Distributed Processing of Spatial Join

MILTON RAMOS RAMIREZ
JANO MOREIRA DE SOUZA
COPPE/Sistemas-UFRJ, Centro de Tecnologia, H-319, CxP. 68.511. CEP 21.945-970
Instituto de Matemática-UFRJ, Centro de Tecnologia, C-114, CxP: 68.530. CEP 21.945-970
Cidade Universitária, Rio de Janeiro, RJ, Brasil.
milton@cos.ufrj.brj
jano@cos.ufrj.br

Abstract. This paper describes a new strategy for distributed processing of intersect spatial joins which are appropriate for the development of DSDBMS – Distributed Spatial Database Management Systems. We describe the experiments we carried out to compare the performances of our strategy to others found in the literature. We have concluded that ours is suitable to DSDBMS and also more efficient than the other strategies.

1 Introduction

Nowadays Geographic Information Systems – GIS – that use the technology of distributed systems on computer networks are very common [12][7]. This has represented a current challenge for the several disciplines involved with GIS development.

The GIS based on distributed systems – or Distributed GIS (DGIS) – can be defined as those that have a varied number of autonomous geographic systems, not necessarily homogeneous, interconnected by a computer network and which cooperate in the accomplishment of *geographic data processing* with the data stored in their local spatial databases.

The database management component of a DGIS must be capable of managing a collection of multiple spatial databases logically interrelated, which are scattered over a computer network and must also be able to offer, at least, the transparency of the distribution. Such a category of Spatial Database Systems – SDBS – was denominated *Distributed Spatial Database Systems* – DSDBS [1].

The development of a DSDBS brings several interesting challenges. Among them we consider the *efficient distributed query processing* as the main one. This because it is such a central issue for all categories of DBS, as well as, the efficiency of a GIS is dependent of the effectiveness of the query processing of its database system component [6].

The debate of query processing in DSDBS involves many aspects. The type of fragmentation and distribution of spatial database over the network, the resolution of heterogeneities, the autonomy of local systems that compose the DSDBS, the distribution architecture of the controls and the mechanisms of interoperation of data, metadata and resources [12][15].

In this scope, we can identify specific issues in distributed query processing, such as the data location, global query fragmentation according to spatial rules, generation and optimization of the execution query plan and issues related to the strategies for efficient distributed processing of spatial operations such as the spatial join [12]. The investigation that we developed focused on this last issue.

Thus, the objective of this paper is to present our latest results concerning the proposition of a new strategy for distributed spatial join processing with an intersection predicate between two spatial data sets which are distributed between two distinct components of a DSDBS.

Our strategy was developed based on two directives, which were obtained from our investigations on this type of distributed query processing. In summary, the directives determine that the strategies for this type of query processing must (1) take the maximum advantage of the parallelism provided by the distribution and also (2) use the interoperation mechanisms to optimize the *local processing* portion of distributed query processing and also to aid reaching parallelism [12]. These directives made our strategy more efficient than others found, until then, in the literature. Such a fact was proved through experimental results.

This work is divided into more six sections. In the next one, we present, in general, the main issues involved with distributed query processing in DSDBS. We center our argument on the distributed spatial operation processing, for which we present our directives. In section 3 we discuss the strategies that we found in the literature and we point out its limitations in relation to our directives. In section 4 we present our strategy proposal. In the last three sections we present our experiments, results and conclusions on the performance comparisons between the strategies.

2 Distributed Processing of Spatial Operations

In Distributed Spatial Database Systems the sequence of steps of the query processing does not differ from any Distributed Database System – DDBS. However, despite the objectives of each step remaining the same ones, in this case they involve the new complexities identified for any centralized Spatial DBS. Therefore those steps of query processing had become *non-conventional* [13]. Thus, the solutions established for the Distributed Relational DBS to processing queries in DSDBS cannot be directly adopted, due to the fact that the query processing can require long and recursive transactions involving spatial data that can be complex and large.

In summary, the steps of the query processing in DSDBS have issues that are a union between those raised from the *distributed* query processing in conventional Distributed DBS with those identified for the *spatial* query processing in Spatial DBS. Most of them are not yet enough studied for all possible scenes of distribution [15].

We find in the literature some articles on query processing in DSDBS. In [15] we find a prototype of a DSDBS for Federated GIS that is implemented using CORBA. It proposes a solution for entire query processing and it also presents particular contributions to the solution for the generation of optimized plans for the execution of the distributed spatial query. In KUMAR et al. [7] we can find a prototype of a distributed object-oriented GIS that contributes with a solution for the query processing based on the use of a global module that coordinate several other modules that manage the component databases.

