Overlapped Regions with Distributed Spatial Databases in a Grid Environment

Wladimir S. Meyer[1], Jano Moreira de Souza[2]

[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] Computer Science Department, Institute of Mathematics
Federal University of Rio de Janeiro - Brazil
{wsmeyer,jano}@cos.ufrj.br

Abstract. Grid computing is becoming a natural tendency to be adopted in geographic information systems (GIS). Its characteristics, based strongly in resource sharing and standardization in critical areas like interfaces, security and data transfer, attract the attention of GIS research. On the other hand the design of distributed spatial Databases (DSDB) behind this information Systems contains many kinds of challenges inherited from both distributed database and spatial database areas. The performance of queries in a weakly-coupled environment, where data is spread, is one of the most important goals to be reached. Based on the Secondo-grid [1], a framework proposed to be used as test bed for distributed spatial databases, this paper introduces some modifications to allow the exploration of parallelism when executing queries over spatial databases servers in a computational grid. A federated architecture using spatial data partitioning with some level of overlapping, and the adoption of local index structures for spatial data, are behind the environment proposed. A query broker is suggested to generate and address the subqueries to correct servers based on data locations and servers status. Web services offered by the grid middleware are used to provide reliable file transfer, database register, replica location, and automatic monitoring of resources. Every time a region of interest is stored in more than one server, the query is split into the correspondent number of subqueries to be executed simultaneously and, after the processing is done the results are transferred and combined in the requestor machine.

1 INTRODUCTION

Distributed GIS is becoming a natural tendency as a decision support tool for many organizations from the most diverse areas of research around the world. Environmental pollution, deforestation monitoring and control, urban security and geological studies are some of them. The complexity of geographic data is frequently responsible for the specialization in its generation and the sharing of elements like huge amounts of data, specialized algorithms and computational resources has been attracting the interest of organizations as a whole. Governmental organizations, private organizations and research centers may act either as data sources or as data consumers and they may combine their resources to make up a distributed GIS that looks like a single entity from a client point-of-view. Issues related to integration of several spatial database management systems (SDBMS) belonging to different owners take a major grade of relevance in this scenario.

This work intends to cover the case of a distributed GIS developed in a top-down manner with an unique global schema being adopted by all member organizations. With this approach the rules to become a member of this virtual organization (VO) must be followed by all candidates. The solution proposed may be adopted, for example, by governmental organizations involved with the systematic mapping of defined regions, as in the case of Brazil. The top-down design means that there are no legacy systems incorporated in a new system. The purpose of this work is to present an architecture which allows reducing the time spent in queries that are executed over distributed SDBMS servers linked in a weak manner, like in the Web, based on parallelism and on the dynamic selection of remote SDBMSs that should respond to the query. The scalability of this solution is guaranteed by an adequate use of grid tools based on stateful Web services that are present in the Globus Toolkit release 4 [2].

The tight relationship between Distributed GIS and Distributed Spatial Database Management Systems (DSDBMS) has motivated much research focused in the latter one. DSDBMS connects problems from both domains: distributed systems and spatial databases management systems. Related to the former, some aspects like transmission costs, access control and replica locations may be highlighted. From the latter domain, aspects like long transactions, complex elements and spatial operators should receive special attention from researchers.

A data partitioning policy for local SDBMS should be chosen to be implemented. These policies may use a thematic approach; in this case each SDBMS stores specifics layers (themes related to a domain) or each of them stores all themes about a specific geographic region. In some case hybrid solutions can be adopted. In this work, the solution proposed involves a federated architecture and a data partitioning arrangement that follow a geographic region division.

Another important topic involving these kinds of database is concerned with spatial index structures that are used with operations that act over spatial data as spatial join and spatial select. R-trees and Quadtrees are the most commons structures used.

The infrastructure needed to support DSDBMS in a DGIS context should be capable of:

- offering a reliable manner to transfer huge amounts of data,
- authenticating users at once to allow access to all local SDBMS transparently,
- offering access only to authorized resources,
- hiding resource locations from final users,
- providing ways to guarantee data security.

