB.

OGSA-DQP (UK Database Task Force 2006), a Distributed Query Processor based on the OGSA architecture (a grid architecture based on services), has all the functionalities of a query broker: a *grid distributed query service* is responsible for compiling, optimizing, partitioning and scheduling distributed query execution plans over multiple execution nodes. Its *query evaluation service* is used by the previous service to execute a query plan. Some functional modules from the OGSA-DQP are

being hosted by the SQB. The adoption of the OGSA-DAI, to permit uniform access in databases, is another common point between these architectures.

As pointed above, the spatial query broker's architecture proposed in this paper adopts some guidelines that were present in the previous works and is a consequence of our previous work (Meyer et al. 2005; Meyer and Souza 2006).

6. Proposed Spatial Query Broker (SQB)

In this work a Spatial Query Broker is being proposed to cover the needs of a specialized mechanism to perform queries over SDBMS spread geographically and belonging to member organizations of a VO. The inability of traditional brokers to deal with databases as resources (Andretto 2004; GridWay Team 2006; Buyya and Venegupal 2004), particularly spatial databases, led us to research this topic. There was the need to incorporate in the SQB some functions that are specific of database management systems, like those related with the query processor. On the other hand, as a grid broker, it has to receive others skills from these specialized tools.

The main idea is to explore the skills from SDBMS members of the VO to perform some steps of spatial queries and, when necessary, divide the processing cost of the expensive ones with several grids' nodes, including those that are not specialized (without SDBMS), as depicted in figure 2.

The centralized approach seen in this figure: with central SQB, information service and replica location service are typical in some environments like the OGSA-DQP, Workload Management Service (WMS) and GridWay, these two last schedulers from gLite and Globus projects respectively. However, backup structures could coexist in virtual organizations in order to avoid a service break, in case of failure of the main module.

Following the gLite nomenclature conventions (EGEE 2006), each node capable of processing jobs is named Computing Element (CE). CEs in a virtual organization can have specific skills like high performance hardware or specialized services. The proposed architecture was tailored to treat spatial queries in a similar way as happens with common jobs: a query, after being typed by a user or passed by an application is guided to a specialized broker (spatial query broker).

The architecture (figure 3) can be presented based on a description of its component's modules as follows:

The *Coordinator* module has the role of manage all data flow since a query is received until it is finished. Additional tasks, such as checking an user's credential and starting a proxy session, are also performed by it.

The *Query Decomposition* module has almost all the functionalities found in traditional distributed query processing architectures (Özsu and Valduriez 2001). The first of them is semantic analysis, based on a global schema, of the received query that can be performed by means of graph derived from the query and its analysis: when a node, representing a relation or a sub-graph is disconnected from the result, the query should be refused, since in this case some join predicate is missing. Another functionality is avoiding redundancy in the predicates, through simplifying both the geometric and non-geometric predicates, using the idempotent rules (Özsu and Valduriez 2001).