In a deep probe, we found more specific work on distributed query processing. They are about strategies for the distributed processing of spatial operations like the spatial join, which represents the kernel of almost all distributed spatial queries. We found in [1], [2], [12], [13] and [14] distributed strategies for the spatial join operation with the intersection predicate. AREF [2] proposes also a distributed strategy for the selection of pairs of nearest neighbor objects. The work by TAN et al. [14] is an extension, by the same authors, of the work presented in ABEL et al. [1]. Both carry through experiments with their proposals using a prototype of Federate DSBDS. We will comment these strategies particularly in the following section.

RAMIREZ [12] discusses all involved aspects with query processing in DSDBS. He discusses from the general issues related with the development of the DSDBS to the specific issues of the distributed processing of spatial operations, including the discussion about the spatial fragmentation of the database, the type of architecture of the DSDBS according to the degree of autonomy and problems with spatial objects that cross over borders of the subdivision of space generated by the spatial fragmentation.

Not regarding all the context of complications involved with this subject, our interest in this work is centered in the issue related to the strategies used for the distributed processing of the spatial join operation. RAMIREZ [12] points out that there are several alternatives of strategies for this type of spatial operation, since the strategies can vary in accordance to the predicate of the spatial join, types of data set fragmentation and distribution, autonomy degree of local Spatial DBS components and existence of local spatial indices related to the local spatial data sets.

In general, the strategies proposed in the literature are based on the idea of using spatial approximations of objects to minimize the transmission cost due to the reduction of the data sets to be transmitted to the final join. However [1], [2] and [14] do not take advantage of parallelism provided by the distribution in the final step of the processing of the join operation, neither optimize the step of the local geometric processing, since they do not make use, without additional cost of the known method for efficient spatial join, nominally, the filter and refinement process.

Based on the observation of these deficiencies, RAMIREZ et al. [13] stated that this was a good track to be followed on the proposition of strategies for the spatial join processing to adopt the process introduced by BRINKHOFF et al. [5], this embraces the use of *local* spatial indices and the accurate spatial approximations of spatial objects selected to join for the filtering step. This would allow, with an optimized cost, the prune of spatial objects that do not belong to the final result of the query, thus allowing the cost reduction of the local processing and minimizing the data transfer in the communication network.

In this way, RAMIREZ [12] proposed and concluded that to produce strategies that would be more efficient for query processing in the context of a DSDBS, two directives must be taken as a guide:

1. *Maximize the parallelism of the process;*
2. *Use the principle of interoperation in the limit that the autonomy of the SGBDED permits.*

The first directive implies that the strategy must execute the largest possible part of the query processing in parallel between, at least, the local Spatial DBS that contain the data sets that were referenced on the query. Thus, taking maximum benefit of the parallelism intrinsic to the distributed context to speedup the query processing.

The second directive implies that the strategy must use all the possibilities of sharing, interdependence and interoperation facilities of the data, meta-data, transactions and resources among the local spatial DBMS components of the DSDBMS in order to optimize the local portion of processing, in particular the geometric calculus, and, also, favoring to reach better parallelism.

The justification of these directives was because of the goal to minimize the total cost of the distributed processing of spatial queries in the DSDBS, which could be translated as the goal of minimizing the volume of data transferred through the communication network, to maximize the parallel processing among the involved local Spatial DBS, dividing the work load among them, and to minimize the manipulation and geometric calculus with data objects that will not be part of the final result of the spatial query.

This result is the basis of this work. The initial hypothesis was that if the two directives proposed in RAMIREZ [12] were followed to propose a new strategy for the distributed processing of a spatial join operation, we might have a more efficient one. Thus, the new strategy introduced in this work follows these two directives.

3 Strategy for the Distributed Processing of Intersection Spatial Joins

The processing of the intersection spatial join between two spatial data sets can have a high cost due to the size and complexity of the spatial objects processed [11]. This operation has classically been resolved by *filter and refinement* strategies, using spatial approximations of spatial objects to reduce the set of candidates to be geometrically processed [9]. There is much work on this subject. They proposed strategies for several scenes of the spatial join processing, from the parallel context until the situation where data sets don't have spatial indices.