Computational grids may attend adequately the needs above for a DSDBMS infrastructure. The virtual organization concept allows one to aggregate data sources and other resources from different organizations as if they were part of an unique organization despite of their locations. Another important feature of this environment is that an user can be authenticated in these VO only: there is no need to follow this step for each local SDBMS. The grid access control can supply an accurate granularity of access to address a specific resource. More recent grid tool kits, like Globus 4, offer a set of powerful tools, based on Web services, which permit reliable file transfer, registering, monitoring, discovery resources, and so on. These abilities qualify the computational grid as an adequate platform to be used as base infrastructure in a distributed GIS. Recently, many works adopted computational grids with this purpose [3-6].

In this work, the architecture proposed intends to explore some functionalities of a grid environment to permit building, in a top-down manner, a federated spatial database structure to be adopted in a GIS context. Like the work presented in [7] this architecture uses Web services and an index structure based on R-trees constructed in each node (database server). A global schema and a fragment map are published in a central server [1] to be accessed by all grid members. A macro region must be split into smaller regions based on a regular grid. As a suggestion, the grid defined by the millionth map, the International Map of the World , can be used.

2. RELATED WORK

In [3] an approach based on a virtual organization (VO) over a computational grid is suggested to solve several previously described problems as reliable data transmission and resource sharing, capable of integrating different organizations. However no considerations were made on the accesses in spatial databases, but only execution of geographic models in both parallel and single contexts. The architecture proposed was studied and a relation between the response time from both single processor and multi-processors was established. Using that relation, in theory, it is possible to determine the number of processors needed to have a specified response time. That solution is aimed at the use in multi-agency geospatial projects involving the collaboration of scientists, policy-makers, members of the public, and the approach involving a virtual organization was the most adequate.

The use of Web services as seen in [7] was a powerful solution the authors found to reach scalability in an infrastructure designed to optimize query processing in distributed spatial databases (DSDB). The scenario presented consists of independent organizations that produce data which may be overlap geographically. This data is not intentionally replicated over the members' 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 were 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 Gridbus project [8;9] is related to the construction of a high level broker that is capable of finding the best way to request data from a set of machines that belong to a computational grid. Factors like bandwidth, capability and performance of a resource are used to schedule the computers that will be responsible for the execution of a sub job. The Gridbus runs over several kinds of grid middleware like Globus, Nimrod-G and so on.

The current work intends to use some of the previous concepts to cover specifically distributed spatial databases spread in a computational grid. Share spatial data and break spatial queries, based on data locations and on resource status, are some of the desirable features of the proposed environment.

3. THE SOLUTION PROPOSED

This work covers a small segment of Distributed GIS that is involved with the production of geospatial data from well defined regions as it can be seen in several governmental agencies around the world. In Brazil, the Directory of Geographic Service (DSG) is one of the governmental organizations that systematically maps the country as a whole, its five sub organizations are responsible for mapping specific regions of the territory. In this specific case, an organization is responsible for the generation of several thematic data items. A huge amount of spatial data is generated by this kind of organization and the sharing of these data, for query purpose, is a considerable challenge.

The solution proposed consists of linking several SDBMS servers that produce and store spatial and non-spatial data from regions belonging to a regular grid. To make the approach easier, the regions' grid was based on the millionth map of the world [10] as shown in Fig.1.

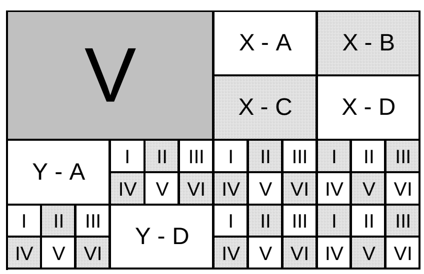

Fig. 1. The millionth map articulation and its subdivision until 1:100,000 scale

With this regular grid, a region equivalent to a 1:100,000 chart could be named as SE-23-X-C-I, for example, and the geographic limits of this region could be expressed as: long 45° W (left border), long 44.5° W (right border), lat 17° S (upper border) and 17.5° S (bottom border). Each cell corresponds to a region of 30'x 30', when talking about 1:100,000 regions.

Each server has the roles of data producer and also of replica updater. Complementarily, each of them stores the contents of fifty percent of cell located at North, South, East, West and twenty five percent of cells located at SW, SE, NW and NE (a cell is a region unit belonging to the regular grid) as in Fig. 2.