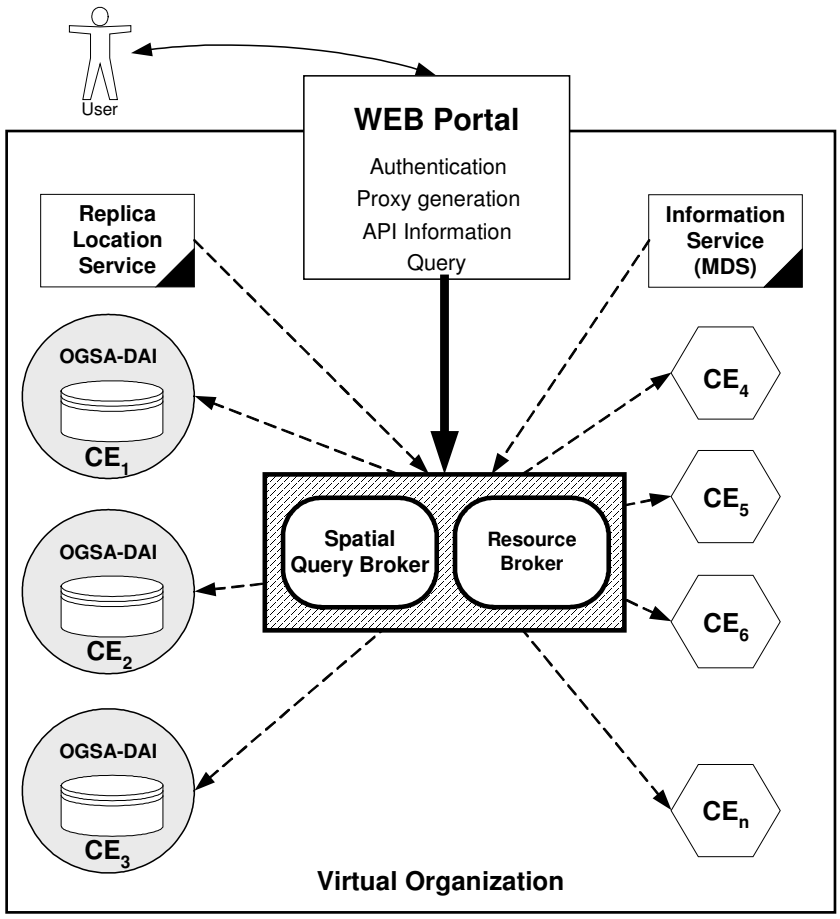

Figure 2. VO composition with a Spatial Query Broker

Locator module acts as a match-maker in a conventional broker (EGEE 2006). Specifically, it is responsible for locate all sources of the themes mentioned in the received query, including their replicas and acquire statistical and metadata information like spatial data quality, databases' status, computing elements' status and their throughputs, besides of communication channels quality. The optimizer uses all this information in order to propose a good query plan and must be periodically updated in a grid information service (Globus MDS in this case). Replicas of databases are managed by the Replica Location Service (RLS), a service that maintain an index of all local replicas' catalogs, hosted on each CE, and permits queries about location of replicas related with a specific logical name.

An Optimizer has the task of dealing with the most common kinds of queries in a spatial database: window queries and spatial joins; based on this premise all its functioning aims at following a good strategy to reach the solution. In both cases the filter/refine strategy has proved to be a good one (Brinkhoff et al. 1994; Hanssen 2005; Ramirez 2001).

