Distributed Spatial Query Processing over MPI

Milton Ramos Ramirez[1], Jano Moreira de Souza[1]

[1] COPPE/Sistemas–UFRJ, IM-UFRJ,

{milton, jano}@cos.ufrj.br
http://www.cos.ufrj.br

Abstract. Communication between multiple database nodes in Distributed Spatial Database Systems (DSDB) must be as efficient as possible. This requires a message passing paradigm fit to achieve this. In this way, we have tried to apply MPI standards to implement DSDB kernel. In this work we present a useful result of this attempt that confirms the feasibility and suitability of MPI standards used on the implementation of DSDB kernel. We present the implementation, using MPI standards, of an efficient solution to distributed processing of the intersect spatial join operation, which combines two large spatial data sets that are located on two distinct nodes of a DSDB. This operator is considered a core operator of any DSDB.

1 Introduction

A Distributed Database System (DDB) – has multiple interrelated databases scattered over a computer network. Thus, it can be viewed as a system base on a distributed memory model if we adopt parallel computing point of view. This way, the use of the message passing paradigm to implement the kernel of DDB is a promising idea.

Existing implementations of DDBs have presented acceptable performance in spite of the communication paradigm it uses [11]. However, the same does not occur to a DSDB – Distributed *Spatial* Database System [2]. The characteristics of spatial databases preclude the direct application of solutions similar to the ones used on conventional DDBs.

As an illustration, the DSDB has been used with the database system component of DGIS – Distributed Geographic Information Systems – which can be defined as those systems that have a number of autonomous geographic systems, not necessarily homogeneous, interconnected by a computer network and which cooperate towards *geographic data processing* with the data stored in their local spatial databases.

This type of processing is considered non-conventional because it can take a long time and have recursive database *transactions* [15]. For instance, when we need to overlap two large maps, the DGIS will submit an ordinary *spatial query* to the DSDB, to retrieve the entire map and its components, which takes a long time and a recursive database operation.

The development of DSDBs presents several interesting challenges [2]. *Efficient distributed query processing* can be considered the main one. The reason is that it is a

central issue for all categories of database systems [11], as well as the fact that the feasibility of a DGIS is dependent on the efficiency of the query processing of its database system component and its capability to offer transparency to the distribution [1].

Hence, we decided to use the message passing paradigm to explore the possibilities of implementation of a DSDB kernel to support distributed query processing for queries submitted by a DGIS. In this scope, we do not address the other issues of distributed spatial databases, such as the spatial data location, global spatial query fragmentation, generation and optimization of the execution spatial query plan.

In this paper we present our first MPI implementation of a kernel of a DSDB that is able to execute the distributed processing of *intersect spatial join operation* between two spatial data sets which are located on two distinct nodes of a DSDB. The spatial join is considered a prime spatial operator of the DSDB and DGIS [6].

In the next section we describe our MPI implementation of the DSDB kernel. In Section 3, we present how the distributed processing of intersection joins was implemented over MPI DSDB kernel. In the last two sections we discuss the result of some tests and the performance of the distributed processing of spatial joins and present some conclusions.

2 The DSDB Kernel Prototype

When constructing a DSDB kernel we must consider three main aspects related to distributed databases: (1) the type of data distribution over the nodes of the DSDB, (2) the autonomy that each local database node has regarding the global DSDB; and (3) the kind of heterogeneity that exists among the nodes that belong to the DSDB.

In respect to type of distribution, we assumed that the database was spatially fragmented in one form as stated in [15]. In particular, the core tests used in the processing of spatial joins were thematically fragmented. Regardless of the kind of heterogeneity, we assumed homogeneous local nodes. In brief, we implemented a homogeneous, integrated distributed spatial database kernel.

Because our first objective was to construct a prototype of a DSDB kernel able to support query processing we concentrated on distributed spatial query processing issues and left other issues for future development. We found in the literature some articles on query processing in DSDBs, such as [2] [7] [9] [13] [14] [17].

A spatial query in DSDB is considered to be a non-conventional query processing mechanism because of the characteristics of spatial data processing. Consequently, it cannot directly adopt the solutions established for the conventional DDB. The query processing in DSDB can be considered to have issues pertaining to both: the *distributed* query processing in conventional DDBs [11] and *spatial* query processing in centralized Spatial Databases Systems [6]. Most of them have not been extensively studied, and there are still scenarios of spatial database distribution to be addressed [17].