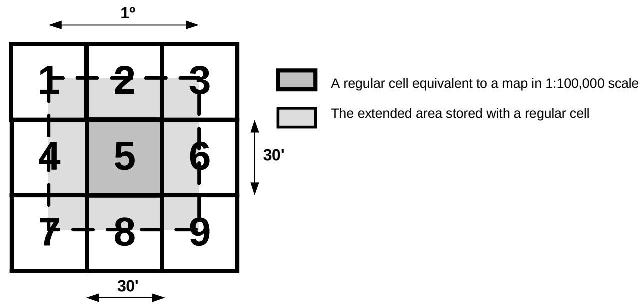

Fig. 2. Regions stored in a database server

This extra data however, has the sole purpose of reducing the time spent with queries involving adjacent regions, since the overlapped area can be subdivided during query processing to allow a parallel approach. Only the producer of the data should have the authority to update it when necessary. This policy guarantees data integrity since each region has an unique owner enabled to write, update or delete objects from it. If the query's region is totally included in the area covered by a local SDBMS, its original region plus the extended area, the user may choose to use only that machine to process the entire query, avoiding costs with data transmission but increasing the cost of processing.

The information on which server stores a specific region is published in a central MDS and can be delivered to any server in the virtual organization.

This information will be used by the broker module when determining the servers that may receive the sub queries.

When a new SDBMS intends to join the structure it should inform the central MDS about the region stored that it will share with the others servers. It should also request the extended region adjacent to it from the other servers.

Spatial indexes are generated locally on each node and must be updated when some change in the spatial data produces a modification in its minimum bounding rectangle (MBR) that is the basis of the R-tree mechanism.

The extended region consists of twelve pieces that correspond to quadrants (NW, NE, SE, and SW) from the adjacent regular cells.

A computational grid infrastructure is built to reach those desirable features listed in section 1. The grid middleware adopted was Globus because of its powerful set of tools already employed in several important research projects like GridBus [8], GriPhyN [11], High Energy Physics at CERN[12] and so on.

A prototype is being built using the Secondo-grid framework presented in [1]. Some modifications were made, specifically on the partitioning of spatial data that was changed from a thematic to a regional kind, the latter employing a level of overlapping as explained before.

Each server stores several kinds of themes like hydrography, buildings, roads, vegetation and so on. It was also necessary to change the routine used to split the global query into sub queries (the broker) to support parallel processing by the servers. Since the Secondo grid is a framework for studying spatial databases in a grid, others query brokers may be implemented for the purpose of performance testing, allowing comparisons between them. The broker used in this prototype is shown in Fig. 3.

All of this structure was inherited from the original framework but the Query Plan Maker Module was modified to split the queries depending of the region being queried and to understand the new format of the fragments map, based on region and not on themes.

A global schema and a fragments map are published in a central server that is running a monitoring and discovering service (MDS), as in the original framework, to be accessed by all grid members. The Global Schema is a GML document that can be accessed with xPath sentences.

Each local SDBMS server has a Java container with a set of grid services running as proposed in [1]: MDS, RFT, GRAM and RLS.

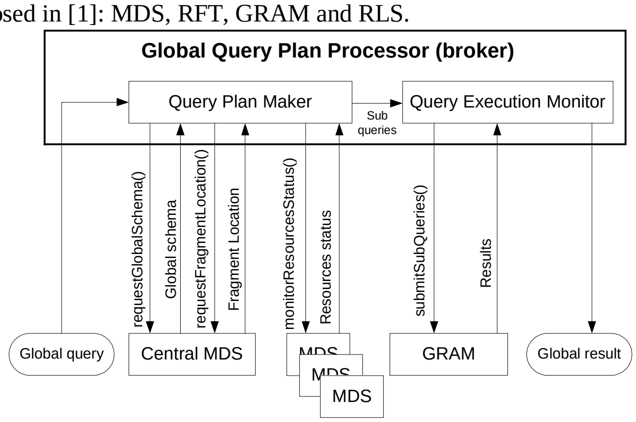

Fig. 3. Global query plan processor (broker)

3.1. SYSTEM OVERVIEW

To reinforce some characteristics of the architecture proposed, some important features are summarized below:

- The overlapped regions may be used by the owner of a regional server to link adequately the trespassing objects, avoiding discontinuities in the region's border;
- Replicas are automatically updated by means of the Replica Location Service, the Monitoring and Discovery Service and the Reliable File Transfer Service;
- Parallelism is used to improve queries based on architecture;
- A broker is responsible for query and region subdivision after a global query is supplied;
- A new SDBMS can be more easily added to an existing system, since all of them have the same functionalities and must know only the IP address of the central MDS server to be capable of acquiring the global schema and the fragments map. Moreover, this SDBMS should register its stored region in the central MDS;
- On each server the 300% increase provided by the extra area makes it possible to run queries near the region's border in a unique site when

necessary. This can be the case when an adjacent server is out of order or powered off;

– The index structure is local, thus avoiding unnecessary traffic increments;
– There is not a global mediator and each server has the necessary abilities to control locally all steps needed to proceed with a query execution, a replica update and so on. The only central resource is the central MDS service, responsible for supplying the nodes either with a global schema or with a fragments map;

If there is more than one database server storing the same region from the regular cell, the broker of the requestor machine is capable of deciding about the best one to address the sub-query, based on information collected from their MDS services. A small program was developed to collect some necessary information such as CPU load, total amount of memory, total amount of free memory, number of running processes, number of active processes, number of users logged in, and the total amount of free space in the hard disk. The result is that when the MDS is asked about this information the program is called and, having collected the necessary data a XML document is generated.

3.2. MODULES DESCRIPTION

To describe the environment, the explanation will be subdivided in two modules: a query processor and a replica management module.

The first one starts when a user fills the query form which is embedded in the Secondo's Graphic User Interface and a routine checks its syntax to avoid typing mistakes.

At this moment, if there is not a global schema replica stored locally, it is requested to the central MDS. The result, returned in XML format, is loaded on memory to increase future queries. This schema is helpful to a semantic analysis of non-topological associations. The fragments map is also read and stored locally to allow the broker to find the servers that store regions that intersect the query's region. Since the region covered by the query is directly acquired from the onset, the indexes of regions, from millionth map, which intersect it, are discovered before and the fragments map is scanned to discover the servers that store them.

Afterwards, the query's region is split into smaller regions, to be addressed to specifics servers. If there is more than one server storing a region the broker should request their status through the local MDS. With this information, the server that provides the best conditions should be chosen. These conditions must be pre-defined by the local administrator.

After all these preliminary steps, the servers, the sub-queries, and the regions that they must process are defined and job description files should be generated to be sent to each of them. These files are submitted to their respective GRAM servers and the same number of end-point-reference (EPR) is created locally to handle the job status. The job description files must inform the remote machines, the results generated being to the requestor.

The final result is obtained from the partial results combination and, if errors are detected, they should be informed in the GUI. The integrated viewer in Secondo GUI could be used to show the result.

All the interactions with remote machines are made by means of Java client stubs that provide an adequate manner to activate the grid Web services: MDS, GRAM and so on.

The module responsible for replica management has as main function the updating of replicas when the original one is changed. After data about a region is updated by its owner, the modified quadrants must be saved in specific files; for example, if the southwest quadrant from SE-23-X-C-I was modified, the owner would have to save the new southwest quadrant in the file SW-SE-23-X-C-I. On each server, the quadrants' files are registered with the MDS and RLS. MDS registration triggers an action every time the file is changed and RLS registration permits to determine where each replica is stored. The action associated with the MDS trigger must generate as many jobs as replicas to be submitted to the remotes servers. The job description files make use of the RFT service for all transfer tasks. The four quadrants from a region, SE, SW, NE, and NW have specific files associated to them in the format: SE-XX-XX-X-X-X, SW- XX-XX-X-X-X, NE- XX-XX-X-X-X, NW-XX-XX-X-X-X, where XX-XX-X-X-X is the millionth index.

It is important to emphasize that only the themes changed by the owner should be transferred to remote machines. When replicas are updated, the administrators for the machines involved should load and re-index them to reflect the changes made.

4. SOME PRELIMINARY RESULTS

The prototype of the current architecture is under construction and not all functionalities are activated, but some tests were made, in the attempt to validate it. Two kinds of tests were evaluated to provide information on response time to queries.