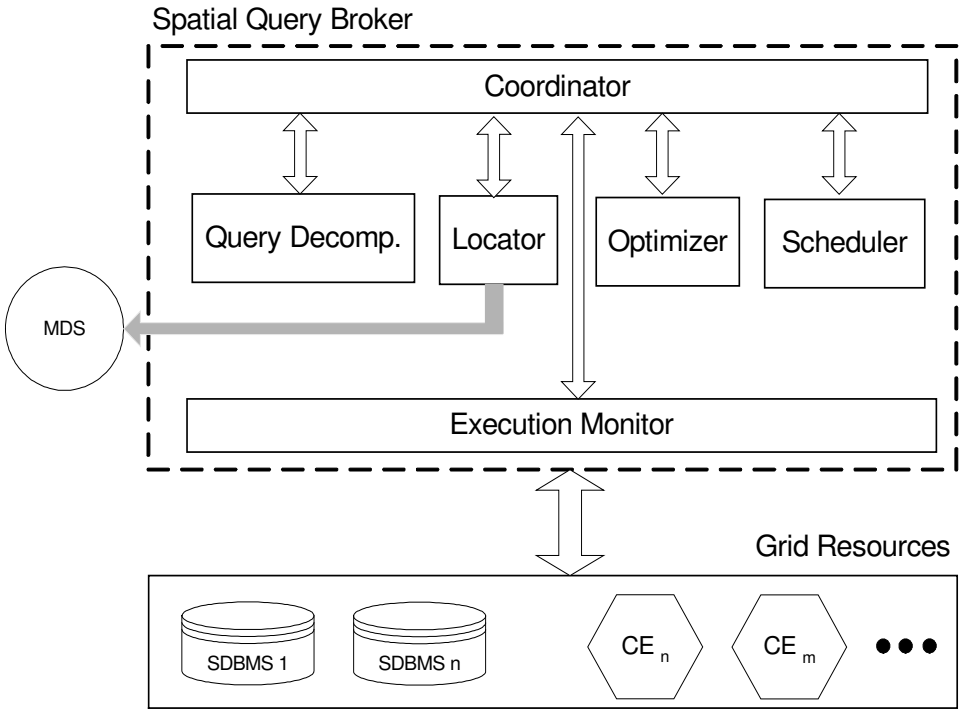

Figure 3. Spatial Query Broker architecture

The optimizer takes the simplified query received and, based on the fragments' locations and status of replica-storing computing elements, chooses the best set of servers to execute the filtering phase of a spatial join operation. This phase is part of the multi-step filtering proposed in (Brinkhoff et al. 1994) and its main purpose is to discard false hits based on a MBR join operation followed by a geometric filtering like 4CRS (Azevedo et al. 2004). The engine used to decide among the best servers makes use of the information collected from each of them in the *location module*. When dealing with either thematic or hybrid spatial data partitioning schema, the optimizer must request the transmission of the MBR approximations for the missing themes (and the 4CRS signature) to the specifics SDBMS, before proceeding with the filtering (figure 4). This procedure is well defined in (Ramirez 2001).

The *Execution monitor* is responsible for submitting, to a set of servers defined in the previous module, the query plan received, and monitor its execution. After finishing their approximate sub-queries (MBR filtering and geometric filtering), the servers return two distinct sets of data to the execution monitor. The identifiers of the pairs of objects that attend to the intersection's predicate build the first set, and the pairs of inconclusive identifiers compose the second set with their vertices' numbers. It should be noticed that after executing the geometric filter phase based, for example, on 4CRS (Zimbrão and Souza 1998), some true hits may be already detected.

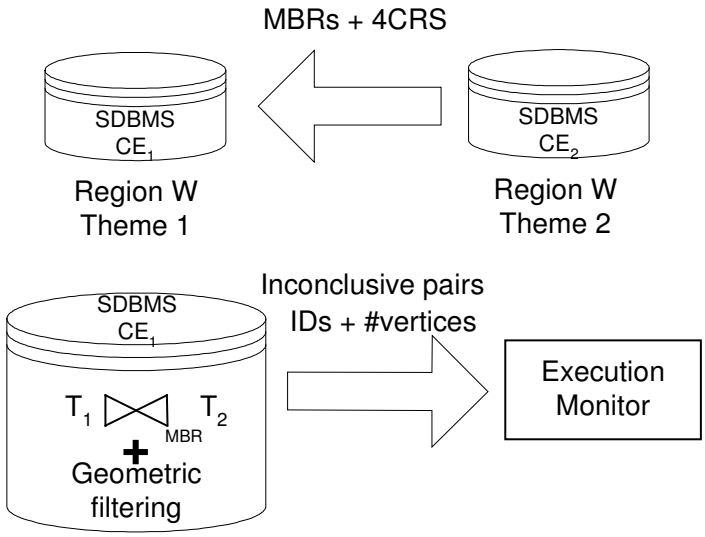

Figure 4. Sub-queries' running sequence in CEs with SDBMS

A *scheduler* module was included to assign exact geometries' tests to common computing elements. These CEs receive inconclusive pairs to process as a result of their status, capability (throughput) and communication channels characteristics. The scheduler can adopt user's directives, collected from a web portal or from a file, to change the criteria used to sort the CEs, in order to receive the inconclusive pairs. The scheduler builds two queues with the inconclusive pairs, one with the pairs that have a total number of vertices above a threshold limit (supplied by the user) and the other with the other pairs. To proceed with the scheduling, each CE receives a pair of geometries to process (without alphanumeric data) as a result of its processing power (throughput) and new pairs are distributed as fast as they complete processing. The powerful CEs are fed with the pairs from the queue that stores those with greater number of vertices, while the other CEs receive their pairs from the other queue (figure 5).

Finishing an exact test, a CE informs the execution monitor whether that specific pair satisfies or not the intersection predicate. At the end, when all pairs have been processed, the execution monitor eliminates all redundant information and the coordinator orders the SDBMSs to transfer the tuples that satisfy the predicate to the requestor machine.

The dynamic behavior of the scheduling process permits the addressing of more complex geometries to more powerful CEs. This granularity based on pairs may lead to a fine adjustment when coupling geometries complexity with processor nodes.