In despite of this, in a DSDB the sequence of steps of the query processing does not differ from any conventional DDB. Thus, the architecture of our MPI DSDB kernel prototype, in an integrated and homogeneous approach, could be derived from

the architecture of the integrated and homogeneous DDB, based on a client/server approach. The client part of the kernel is a node of the DSDB that commands the global distributed query processing. It plays the role of the master node. The server parts of the kernel were the nodes of DSDB that would actually execute, among them, the spatial operations needed to answer back the query over its local databases, as can be seen in figure 1.

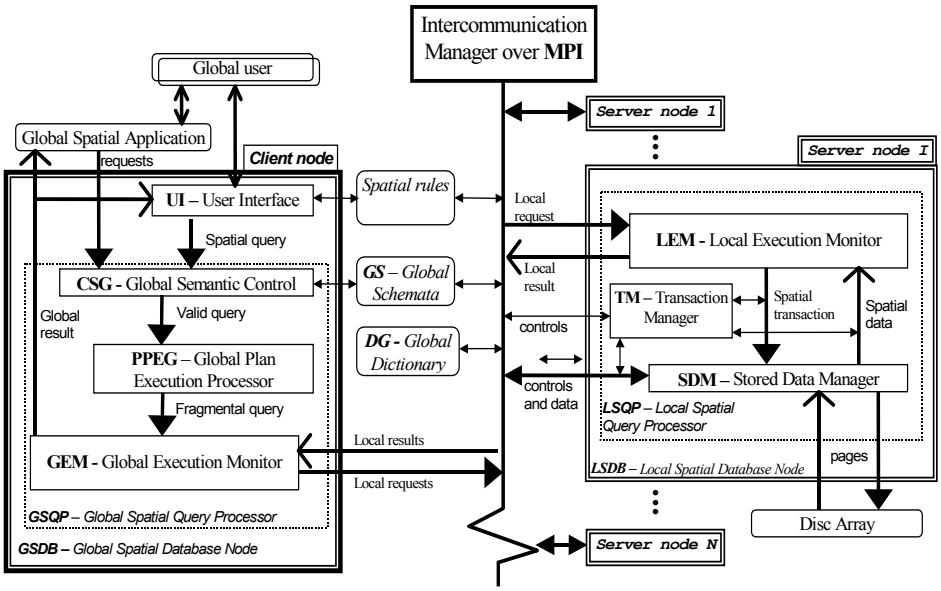

Fig. 1. The Architectures of local and global nodes of an Integrated DSDB

As we can observe in figure 1, our DSDB kernel prototype is composed of one spatial database client node and an array of spatial database server nodes. Because of the low autonomy of integrated DSDBs we can have interoperation at low levels of the manager controls. So, this type of architecture allows the strategies for the distributed processing of spatial operation to be implemented as distributed algorithms [3] among the *Global Execution Monitor* – GEM – of the client node and several *Local Execution Monitors* – LEM – of the servers nodes.

Therefore, after a global query is transformed into a distribution execution query plan, it is submitted to the GEM. The GEM sets in the LEM of nodes that have the spatial data sets involved in the solution of the query. Hence the selected LEM interoperate among them to answer the query. They use subroutines of *Intercommunication Manager* and *Stored Data Manager* to share and change data and controls.

In our integrated architecture all local persistent data objects have a unique *global identification*. Thus all the spatial objects, spatial indices, spatial approximations and its parts have a unique identification (global OID) valid globally for every node of the DSDBS.

This allowed for transparent data interoperation among nodes of DSDB, in particular the sharing of local indices and accurate spatial approximations of spatial objects. Transparency of data distribution is also a key capability of DSDB required by DGIS.

Therefore, when a LEM needs a subset of remote spatial data objects, it submits the request for them directly to the Stored Data Manager of its node. The Stored Data Manager uses the global OID of objects to remotely retrieve them from the nodes where they are stored. The "remote retrieve" is done through the subroutines of the Intercommunication Manager, which is implemented over MPI.

Our prototype was implemented in C++ using the MPI library MPICH for NT v.1.2.0.4. We used a network with eight Pentium® III/450 MHz computers (64 Mbytes/NT Server 4.0) and the network bandwidth was between 7 and 4 Mbits/s.

3 Distributed Processing of Intersect Spatial Join

Our DSDB kernel prototype was based on a workbench that was build to run performance tests of distributed processing of spatial operations. The main use of workbench was to compare the performance of strategies to distribute processing of the intersect spatial join of two large spatial data sets located on two distinct nodes of the DSDB. Experiments produced clear results [14], and we decided to apply the same type of tests used in the workbench to analyze the feasibility and suitably of our MPI DSDB kernel prototype.