The first one is based on a simple query for a specific theme which is into a region corresponding to a 1:100.000 scale map, this region corresponding to a cell of the millionth map as seen before. The queries in this case can be seen as spatial selects and themes with the three standard geometry types were used (point, line and region). All themes were pre indexed with R-trees.

The second kind of tests involves spatial joins over two themes. Some theme combinations with different geometries were made: line x line (LL), line x point (LP) and point x point (PP).

All tests were executed in three situations: the whole query at once in a single machine that stores a millionth map (referred as "1 slice" in tables below), the query over the same region split into four slices (NW, NE, SE, and SW) distributed among four identical machines (referred as "4 slices" in the tables below) and the query over the same region split into nine slices, distributed among nine identical machines (referred as "9 slices" in the tables below). In a real situation, these nine machines are: the server owner of the map and one's eight neighbours (servers that store the adjacent regions).

Tests involving cost for communications and processing of trespassing objects have not been executed yet. However, some simulations were made based on a network with pass-width of 256kbps and the files generated by the tests described above.

4.1. THE RESULTS

The region used for all tests is the SE-22-X-B-III and the queries used in the tests are presented in Table 1. Results acquired from the tests are presented in Tables 2 and 3, respectively. The notation “NW (SE-XX-X-X-X)” means the northwest from region SE-XX-X-X-X and, in a similar manner, the notation “1 (SE-XX-X-X-X)” means the slice number one from region SE-XX-X-X-X. The tables show the CPU response time (RT) spent to perform each result and the time needed to write the results to a file (I/O). All times presented are in “ms”, and RT_i means the response time spent executing the query in the “i-nth” server, while the RT is the worst case (just one slice).

Table 1. Queries employed with tests

Test	Query
I	Select all drain_lines from SE-22-X-B-III Select all edifications from SE-22-X-B-III Select all vegetations from SE-22-X-B-III
II	Select tracks that are crossed by drain_lines (LL) Select edifications with distance < 100m from tracks (LP) Select edifications with distance < 1000m from schools (PP)

Table 2. CPU time spent with queries from the first kind

	Drain Line (line)					Edification (point)					Vegetation (region)				
	CPU	I/O	RT	N° obj	Size(B)	CPU	I/O	RT	N° obj	Size(B)	CPU	I/O	RT	N° obj	Size(B)
1 SLICE															
SE-22-X-B-III	59942	1454	61396	**2445**		2355	91	2446	**1382**		59780	3019	62799	**268**	
4 SLICES															
NW(SE-22-X-B-III)	14008	407	**14415**	708		767	30	**797**	456	**48492**	24032	1095	25127	80	
NE(SE-22-X-B-III)	13804	428	14232	759	**605335**	641	25	666	373		22765	1053	23818	44	
SE(SE-22-X-B-III)	10987	354	11341	558		612	22	634	324		27242	1217	**28459**	96	**1910523**
SW(SE-22-X-B-III)	9693	322	10015	480		425	16	441	229		20666	938	21604	74	
Total				**2505**					**1382**					**294**	
RT i MAX / RT			**0,235**					**0,326**					**0,453**		
9 SLICES															
1 (SE-22-X-B-III)	5525	181	5706	313		503	19	**522**	262	**27920**	7707	357	8064	24	
2 (SE-22-X-B-III)	5756	193	5949	394	**296051**	313	12	325	151		13024	638	13662	34	
3 (SE-22-X-B-III)	5038	173	5211	298		415	38	453	222		9574	441	10015	19	
4 (SE-22-X-B-III)	4868	166	5034	236		318	11	329	157		12969	589	13558	69	
5 (SE-22-X-B-III)	5916	200	**6116**	347		186	5	191	61		17982	896	**18878**	48	
6 (SE-22-X-B-III)	5600	199	5799	325		265	10	275	126		12595	670	13265	24	
7 (SE-22-X-B-III)	3971	149	4120	230		259	9	268	124		8225	378	8603	21	
8 (SE-22-X-B-III)	3651	135	3786	196		200	6	206	82		11817	510	12327	20	
9 (SE-22-X-B-III)	4545	161	4706	238		392	14	406	197		13788	607	14395	72	**961953**
Total				**2577**					**1382**					**331**	
RT i MAX / RT			**0,100**					**0,213**					**0,301**		