A simplified sequence diagram of the entire SQB functioning is presented in figure 6, which can be described as follows:

- A user interacts with the *coordinator*, by means of a Web portal, submitting a query that has its themes chosen from a list (according to a global schema). This query should be constrained by a defined region;

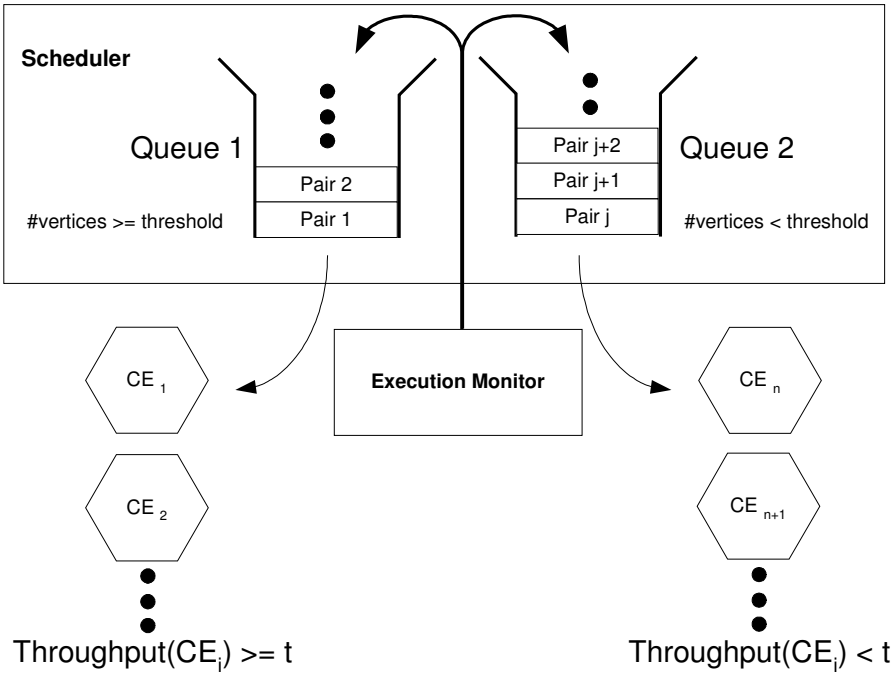

Figure 5. Queues in the scheduler

- The *coordinator* calls a *decomposer* instance to analyze and simplify the query based on a global scheme and idempotent rules;
- If no problems occur the *coordinator* asks the *locator* to request the information service (MDS) for a list with the SDBMS that can participate of the query, based on the region covered by each one and the themes stored within them. Replicas of the databases associated with these themes are also requested to the RLS. The status of available unspecialized CEs, periodically updated by the MDS, and the network bandwidth between all these nodes and the SQB are also acquired;
- After all this information has been supplied the *coordinator* chooses the best SDBMSs to participate in the query and, by means of the *execution module*, determines the execution of the two initial steps of the Brinkhoff proposal (Brinkhoff et al. 1994): the MBR join (filtering) and the geometric filtering. These two steps occur in a master CE, the one with the largest theme stored as suggested in (Ramirez 2001);
- After receiving the results, formed by a list of pairs that satisfy the predicate and a list with inconclusive pairs, the *coordinator* generates the queues to store the sorted inconclusive pairs (only their ids and location) in order to send them to chosen CEs and proceed with the exact geometry tests;
- The results returned to the *execution module* are then combined.