We found in the literature three main strategies for the distributed processing of spatial join operation with intersection predicates [1], [2], [13] and [16]. The work by TAN et al. [16] is an extension, by the same authors, of the work presented in ABEL et al. [1]. Both carry through experiments with its proposals using a prototype of Federated DSBD.

RAMIREZ [13] points that there are several alternative strategies for this type of spatial operation, since strategies can vary according to the predicate of the spatial join, types of data set fragmentation and distribution, degree of autonomy of local spatial database components and existence of local spatial indices related to the local spatial data sets.

The processing of the intersection spatial joins between two spatial data sets can have a high cost due to the size and complexity of the spatial objects processed [12]. This operation has been classically resolved using a filter and refinement process, using spatial approximations of spatial objects to reduce the set of candidates to be geometrically processed [10].

We will assume in our work that the filter and refinement process adopted will be the Multiple Steps Process – MSP – propose by BRINKHOFF et al. [5], where data sets have indices of type R*-Trees [4]. Also, we will consider that the data sets are thematically distributed between two nodes of DSDB, which is the distribution scenario found in literature about this subject.

In general, the strategies proposed in the literature are based on the idea of using spatial approximation of objects to minimize the transmission cost due to the reduction of the data sets to be transmitted for the final join. However [1], [2] and

[16] do not leave out the advantage of parallelism provided by the distribution in the final step of the processing of the join operation, neither do they optimize the local geometric processing step, 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 that observation, RAMIREZ et al. [15] developed a new strategy for this type of spatial join operator that makes use of parallelism and interoperation limits posed by the autonomy of the DSDB. So, the new strategy takes maximum benefit of the intrinsic parallelism of the distributed processing to speed up the query execution. In addition, the strategy uses all the possible data and control interoperations among the local nodes of DSDB in order to optimize the local portion of spatial processing, in particular the geometric calculus, and, also, to reach better parallelism. In both cases, the use of the message passing paradigm is set to achieve this type of enhancement on distributed processing.

The first strategy, for distributed processing of spatial joins, namely, the Naive strategy [1] works by fully transferring one of the data sets to the other node and processing the operation utilizing a centralized spatial join algorithm. The main problem with the naive strategy is that it may have high execution cost due to unnecessary processing.

The worst case with this strategy 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 the unnecessarily high cost will ensue, due to I/O operations 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 will be an unnecessary cost due to the geometric processing and the building of the spatial indexes for these unnecessary objects.

The second strategy found in literature is named the Semijoin strategy [1] [16]. It is an adaptation of a strategy used on relational DBS (conventional distributed database) for distributed join processing of two relations, which is based on the semijoin relational operator to reduce the communication cost [11]. In [1] there is a proposal of a *spatial* semijoin operator, which is the spatial version of the relational operator.

Analyzing the Semijoin strategy one can observe that it basically reduces the communication cost by reducing the processing of unnecessary spatial objects. This is achieved through previous spatial join processing (the semijoin) with the spatial indices about involved data sets, which is based on a simple approximation of true spatial objects. This allows the pruning of the subset of spatial objects that will be retrieved and transmitted over the network.

The Semijoin strategy is almost always more efficient than the Naive, with some exceptions [16]. However, there is still something to be improved. The more noticeable problem is the fact that this strategy, as the Naive, also does not take advantage of the parallelism made possible by distributed processing. The last 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 to minimize the data transmission over the network.

Another situation where the Semijoin strategy is problematic occurs when the spatial semijoin operation does not produce a good reduction of a subset of spatial

objects to be sent over the network. When this situation arises the performance can be close, or worse than the one of the Naive strategy. This can occur when the operation accepts many objects that do not belong to the join predicate, or when almost all objects match the predicate of the spatial join.

The third strategy found in literature is named MR2 – Multiple steps with Remote indices, version 2 (version 1 was defined with accurate spatial approximations) [14]. It exploits the parallelism made possible by distributed processing and, applying the interoperation principle, it utilizes the idea of sharing the local spatial indices of the involved data sets to optimize processing of the remote spatial query and to reduce the intercommunication workload.

The MR2 strategy parallelizes the local geometric processing, which is a dominant cost of spatial join processing [6]. It uses the shared spatial indices for efficient processing of local spatial joins, avoiding the cost of building an index for the remote data sets, and it chooses the smallest subset of data objects to be transmitted over the network. Finally, it looks forward to prevent the processing of spatial objects that will not belong to the final result set.