Table 3. CPU time spent with queries from the second kind

<table>
<tr><th></th><th colspan="5">LINE x LINE</th><th colspan="5">LINE x POINT</th><th colspan="5">POINT x POINT</th></tr>
<tr><th></th><th>CPU</th><th>I/O</th><th>RT</th><th>N° obj</th><th>Size(B)</th><th>CPU</th><th>I/O</th><th>RT</th><th>N° obj</th><th>Size(B)</th><th>CPU</th><th>I/O</th><th>RT</th><th>N° obj</th><th>Size(B)</th></tr>
<tr><td colspan="16">1 SLICE</td></tr>
<tr><td>SE-22-X-B-III</td><td>218598</td><td>419</td><td>219017</td><td>506</td><td></td><td>113914</td><td>116</td><td>114030</td><td>495</td><td></td><td>1979</td><td>9</td><td>1988</td><td>67</td><td></td></tr>
<tr><td colspan="16">4 SLICES</td></tr>
<tr><td>NW(SE-22-X-B-III)</td><td>20032</td><td>251</td><td>20283</td><td>125</td><td></td><td>10428</td><td>20</td><td>10448</td><td>157</td><td>6992</td><td>693</td><td>7</td><td>700</td><td>18</td><td></td></tr>
<tr><td>NE(SE-22-X-B-III)</td><td>31934</td><td>147</td><td>32081</td><td>173</td><td>178642</td><td>11639</td><td>15</td><td>11654</td><td>131</td><td></td><td>388</td><td>2</td><td>390</td><td>10</td><td></td></tr>
<tr><td>SE(SE-22-X-B-III)</td><td>17049</td><td>137</td><td>17186</td><td>118</td><td></td><td>7924</td><td>9</td><td>7933</td><td>127</td><td></td><td>419</td><td>6</td><td>425</td><td>25</td><td></td></tr>
<tr><td>SW(SE-22-X-B-III)</td><td>12751</td><td>92</td><td>12843</td><td>97</td><td></td><td>4384</td><td>8</td><td>4392</td><td>80</td><td></td><td>345</td><td>5</td><td>350</td><td>14</td><td>698</td></tr>
<tr><td>Total</td><td></td><td></td><td></td><td>513</td><td></td><td></td><td></td><td></td><td>495</td><td></td><td></td><td></td><td></td><td>67</td><td></td></tr>
<tr><td>RT i MAX / RT</td><td></td><td></td><td>0,146</td><td></td><td></td><td></td><td></td><td>0,102</td><td></td><td></td><td></td><td></td><td>0,352</td><td></td><td></td></tr>
<tr><td colspan="16">9 SLICES</td></tr>
<tr><td>1 (SE-22-X-B-III)</td><td>7870</td><td>55</td><td>7925</td><td>73</td><td></td><td>3689</td><td>6</td><td>3695</td><td>83</td><td>3739</td><td>366</td><td>2</td><td>368</td><td>18</td><td>867</td></tr>
<tr><td>2 (SE-22-X-B-III)</td><td>6288</td><td>44</td><td>6332</td><td>58</td><td></td><td>2086</td><td>6</td><td>2092</td><td>43</td><td></td><td>314</td><td>2</td><td>316</td><td>3</td><td></td></tr>
<tr><td>3 (SE-22-X-B-III)</td><td>10337</td><td>86</td><td>10423</td><td>99</td><td>113289</td><td>4335</td><td>5</td><td>4340</td><td>83</td><td></td><td>333</td><td>1</td><td>334</td><td>5</td><td></td></tr>
<tr><td>4 (SE-22-X-B-III)</td><td>4623</td><td>38</td><td>4661</td><td>42</td><td></td><td>2066</td><td>8</td><td>2074</td><td>71</td><td></td><td>325</td><td>1</td><td>326</td><td>1</td><td></td></tr>
<tr><td>5 (SE-22-X-B-III)</td><td>4350</td><td>32</td><td>4382</td><td>42</td><td></td><td>1154</td><td>6</td><td>1160</td><td>33</td><td></td><td>292</td><td>2</td><td>294</td><td>1</td><td></td></tr>
<tr><td>6 (SE-22-X-B-III)</td><td>6262</td><td>35</td><td>6297</td><td>52</td><td></td><td>2154</td><td>7</td><td>2161</td><td>50</td><td></td><td>293</td><td>2</td><td>295</td><td>2</td><td></td></tr>
<tr><td>7 (SE-22-X-B-III)</td><td>4391</td><td>57</td><td>4448</td><td>55</td><td></td><td>1397</td><td>6</td><td>1403</td><td>26</td><td></td><td>303</td><td>6</td><td>309</td><td>11</td><td></td></tr>
<tr><td>8 (SE-22-X-B-III)</td><td>3474</td><td>30</td><td>3504</td><td>32</td><td></td><td>1066</td><td>4</td><td>1070</td><td>24</td><td></td><td>300</td><td>7</td><td>307</td><td>13</td><td></td></tr>
<tr><td>9 (SE-22-X-B-III)</td><td>6738</td><td>69</td><td>6807</td><td>73</td><td></td><td>3299</td><td>9</td><td>3308</td><td>80</td><td></td><td>336</td><td>5</td><td>341</td><td>13</td><td></td></tr>
<tr><td>Total</td><td></td><td></td><td></td><td>526</td><td></td><td></td><td></td><td></td><td>493</td><td></td><td></td><td></td><td></td><td>67</td><td></td></tr>
<tr><td>RT i MAX / RT</td><td></td><td></td><td>0,048</td><td></td><td></td><td></td><td></td><td>0,038</td><td></td><td></td><td></td><td></td><td>0,185</td><td></td><td></td></tr>
</table>