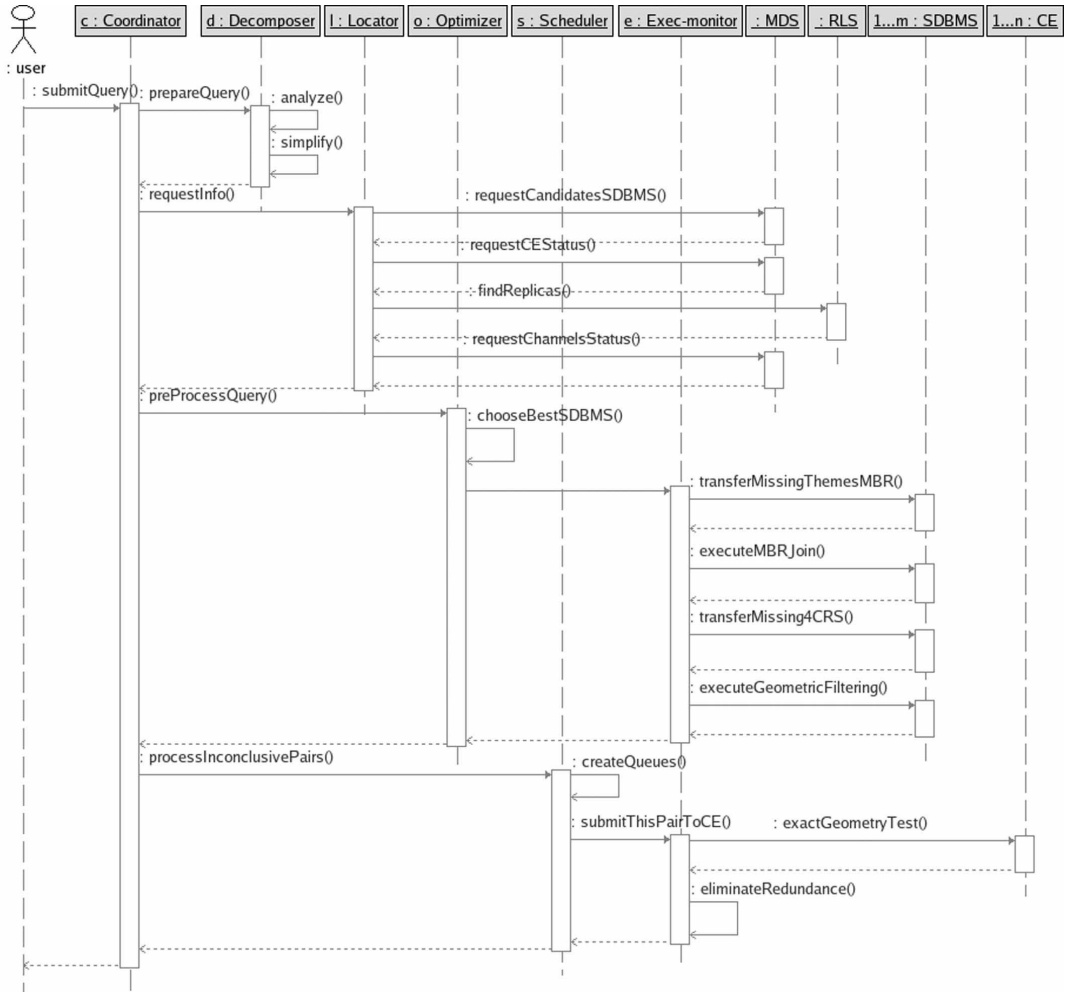

Figure 6. Simplified sequence diagram of the SQB

It is desirable that all interactions between broker and SDBMS should take place under an OGSA-DAI interface, an implementation of data access and integration interface proposed by the Global Grid Forum, which allows dealing with a possible heterogeneity among these SDBMSs.

Complementing the description of the SQB components, it is necessary to depict some other premises considered in this proposal in order to clarify the reasons for the strategies chosen. The next paragraphs cover these aspects.

Queries being analyzed are restricted to spatial joins involving themes with polygon geometries under a set of common spatial predicates, like those defined by Egenhofer (Egenhofer and Herring 1994): touch, overlap, inside and so on.

The family of spatial query joins covered by the present proposal can be seen in equation (*i*).

$$\text{Query} = \sigma_{w}(T_1) \underset{P}{\bowtie} \sigma_{w}(T_2) \qquad (i)$$

P = intersection

Where:

T_1 and T_2 are spatial themes,

W is a rectangular window and

P is the predicate used with the join operation.

There is also the premise that the VO offers some services needed by the SQB to interact with others VO's components. In the table 1 these services and their correspondent operations are listed.

Table 1: Other services needed in the VO

VO's Component	Service	Operations
Information Service	MDS	RequestGlobalSchema RequestCEStatus RequestCandidatesSDBMS RequestChannelsStatus RequestAGlobalID
CE (SDBMS)	OGSA-DAI WS-GRAM WSRF	Many Many ExactGeometryTest
CE	WS-GRAM WSRF	Many ExactGeometryTest

The *ExactGeometryTest* service, deployed in CE nodes, has the task of receiving a pair of geometries with their IDs and execute the exact geometry processing to verify if they satisfy a predicate (intersection in this case). Its result is true or false according to the processing.