The MR2 strategy is more efficient than the previous ones in contexts where the CPU cost is dominant, which is almost always case with spatial join queries [13]. In particular, this strategy is more efficient when all or almost all the objects match the spatial join predicate; when there is a prominent difference between the average sizes of objects of two data sets; when the data sets will be large, i.e., the combination between high cardinality and objects with large average size; when the transmission cost is dominant, i.e., the bandwidth of communication network is low, or the data traffic is high, typically the case of the Internet today and of mobile computing [8]; or when the selectivity of the spatial approximation is poor.

The worst case of MR2 strategy occurs when the cardinality of the spatial data sets are small, in combination with few objects matching the join predicate. However, even in this worst case, the MR2 strategy still presents slightly better performance than the other two aforementioned strategies [13], in particular due to load balancing and the sharing of spatial indices.

The architecture of the integrated MPI SDBD kernel prototype allowed easy implementation of those three strategies for performance comparison. In the case of MR2 strategy, the MPI paradigm was suitable to implement its parallel algorithm. Also, the global OID interoperability feature provided a good basis to implement the shared and remote use of local spatial indices and accuracy of spatial approximations of objects. It was useful not only to MR2 strategy but also on implementation of other strategies.

For example, in the case of spatial indices used (the R*-Trees), the modification made was to change the type of relative reference used by the internal nodes of R*-Tree index to a global references (global OID) rather than local addresses. For this approach, the relative subroutines of R*-Tree were modified so that they reflect the use of global references.

4 Performance Test

The goal of our experiments was to test the suitability of the MPI SDBD kernel prototype to carry out performance comparison among strategies for processing spatial joins of large spatial data sets. We implemented and ran the MR2, Semijoin and Naive strategies over a spatial database produced from the oceanographic digital maps dataset. The spatial objects tested were convex polygons, the R*-Trees used MBR as spatial keys and the accurate approximations of spatial objects was based on the 4CRS raster signature proposed by ZIMBRÃO et al. [18].

The results of the experiments clearly demonstrate the feasibility and suitability of MPI SDBD kernel to implement and run spatial join algorithms. The test results showed the same result that was obtained with the old workbench [14] and it confirmed the superiority of MR2 strategy over Naive and Semijoin distributed strategies.

5 Conclusion and future works

The use of MPI standard provides an elegant approach to implement a distributed spatial database kernel. In addition, our particular results have demonstrated that the distributed processing of spatial intersect joins between two spatial datasets can be implemented in a feasible and suitable manner on a MPI based DSDB kernel.

Our results strengthen the evidence that MPI allows implementation of distributed algorithms that make excellent use of parallelism. Also, the use of MPI permitted us to set a better and more transparent interoperability on DSDB kernel, which was a significant feature in the implementation of a more efficient strategy for distributed processing of spatial query operators such as spatial joins.

Although in this work we have assumed some simplifications, the results of the experiments carried out until now encourage us to continue on this path. Also, we intend to investigate, in future research, the use of MPI to implement a DSDB kernel for other distributed scenarios, like Federated DDBS and Multidatabases. We are also investigating other types of spatial operations and predicates, in particular the nearest spatial joins for neighborhood matching, considering data sets without spatial indices, or with other spatial access methods not based on the R-Trees family, and also spatial joins of linear and temporal data sets.

In the same way, we are also starting to study the use of MPI to implement a DSDB kernel over wireless networks. Some issues on MPI and wireless are receiving recent attention in the literature, such as fault tolerance and small bandwidth network [8].

References

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

[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] Kranzlmüller D., Kacsuk, P., Dongarra, J., Volkert, J. (Eds.): Recent Advances in Parallel Virtual Machine and Message Passing Interface, LNCS 2474, Springer (2002).

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

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

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

[12] Patel, J.M., D.J. Dewitt: Partition Based Spatial-Merge Join. Proc.ACM SIGMOD, pp.259-270, 1996.

[13] Ramirez, M.R., Souza, J. M.: Processamento Distribuído da Junção Espacial. Anais do III Workshop Brasileiro de GeoInformática - GeoInfo 2001, Set. 2001.

[14] Ramirez, M.R.: Processamento Distribuído de Consultas Espaciais. Tese D.S.c, COPPE/UFRJ, 2001.

[15] 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.

[16] 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.

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

[18] 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.