We will assume in the rest of this work that the filter and refinement process adopted will be the *Multiple Steps Process* – MSP – proposed by BRINKHOFF et al. [5], where data sets have indices of the type R*-Trees [4]. Also, we will consider that the data sets are distributed between two local Spatial DBS that compose the DSDBS, that is, each one of the data sets is located in a distinct node of the computer network. This is the distribution scene used in the work we find in literature. Only [12] presents the existence of other possibilities of distribution of the spatial data, in particular those based on the subdivision of the space of interest

The distributed processing of the spatial join cannot simply be executed by sending the join operation to the two nodes, which would execute it using a centralized strategy, followed by collecting and matching its two results. This is because none of the two nodes have one of the data sets; therefore it is located in the other node, not allowing the direct use of the MSP.

Thus, the first step of the MSP, relative to the MBR-Join, cannot be carried out, for both nodes only have one of the spatial indices, while the other index is located remotely in the other node. In the same way, the geometric filter step that uses accurate spatial approximations of spatial objects also cannot be carried out, for each node only has one of the sets of accurate spatial approximations of objects. However, the last step of the strategy could be carried out if one of the nodes received the spatial data set from the other involved node. However, this procedure would cancel completely the benefit of the MSP strategy [5].

The idea of fully transferring one of the data sets to the other node is known in the literature as the *Naive* strategy [1]. Based on this idea, for the distributed scene mentioned, we can define a Naive strategy that uses the MSP to process the spatial join as follows, where Figure 1 shows a diagram of its steps.

Naive Strategy: Let R and S be two spatial data sets allocated in two distinct nodes of a computer network, respectively node_R and node_S. First, fully send the smaller data set to the other node and, in this node, build an R*-Tree index. After that, the spatial join processing is carried out by the centralized algorithm MSP.

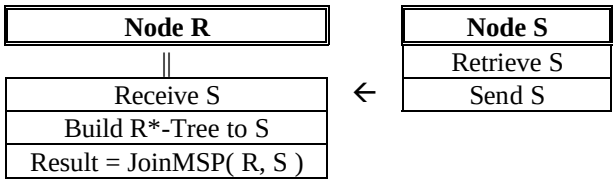


Figure 1: *Naive* strategy

This Naive strategy is applicable to DSDBS of any autonomy degree, being it Integrated, Federated or a *Multidatabase* spatial DDBMS [12].

The main problem that the naive strategy may present is the high cost of execution due to unnecessary processing. This problem has two sources. The first one is that the operation of building the new spatial index may have a high cost, depending on the average size of the spatial objects and the cardinality of the spatial data set S. However, this index has already been built in the other node, thus it is redundant work that could be avoided. An alternative to minimize this cost could be the use of the join algorithms to data sets without spatial indices, such as proposals like [11].

The second source of problems is due to the manipulation of spatial objects that will not belong to the final result of the spatial join. The worst case of this situation is reached when the number of spatial objects that match the predicate of the spatial join is small when compared to the cardinality of the involved data sets. In this case there will be a high unnecessary cost due to the operations of I/O and transmissions over the communication network of a high number of objects that will not be used in the final processing. Also, in this case there is the unnecessary cost with geometric processing and the building of the spatial index with these unnecessary objects.

At last, it is easy to observe that this naive strategy has *zero* degree of parallelism; therefore, as the Figure 1 shows, in each step of the strategy only one node of the DSBDS is working. Also, this strategy does not apply the interoperability principle to optimize none of its steps.

However, there are situations where the result of a spatial join contains almost all the objects of the involved sets. For example, when the spatial join is used to calculate map overlays, the majority of objects fulfills the join predicate. In this situation the cited problems are minimized, particularly if the communication network has a large bandwidth. However, the absence of parallelism persists.

The Literature presents another strategy more efficient than the naive, which is an adaptation of the strategy used by relational DBS for distributed join processing of two relations, which is based on the *semijoin* relational operator to reduce the communication cost [10].

This strategy is the basis of the proposal made by ABEL et al. [1], which is extended, by the same authors, to use with R-Trees in [14]. In [1] a solution to the problem is presented of how to carry out the operation of projection on spatial attributes. This is solved by the proposition of a spatial version of the relational *semijoin* operator: the *spatial semijoin* operator.

The *spatial semijoin* operator was defined as follows: Let R and S be two spatial data sets and S' a set of spatial conservative approximations [5] of objects of S. For the case of the intersection predicate, the *spatial semijoin* operator between R and S' produces a subset R' from R such that its objects intercept some approximations in S'.