The last premise is that there is a global identification structure where each feature stored in the SDBMSs has been already registered. This mechanism makes possible for the components to deal with any feature no matter where it is stored. This structure is similar to a catalogue service.

This global ID mechanism is used during all broker activities and all new created features should be registered.

Most of the work mentioned in the previous section cannot be considered query brokers in a distributed environment: some just execute queries, without planning; others emphasize data migration and replication when scheduling queries, and so on. In table 2, the architecture proposed in this section is compared to some known brokers with the purpose of consolidating some aspects.

Table 2: Brokers comparison

	SQB	OGSA-DQP	GridWay (Globus)	WMS (gLite)
semantic	query	query	job	job
Application domain	databases	databases	Generic job	Generic job
Job/query migration	no	no	yes	no
Dynamic scheduling	yes	no	yes	no
Support to spatial queries	yes	no	-	-
Nodes without database involved with query	yes	no	-	-
Follow GGF standards	yes	yes	yes	yes

7. Preliminary results

A prototype is being built to validate the proposed architecture and many of the functionalities described are under construction. The Globus Toolkit is being adopted as the grid middleware since its basic tools are already robust and used in several projects around the world. Its job manager is responsible for receiving a job description and taking the necessary steps for its execution. This component is named Globus Resource and Allocation Manager (GRAM) and, in release 4, has the ability to deal with 32,000 concurrent jobs against 300 in the previous release (Foster 2005).

The implementation is being made with a Web service approach by means of the Java WS-Core package provided by Globus. The SQB, accessed as a Web service in a specific server, makes use of stubs to interact with the others services like MDS, GRAM and the Replica Location Service (as depicted in figure 2) following a normal Web service style.

Secondo (Güting et al. 2004) was adopted as spatial database management system in this first stage. Its flexibility and modularity are achieved with an architecture based on algebras, which permit working with several data models. Spatial algebra is supplied with the product and changes or adjustments in its implementation can be easily done. The lack of an OGSA-DAI driver however, avoids its use in a heterogeneous group of servers. To test OGSA-DAI interface features, the PostgreSQL 8 / PostGIS will be adopted in a future stage.

Despite its being constructed, a few tests were done with synthetic spatial datasets consisting of polygons in order to give us some relative parameters to guide our work while dealing with spatial joins among polygons.

The tests were executed involving up to nine computers that run subqueries in parallel being the overall response time compared with that one obtained for a single machine running the entire query. With the times acquired, a speedup (Gistafson 1990) parameter was obtained for each configuration.

The conditions adopted on tests are presented below:

- The spatial database servers were used to store regions of a regular grid, each of them with only two themes;

- The themes had their geometric attributes represented by triangles, that could vary in shape and size;
- Two datasets with 10,060 synthetic objects, each one, were partitioned in four and nine regular areas;
- Communication costs were estimated, since all tests were executed in a local area network in spite of using a remote network such as the Internet;
- Only nodes with SDBMS were involved in the query.

The queries assumed during the tests had the form:

Select all pairs

from theme1, theme2

where theme1 overlaps theme2 and region = regionX

The response time used to build the table 3 follows the equation (*ii*).

$$RT = T_{MSG} * \#messages + T_{TX} * \#bytes + T_{CPU} + T_{I/O} \qquad (ii)$$

It was considered during the tests that the number of messages and the time spent with a single message transmission were constant, so the first term from equation (*ii*) was not considered. The communication cost term in equation (*ii*) depends on size of data and on communication's channel bandwidth that was estimated in 256 kbps.

The final response time is limited by the node that computes the worst response time, following equation (*iii*).

$$RT_{FINAL} = \max\{RT_1, RT_2, \ldots RT_i\} \quad i = \text{number of servers working in parallel} \qquad (iii)$$

The results observed for the spatial joins are presented in table 3, where costs are expressed in milliseconds and the resultset size in bytes.