4.2 REMARKS ABOUT THE RESULTS

In these preliminary tests, only two small shares from the whole response time expression (RT) [13] were observed: processing time (T_{CPU}) and input/output time ($T_{I/O}$).

$$RT = T_{MSG} * \#messages + T_{TX} * \#bytes + T_{CPU} + T_{I/O} \quad (1)$$

$$RT_{FINAL} = \max\{RT_1, RT_2, \ldots RT_i\} \quad i = \text{number of servers working in parallel} \quad (2)$$

Final response time, when queries are running in parallel, is the maximum from all of the response times computed for each SDBMS server.

The first two terms from equation (1) are expected to be dominant in a real context and involve the time spent with message exchange and time spent with data transfer after queries execution. In a hypothetical situation with a network band width of 256 kbps, we could have the new overall response time for each kind of test (Table 4).

We see by the results found that the time spent with CPU and I/O processing in all tests was very good. The relative response time for these two terms fell to values between 23,5% and 45,3% from the original response time when executing tests from the first kind with four slices, while the results were better when the original query was split into nine subqueries over the nine servers: 10% to 30.1%.

When running tests involving join operations, which are much more complex than the first kind, the improvement seen remained very good, with response times ranging from 3.8% to 18.5% from the original.

Despite the good results we observed that in some cases the number of objects returned by queries was a little different when comparing the same test with one, four, and nine slices. This fact is related to the existence of trespassing objects that, in some circumstances, are processed more than once. It is expected that at the end, after combining query results, these duplicated objects would be eliminated from the final result.

Table 4. Total Response Time with a hypothetical band-width of 256 kbps

	Drain Line			Edification			Vegetation		
	RT_{MAX}	RT_{COM}	RT_{TOT}	RT_{MAX}	RT_{COM}	RT_{TOT}	RT_{MAX}	RT_{COM}	RT_{TOT}
1 slice	-	-	61396	-	-	2446	-	-	62799
4 slices	14415	18473	32888	797	1480	2277	28459	58305	86764
9 slices	6116	9035	15151	522	852	1374	18878	29356	48234
Query Time (4 slices) %			**53,57**			**93,08**			**138,16**
Query Time (9 slices) %			**24,68**			**56,18**			**76,81**
	Line x Line			**Line x Point**			**Point x Point**		
	RT_{MAX}	RT_{COM}	RT_{TOT}	RT_{MAX}	RT_{COM}	RT_{TOT}	RT_{MAX}	RT_{COM}	RT_{TOT}
1 slice	-	-	219017	-	-	114030	-	-	1988
4 slices	32081	5452	37533	11654	213	11867	700	21	721
9 slices	10423	3457	13880	4340	114	4454	368	26	394
Query Time (4 slices) %			**17,14**			**10,41**			**36,28**
Query Time (9 slices) %			**6,34**			**3,91**			**19,84**

The sizes of files generated during the tests directly affect communication costs. The larger file generated, when using four regions, had 1.8MB whereas the largest one, when working with nine slices had 939KB. It is clear that, in some scenarios, the cost to transfer these files could be very high.