This definition implies that the result of the *spatial semijoin* between R and S' certainly contains all pairs of objects that belong to the final result set of the spatial join. However the result set can contain some unnecessary ones. This depends on the type of approximations which was used to generate S' from S. Thus, the strategy proposal in [1] can be summarized as follows (Figure 2 shows a diagram of its steps):

Semijoin Strategy: Let R and S be two data sets located in two distinct nodes of DSDBS, node_R and node_S respectively, where each set has a local R-Tree spatial index:

1. In node_S retrieve S', which is the set of distinct conservative spatial approximations of objects of S, and transmits it to node_R;
2. In node_R, calculate R' as result of a *spatial semijoin* between R and S' and send it to node_S;
3. In node_S, carry out the spatial join between R' and S to get the final result set.

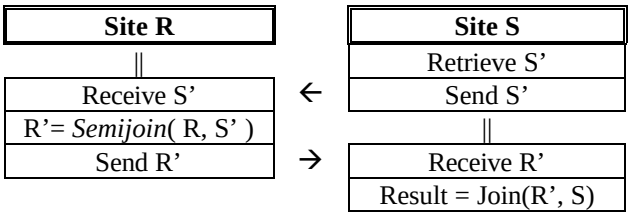


Figure 2: *Semijoin* strategy

Analyzing this strategy we can observe that it basically reduces the transmission cost, which was the objective of the original strategy in the distributed relational DBS. It assumes that the total transmission cost of the subset R' and S' is smaller than the transmission of the smaller between sets R and S. The implementations of the Semijoin strategy presented in [1] and [14], also showed that it induces a reduction of I/O and CPU cost. This is due to the reduction of the manipulation with objects that will not belong to the final result set.

Even though the Semijoin strategy is more efficient than the Naive, with some exceptions [14], we can observe that there are still some things to be improved. The most noticeable problem is the fact that this strategy, as much as the Naive, also does not take advantage of the parallelism proportioned by the distributed processing. We can observe in Figure 2 that on each step of the strategy only one local node of the DSDBS is running, i.e., it is a sequential processing divided between two computers. In this way, the two strategies have the same CPU cost for the geometric processing. The reason for this is on the origin of the Semijoin strategy, which has as a goal only minimizing the traffic in the communication network, once in distributed relational SGBD the cost of local processing is irrelevant in comparison to the one of communication, assumption which is not valid for the spatial context [6][14].

Also, there are no spatial indices for *received* subsets S' and R', something that can be confirmed by experiments carried out in both works [1] and [14]. Thus, it cannot directly apply an efficient strategy for processing of the local joins without additional costs. Finally, in the full transmission of R' to node_S situations can occur where the transmission cost is high, something confirmed by the results in [14]. They indicate that the best case for this strategy occurs when the objects of R and S have nearby average sizes, this happens because when the objects of S are much smaller than the ones of R, the better choice would be transmitting the objects of S for related objects in R'.

Another situation where the Semijoin strategy presents trouble occurs when the *spatial semijoin* operation does not produce a good reduction of set R. When this situation arises the performance can be close, or worst than the one of the Naive strategy. This can

occur when the operation accepts many objects which do not attend the join predicate, or when almost all objects have intersection with some other object in S' [14].

In summary, there are two aspects that can be improved for the presented strategies: the exploitation of parallelism inherent to the distributed processing and the utilization of the existing local spatial indices of the involved data sets. The justification of this statement is in the fact that none of the mentioned strategies is parallel, even for geometric processing – which can have high cost – and they do not use the existing spatial indices to carry out the local spatial joins, since they are not considered interoperating features, leading to processing with higher cost algorithms.

4 MR2 Strategy

Keep in mind the difficulties observed in the two first mentioned strategies, in this section we will propose a new strategy for the distributed processing of the spatial join with intersection predication. We developed our strategy in agreement with two directives presented in section 2. Thus, based on the first directive, our strategy is proposed to make parallel the local geometric processing stage, for we considered that its cost is dominant in the query processing on Spatial DBS [6]. Based on the second directive our strategy solved the problem of not using the local spatial indices through the sharing of theses remote indices. This allowed the optimization of local processing.

In our proposal we considered that the cost of the third step of the MSP – which is relative to processing with the exact geometry of objects – dominates the total time of processing, in particular when the queries require the calculation of the intersection region between objects. As a whole, our strategy praises the exploitation of any spatial index by remote processing, i.e., we consider that the spatial indices can be shared. The central idea is to allow that any local index or accurate spatial approximations of objects can be remotely used by any node in its local processing.