Table 3: CPU, I/O and communication costs

# of SDBMS	Region Name	# obj	Resultset size	Costs			Total Cost	Speedup
				CPU	I/O	Comm		
1 SDBMS	FULL	10912	2538685	2213416	735	77475	2291626	1,00
4 SDBMS	NW	2385	533445	133680	172	16279	150131	11,94
	NE	2706	628885	147175	194	19192	166561	
	SW	3154	734406	169332	233	22412	**191977**	
	SE	2747	640873	148509	209	19558	168276	
9 SDBMS	1	1065	246074	24794	65	7510	32369	45,07
	2	922	212832	24820	64	6495	31379	
	3	1167	269244	29273	85	8217	37575	
	4	1140	266527	28396	81	8134	36611	
	5	1338	312329	34590	97	9532	44219	
	6	1252	292853	29057	90	8937	38084	
	7	1440	336147	35398	96	10258	45752	
	8	1547	360394	39730	113	10998	**50841**	
	9	1138	265508	27332	81	8103	35516	

Figure 7 sketches the improvement achieved when executing the join operation in a parallel manner.

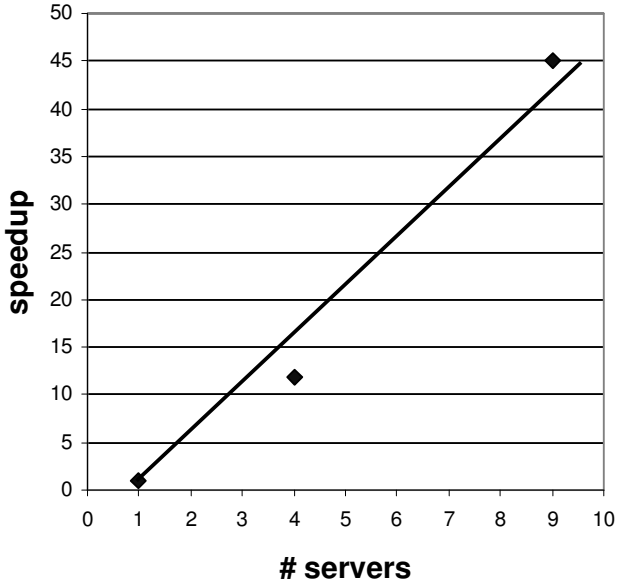

Figure 7. Speedup

The results observed give us a small, but important contribution in the sense that we can perceive some aspects of the spatial fragmentation adopted with the proposed architecture.

When member organizations work like regional data producers, i.e., generating all themes on a spatial region, a global query can take advantage of pre-existing spatial indexes already created on each SDBMS and the original query can be easily broken into sub-queries. The executed tests fall in this category and show that:

- Processing (CPU plus I/O) cost of the spatial join queries is normally greater than communication costs when dealing with large window areas;
- Some objects are processed more than once because they cross the boundaries as presented in table 2 (the number of objects, when summed, is greater than the whole dataset);
- When subqueries are executed in parallel the ratio: *available memory / query complexity* increase, leading to a superlinear speedup. Superlinear speedup, as seen in table 2, means that the resources used to process the whole query at once in a server were insufficient and the operation consumed too much time.

We can conclude that expensive operations, like spatial joins, normally spend more time processing than transmitting data and, for this family of operations, we can notice expressive improvement in their response time depending of the number of machines involved in their execution, since the original query has the possibility to use an extra amount of free resources not available in a single machine.

However, an extra communication overhead can be expected when an excessive number of servers is involved with an operation. In these situations communication

costs tend to be in the same order of magnitude or greater than processing costs, dominating overall response time. The effort made during the combination of partial results is another parameter that should be observed before a query subdivision, in order to avoid poorer performance.

Another point that should be emphasized in this structure is that the multi-processing of the same object must be avoided in order to improve the final response time. This problem causes great impact during the processing of exact geometries, since the algorithms' complexities, used to evaluate the queries' predicates, are directly dependent on their vertices' number. Besides that the amount of data to be transferred is increased, a post-processing stage to suppress redundant results being necessary.

8. Final remarks

The SQB architecture was conceived to explore the computational power of a virtual organization by means of dynamic scheduling in the most critical phase of a spatial join operation: the exact geometries processing. It is expected that queries over large spatial databases made up by several spatial database management systems, belonging to distinct organizations, can take advantage of this proposal, minimizing the effects of the natural grid's latency.

Government agencies responsible for the production of basic spatial data from large areas can benefit from it when offering query services over these huge amounts of data dynamically to other agencies, without the need of expensive clusters or equivalent equipment.