The simulations also give us a good scenario. When working with nine servers the sizes of files are smaller, reducing communication costs and, as result, we have the relative response time ranging from 3.91% to 19.84%.

5. FINAL REMARKS

To improve response times related to a distributed query, the final cost of the operation should be reduced. This cost is made of processing and communication costs, the latter one usually having greater impact on the final response time. Despite the reduction of the number of messages exchanged between servers, it is important to emphasize the need for an adequate operating system and network tuning when using high performance network paths [14]. The number of buffers and their sizes are some items that should be checked.

Load balance is another big challenge to be highlighted since geographic slices produced by the broker do not consider the nature of data inside them. In some circumstances servers with low processing power receive areas more complex than others with high processing power increasing response time. So other algorithms should be tried in the broker module to supply better results when the nature of data is not uniform between regions.

The flexibility offered by Secondo, as a SDBMS, allows the use of specific data models, like temporal, climate, etc., and can open a new range of possibilities, enabling, for example, research with mobile objects in this DSDBMS environment.

On the other hand the power of the GT4 middleware, increased by the use of stateful Web services, can allow an easier sharing of new services tailored for this distributed environment.

As major challenge we can emphasize the need of research involving more sophisticated brokers capable of optimizing the global query under many circumstances like cost (time, resource value), resource performance, load balance, trespassing objects, and resource locations.

The proposed architecture showed good results related to response time involving CPU and I/O processing, but only complementary tests capable of covering communication costs could give us the knowledge necessary to establish the context where it can be better employed.

6. REFERENCES

[1] Meyer, Wladimir Silva, Souza, Jano Moreira, and Ramirez, Milton Ramos. Secondo-grid:An Infrastructure to Study Spatial Databases in Computational Grids. VII Brazilian Symposium on Geoinformatics. 2005. Campos do Jordão-SP, VII Brazilian Symposium on Geoinformatics.

[2] Foster, Ian. A Globus Toolkit Primer Describing Globus Toolkit Version 4 - Draft. 24-2-2005.

[3] Shi, Yi, Shortridge, Ashton, and Bartholic, Jon. Grid Computing for Real Time Distributed Collaborative Geoprocessing. 2002.

[4] Ilya, Zaslavsky, Memon, Ashraf, Petropoulos, Michalis, and Baru, Chaitan. Online Querying of Heterogeneous Distributed Spatial Data on a Grid. 2004.

[5] Wang, Yufei, Ge, Linlin, Rizos, Chris, and Babu, Ravindra. Spatial Data Sharing on Grid. 2005.

[6] Zhao, P., Chen, A., Liu, Y., Di, L., Yang, W., and Li, P. Grid Metadata Catalog Service-Based OGC Web Registry Service. 2004.

[7] Roger Zimmermann, Wei-Shinn Ku, and Wei-Cheng Chu. Efficient Query Routing in Distributed Spatial Databases. 2004.

[8] Venegupal, Srikumar, Buyya, Rajkumar, and Winton, Lyle. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. 2004.

[9] Buyya, Rajkumar and Venegupal, Srikumar. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An overview and Status Report. 2004.

[10] United Nations. Annual Report on The International Map of the World of Millionth Scale. 1951. NY, Social and Economic Affairs Division.

[11] Zhao, P., Wilde, M., Foster, Ian, Voeckler, J., Jordan, Thomas, Quigg, Elizabeth, and Dobson, J. Grid Middleware Services for Virtual Data Discovery, Composition, and Integration. 2004.

[12] Stockinger, Heinz. Distributed Database Management Systems and the Data Grid. 18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies. 2001. San Diego, CERN (European Organization for Nuclear Research).

[13] Özsu, M. T. and Valduriez, P. Principles of Distributed Database Systems, Prentice-Hall, 1999.

[14] Mathis, Matt and Reddy, Raghu. Enabling High Performance Data Transfers [PSC]. Advanced Networking - Pittsburgh Supercomputing Center . 2006