This allows the elimination of the cost of building the index for received data sets and directly using the MSP [5] strategy to carry out local spatial joins that involve remote data sets. So that this is possible, the spatial indices must be modified in such way to be interoperable. In the case of an R*-Tree, we propose the modification of the type of relative references used by the internal nodes of the R*-Tree in such way that they become *global* references in the DSDBS. For this approach, the relative algorithms of the R*-Tree will have to be modified so that they reflect the use of global references.

On the other hand, in order to minimize the I/O cost, the literature about spatial joins based on MSP suggests that the leaves of the R*-Tree contain the accurate spatial approximations of objects pointed by them, which will be used in the geometric filter step of MSP. This is reasonable when the processing is centralized, therefore minimizes the I/O cost [5]. However, for the context of our distributed strategy, these approaches would lead to transmission of data of the all accurate spatial approximations by the time of the transmission of the R*-Tree. As the accurate spatial approximations of objects, in general, they are not so small as MBRs (Minimum Bounding Rectangle) [16], this can mean unnecessary additional cost, in particular when the cardinality of the data set is high and the number of objects that match the predicate of the spatial join is small. In this case there will be transmission of many accurate spatial approximations that will not be used by the geometric filter, unnecessarily magnifying the cost of communication and I/O.

A simple but efficient solution for this case would be only to transmit a small subset of the accurate spatial approximations relative to the objects, which belong to the set of candidate pairs produced by the MBR-Join step of MSP. For such a thing we propose two other changes. First, the leaves of R*-Trees would not contain the accurate spatial approximations of objects anymore, but only its respective MBRs and a global reference for the respective accurate spatial approximations, which would be stored in a separated set. In this mode only the R*-Tree would be transmitted without the accurate spatial approximations, which would be requested for the following steps.

Thus, our strategy uses shared spatial indices and parallel processing and can be defined as follows (Figure 3 shows a diagram of its steps). We call it MR2 – *Multiple steps with Remote indices, version 2* (version 1 was defined with accurate spatial approximations [12])

MR2 Strategy: Let M and E be two data sets located in two distinct nodes of DSDBS, node_M and node_E. Each node has a local R*-Tree, rt_M and rt_E respectively – that have MBR approximations of its objects as geometric keys – and a separated set that has the accurate approximations of its objects for the geometric filter step. The steps of the strategy are as follows:

1. Identify the smaller of the spatial indices rt_M and rt_E. Assuming that rt_E is smaller than rt_M, so node_M will be the Master node and node_E will be the Slave node;
2. Transmit the index of the Slave node to the Master node;
3. In the Master node carry out the MBR-Join generating a set of candidate pairs, called ME1. After that, request to the Slave node the respective remote set of the accurate approximations related to objects in set ME1;
4. In the Master node carry out the *geometric filter* step, to reduce the candidate set of pairs that do not

belong to the final result, generating a new set of candidate pairs called ME2, which contains the pairs that match the predicate of the spatial join as the pairs that remain inconclusive;

5. In the Master node, divide the set ME2 – generating two sets of pairs ME2_M and ME2_E – in such a way to balance the workload for parallel processing and minimize the transmission cost. Transmit the ME2_E set of pairs from the Master node to the Slave node;
6. In parallel, Master and Slave nodes request remote objects specified by its respective sets of pairs ME2_M and ME2_E;
7. In parallel, Master and Slave node execute the exact geometry join process step with the objects from their respective sets ME2_M and ME2_E.

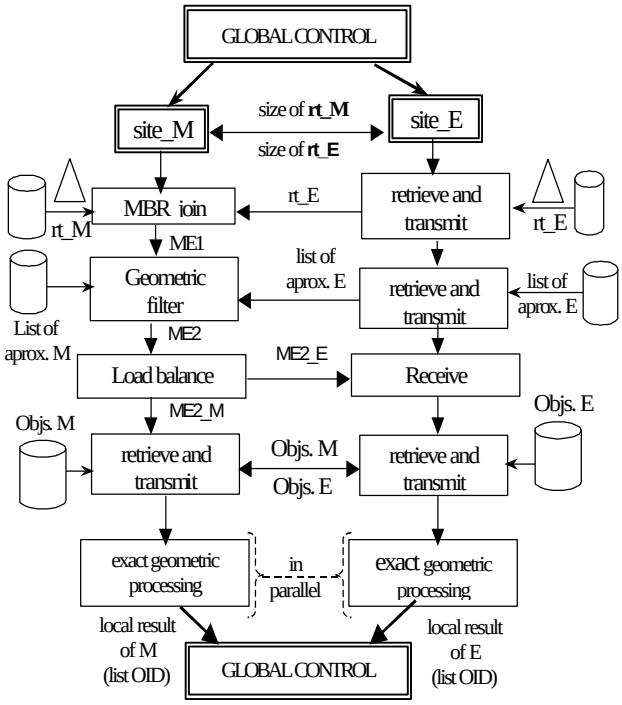


Figure 3: The *MR2 Strategy*

The MR2 strategy intends to prevent the processing of objects that will not belong to the final result set, which optimizes all the involved costs. Also, it minimizes the cost of the communication by the choice of the smallest objects to be transmitted. Because it is parallel, it optimizes the geometric processing costs and, by using shared spatial indices, become efficient and this excuses the cost of building indexes for remote data sets.