To summarize it, the adoption of a SQB in a grid environment avoids the need of a user knowing details of the locations of spatial data sources and gives him the possibility to run a query using the best resources available at that time. We suppose that the amount of computing elements running a query would have a strong impact on the final response time, as shown by the preliminaries tests. Another desirable feature of this architecture is that it explores the pre-existing spatial index of, at least, one of the two themes involved in the operation, reducing the time spent with the filtering and geometric filtering steps.

For the next stages we can highlight the prototype conclusion and the execution of several tests with actual and synthetic spatial datasets, with different parameter configurations such as vertex threshold, and with different cost models, the latter having a direct impact on the performance of the optimizer and of the scheduler.

Reference

Adzigogov, L., Soldatos, J., and Polymenakos, L. (2005). "EMPEROR: An OGSA Grid Meta-Scheduler based on Dynamic Resource." *Journal of Grid Computing*, 3, 19-37.

Afgan, E. (2004). "Role of the Resource Broker in the Grid." ACM, Huntsville, Alabama, USA.

Andretto, P. e. a. (2004). "Practical approaches to Grid workload and resource management in the EGEE project.".

Azevedo, L. G., Monteiro, R. S., Zimbrão, G., and Souza, J. M. (2004). "Approximate Spatial Query Processing Using Raster Signature.".

Brinkhoff, T., Kriegel, H., and Schneider, R. (1994). "Multi-Step Processing of Spatial Joins." Washington,DC - USA, 237-246.

Buyya, R., and Venegupal, S. (2004). "The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An overview and Status Report.".

Câmara, G., and Queiroz, G. (2002). "GeoBR: Intercâmbio Sintático e Semântico de Dados Espaciais.".

Di, L., Chen, A., Yang, W., and Zhao, P. (2003). "The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data.".

EGEE .(2006) "GLite - Installation and Configuration Guide v 3.0 (rev 2)" , European Union.

Egenhofer, M. J., and Herring, J. R. (1994) "Categorizing Binary Topological Relations Between Regions, Lines and Point in Geographical Databases" , NCGIA.

"Globus Toolkit 4."(2005). www.gridbus.org/escience/051205GlobusTutorialeScience.ppt, July/2006.

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

Foster, I., Kesselman, C., and Tuecke, S. (2001). "The Anatomy of the Grid Enabling Scalable Virtual Organizations." *Lecture Notes in Computer Science*, 2150.

Gistafson, J. L. (1990). "Fixed Time, Tiered Memory, and Superlinear Speedup.".

GridWay Team .(2006) "GridWay 5 Documentation: User Guide" Madrid, Spain, Universidad Complutense de Madrid.

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

Hanssen, G. (2005). "The Filter/Refine Strategy: A Study on the Land-Use Resource Dataset in Norway.".

Ilya, Z., Memon, A., Petropoulos, M., and Baru, C. (2003). "Online Querying of Heterogeneous Distributed Spatial Data on a Grid." Brno, Cz, 813-823.

Kang, M.-S., and Choy, Y.-C. (2002). "Deploying parallel spatial join algorithm for network environment." IEEE, 177-181.

Meyer, W. S., and Souza, J. M. (2006). "Overlapped Regions with Distributed Spatial Databases in a Grid Environment." Rio de Janeiro, Brazil.

Meyer, W. S., Souza, J. M., and Ramirez, M. R. (2005). "Secondo-grid:An Infrastructure to Study Spatial Databases in Computational Grids." Campos do Jordão, SP, Brazil.

Mondal, A., Goda, K., and Kitsuregawa, M. (2003). "Effective Load-Balancing via Migration and Replication in Spatial Grids." *Lecture Notes in Computer Science*, 2736, 202-211.

Özsu, M. T., and Valduriez, P. (2001). "Principles of Distributed Database Systems." Prentice-Hall.

Porto, F., Silva, V. F. V., Dutra, M. L., and Shulze, B. (2005). "An adaptive distributed query processing grid service." Trondheim, Norway.

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

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

"OGSA-DQP 3.1 User's Documentation."(2006). http://www.ogsadai.org.uk/documentation/ogsa-dqp_3.1/, July/2006.

Venegupal, S., Buyya, R., and Winton, L. (2004). "A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids.".

Zimbrão, G., and Souza, J. M. (1998). "A Raster Approximation for the Processing of Spatial Joins." New York - USA, 558-569.