Considering only the fact that this strategy is parallel, we can note that it is more efficient than the previous ones in contexts where the CPU cost is dominant, which is the majority of cases of the spatial join queries. In particular, our strategy is more efficient when almost all the objects match the spatial join predicate; or when the difference between the average sizes of objects of two data sets is relevant; or when the size of the data sets is large, i.e., a combination between high cardinality and objects with large average size or when the transmission cost is dominant, i.e., the bandwidth of the communication network is low, or the data traffic is high, a typical case of the Internet today; or when the selectivity of the spatial approximation is poor, that is, sets ME1 and ME2 have many pairs of objects that do not match the join predicate [12].

The worst case of the MR2 strategy occurs when the cardinality of the spatial data sets are small, in combination with a small number of pairs in the ME2 set, i.e., few objects match the join predicate. This is because the CPU cost becomes small, due to the need of few geometric tests or processing with exact geometry of objects, leading to the predominance of the communication cost, at first the I/O cost would be also small due to the fact that few objects are retrieved. However, even in this worst case, the MR2 strategy still presented slightly better performance than other two cited strategies, as it was demonstrated by the experiments, in particular due to the load balancing and the sharing of spatial indices.

5 The Experimental Setup

The goal of our experiments was to carry out a performance study among MR2 strategies, Naive and Semijoin. Thus, we measure total time, I/O time, data transmissions time and geometric processing time. We also measure the data volume transmitted and retrieved (I/O cost).

We implemented the strategies in the specific context of the in the core of a DSDBS, which is integrated and homogeneous. However, as we did not have an operational DSDBS prototype, we decided to implement a workbench for the tests for this purpose, whose architecture can be found in [12].

The strategies were implemented as distributed algorithms [3] to be executed between a client *Global Execution Monitor* and several *Local Execution Monitor* servers. The workbench was implemented in C++ using the *message passing library* MPI to implement the intercommunication part. We use a network with eight Pentium® III/450 MHz (64 Mbytes/NT Server 4.0) computers. The network had a communication bandwidth between 7 and 4 Mbits/s.

The persistent objects had been implemented with a global identification, i.e, a unique identification valid for all nodes of the DSDBS. Thus, the spatial objects, part of the spatial indices and the approximation had a global OID. This allowed their interoperation over several nodes of the DSDBS, in particular this also allowed the sharing of local indices and accurate spatial approximations of objects.

The spatial database used was produced from the digital maps set of "*Macrodiagnóstico da Zona Costeira do Brasil*" produced in 1996 by the Environment, Water Resources and Legal Amazon Ministry of Brazil's

government. The generated objects were convex polygons, the R*-Trees used MBRs as spatial keys and the accurate approximations of spatial objects were based on the 4CRS raster signature proposal by ZIMBRÃO et al. [16].

The spatial data sets were divided in three groups. The first group had data sets that had a variation of the cardinality of objects (group CDD with 1,000 to 85.000 objects that had the same average of vertices). The second group had data sets that had a variation of the average size of spatial objects (group TMOE with objects of 110, 57 and 19 vertices). The last group had data sets that had a variation of percentage of objects that match the join predicate (group CRCR with 10% to 90% of objects that would belong to final result set).

6 Results

The results of the experiments had not only indicated that the MR2 strategy is adjusted for the DSDBS as also they had justified the statements that it is more efficient than the previous distributed strategies.

The MR2 strategy always had better performance than other strategies in all tested situations. Considering the total time of the query, in the worst case, the MR2 strategy was 31% better than the Semijoin strategy and 67% better than the Naive strategy, as Figure 4 shows for a set of 50K objects with an average of 19 vertices and 32% of objects that match join predicate. The general explanation for this fact is that the MR2 strategy optimizes all the involved aspects in the processing of the intersection spatial join.

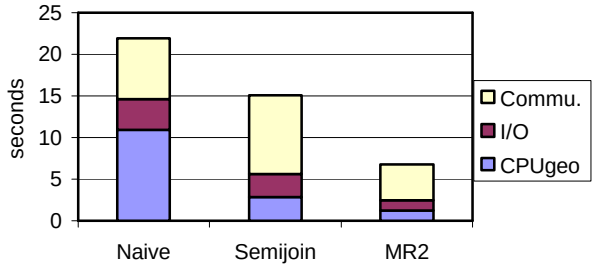


Figure 4: Total time, CCD_G19_50K

The main enhancement is related to the greater parallelism of the MR2, surpassing the Semijoin even when the number of objects that match the join predicate is small, or the data set cardinality and the average size of the objects is large. This can be observed in Figure 5, which shows the CPU time. The time of MR2 varied from 58% to 49% of the time spent by the Semijoin and was 82% of the time spent by the Naive strategy. These were the results for data sets with 1K to 20K spatial objects, which had an average of 112 vertices and had 30% of objects matching the join predicate. Also, the CPU cost will be as high as the number of objects that match the join predicate, or the average size of the objects. This was observed in the experiments with group CRCR, where MR2 strategy got, in the worst case, a gain of 49% in relation to the Semijoin strategy and a gain of 87% in relation to the Naive strategy.

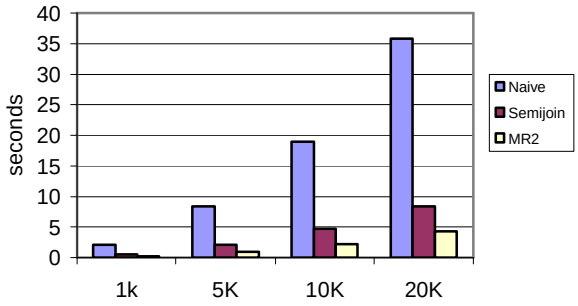


Figure 5: CPU Time for CCD_G112

Observe that the gain in CPU time with the MR2 strategy was also due to the fact that it does not have the cost of building the spatial index for it shares indices remotely. This can be observed in experiments where the cardinality of the involved data sets was high and also the average size of objects was large.

Evaluating the communication cost, the MR2 strategy surpassed the other present approaches since it minimizes all data transmission. First, just as the Semijoin strategy, the MR2 strategy almost eliminates the transmission of objects that will not be part of the final result, with the use of indices and geometric filtering. Second, due to the type of load balancing, the MR2 strategy transfers the smaller objects selected to the final step of the join processing. This aspect, in particular, made it surpass the Semijoin at the variation of the number of objects that matched the join predicate, as show in Figure 6, where the gain varied from 75% to 1%. The worst case occurred when almost all the objects matched the join predicate and had to be transferred.

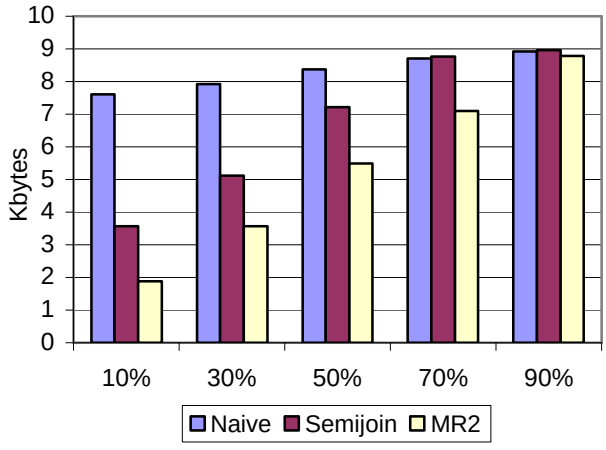
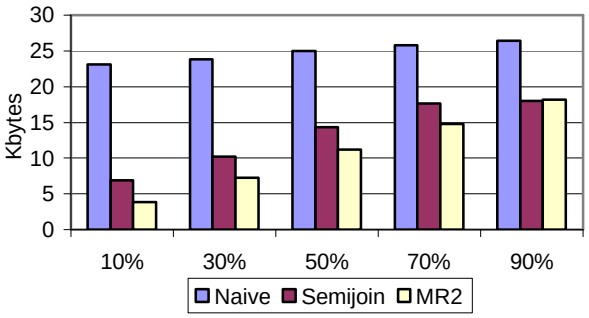


Figure 6: Communication Volume for CRCR.

Only evaluating the communication and I/O costs, the MR2 strategy surpasses the other strategy when the

data sets are large and it has few objects that match the join predicate. In the opposite case, the I/O data volume of the MR2 strategy is equal to the Semijoin, however both are below the Naive strategy for it carries out an additional scan of the data set in order to build the spatial index, as can be seen in Figure 7 as follows.

Figure 7: I/O Volume for CRCR



7 Conclusion

Our results strengthened the evidences that the maximum use of parallelism, inherent to the distribution processing, is relevant. This is because the execution of a spatial query frequently involves an extended and recursive processing, which manipulates complex and large data. Then, if there is not at least the parallel processing between the component systems, that contain the involved data sets, this processing still can be optimized.

Our results indicated that the application of our directive, in relation to the use of interoperation translated by the sharing of local spatial indices, can allow the distributed processing of spatial operations to use the strategies developed for the centralized spatial join with spatial indices and that use geometric filters based on raster signatures, as proposed by [16].

In summary, when we deal with Distributed Spatial DBS, the employed strategies should search for the best interoperability allowed by the autonomy of the DSDBS in order to make the local processing more efficient, as if the processing was centralized in each local SDBS. Also, the strategies should not have to neglect the parallelism potential that can be established among the spatial systems involved in one distributed processing.

In this work we had a series of restrictions. However, our investigations considered other not mentioned possibilities which open a vast field of directions to be taken in future research, in particular when we consider the investigation in Federated DDBS and Multidatabase scenes, with the use of XML and CORBA. We are also investigating other types of spatial operations and predicates, in particular the nearest spatial join, this considering data sets with no spatial indices, or with other spatial access methods not based on the R-Tree family, and also with sets of linear and temporal data.

Bibliographic Reference

[1] Abel, D.J., Ooi, B.C.., et al., “Spatial Join Strategies in Distributed Spatial DBMS”. *Advances in Spatial Databases*, LNCS n. 951, Springer, pp.348-367, 1995

[2] Aref, W., “Query Processing in Distributed Spatial Databases”. *1st Inter. Conf. Interoperating GIS*, Santa Barbara, California, Dec. 1997.

[3] Barbosa, V.C., *An Introduction to Distributed Algorithms*, Cambridge, MIT Press, 1996.

[4] Beckmann, N., Kriegel, H-P, Schneider, R., et al., “The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles”. *Proceedings of ACM SIGMOD Conference*, pp. 322-331, 1990.

[5] Brinkhoff T., Kriegel H.-P., Schneider R., et al., “Multi-Step Processing of Spatial Joins”. *Proc. of ACM SIGMOD Conference*, pp. 197-208, 1994.

[6] Güting, R.H., “An Introduction to Spatial Database Systems”,*The VLDB Journal*, v.3,n.4,pp357-399,1994

[7] Harder, C., *Serving Maps on the Internet: Geographic Information on The Word Wide Web*., ESRI, 1998.

[8] Kumar, B.V., Sinha, P., Bhatt, P.C. P., "DoGIS: A distributed and object oriented GIS". *Proc. of VII Int. Sym. Spatial Data Handling*, pp.263-275. 1997.

[9] Orenstein, J.A., “Spatial Query Processing in an Object-Oriented Database System.” *Proceedings of ACM SIGMOD Conference*, pp. 326-336, 1986.

[10] Özsu, M.T., Valduriez, P., *Principles of Distributed Database Systems*. 2ed, Prentice-Hall,1999.

[11] Patel, J.M., D.J. Dewitt, “Partition Based Spatial-Merge Join”. *Proc. SIGMOD*, pp.259-270, 1996.

[12] Ramirez, M.R., *Processamento Distribuído de Consultas Espaciais*. Tese DSc, COPPE/UFRJ 2001.

[13] Ramirez, M.R., Souza, J.M., Mattoso, M.L.Q., *Operações Espaciais em Banco de Dados Espaciais Distribuídos*. RT-ES- 477/98, COPPE /UFRJ, 1998.

[14] Tan, K-L., Ooi, B. C., Abel, D. J., “Exploiting Spatial Indexes for Semijoin-Based Join Processing in Distributed Spatial Databases”, *IEEE Transaction on Knowledge and Data Engineering*, v. 12, n. 6, pp. 920- 937, Nov 2000.

[15] Wang, F., “A Distributed Geographic Information System on the CORBA”. *GeoInformática*, v. 4, n. 1, pp. 89-115, Mar 2000.

[16] Zimbrão, G.S., Souza, J.M., “A Raster Approximation For Processing of Spatial Joins”. *Proc. of 24th Inter. Conf. VLDB*, pg 24-27, Aug. 1998.