International Journal of Computer Applications (0975 – 8887)
Volume 140 – No.9, April 2016

An Approach to Query Processing in Homogenously Distributed Spatial Databases

Monika Yadav
Student, Department of Computer Science,
Banasthali Vidyapith, Tonk,
Rajasthan, India

ABSTRACT

Query processing in distributed environment is basic need of many organizations in order to process huge amount of data in less amount of time. There are many approaches provided for parallel query processing in distributed environment. This paper presents the approach which does the parallel query processing in 4 steps. Firstly query fragmentation is done by using SQL parsing, then query redirection to the specified servers containing remotely distributed databases ,then query execution take place at both the servers and at last result of both part of query transferred to Local server where it get accumulated. Then result gets displayed on Geoserver. Different fragmentation techniques are also discussed in order to store fragmented data on distributed sites.

General Terms

Distributed Databases.

Keywords

Database Fragmentation, Query Execution, Query Fragmentation and Query Redirection.

1. INTRODUCTION

This research paper aims at providing solution for parallel query processing in homogenously distributed spatial databases in order to achieve performance optimization. Query fragmentation or partitioning, Query redirection, Query execution on separate servers and then accumulation of result on local server is main focus of this research paper.

There are many tools available in Postgresql (Spatial database) for supporting distributed environment such as Pgpool-I, Pgpool-II, PG-Cluster-I, PG-Cluster-II, Slony-I, Sequoia. Approach used in this research paper does revaluation of Pgpool-II-3.0.4, which provides parallel query as one of its feature. Parallel query feature includes concurrent query execution on multiple servers. But it will redirect the same query to all the servers without query fragmentation [5] [6] [7].

There are many approaches as well for parallel query processing .One approach of parallel query processing includes separation of local and remote references from query according to qualification, then part of query execution on local server and on remote server and then transferring of some component from remote server to local server in order to perform join take place [14].Some features are supported by Postgresql (Spatial database) which are useful in parallel processing. One such feature is Dblink [8], which comes in particularly handy for querying remote PgServers and doing bulk inserts by specifying the full credentials of the remote server. Another feature is Systemdb which contains User defined rules for data partitioning and merging of result via Dblink.

2. DIFFERENT FRAGMENTATION TECHNIQUES

Different fragmentation techniques like Horizontal, Vertical and Mixed Fragmentation are available for non-spatial as well as spatial databases. Two other types of fragmentation technique Bounding Box wise and feature wise division is described here for spatial databases.

One way of fragmenting spatial data horizontally is based on predicate based division and vertically is based on selection by column and other way of fragmenting spatial data is based on use of affinity matrix and bond energy algorithm [2]. Likewise one way of fragmenting spatial data bounding box wise and feature wise is based on use of affinity matrix and bond energy algorithm[2] and other way of fragmenting spatial data bounding box wise and feature wise is discussed here.

2.1 Bounding Box wise Partition

For Bounding Box wise partition whole data get divided into two polygons by midpoint arithmetic calculation and then with the help of SQL query, intersection of each layer with both polygons can be calculated.

2.1.1 Bounding Box description and midpoint arithmetic calculation:

The bounding box is described by 4 numbers; the x-y coordinates of the lower-left corner of the image, followed by the x-y coordinates of the upper-right corner of the image. Bounding Box of an entity is represented by BBox: BBox(x1 y1, x2 y2).By knowing Bounding Box of an entity, whole data can be divided in two Polygons. Bounding Box can be divided into two portions by midpoint arithmetic calculation of x1 and x2. X3 is taken as midpoint of x1 and x2 and is calculating by using equation [1].

$$x3 = \frac{x1+x2}{2} \qquad [1]$$

Polygon1=(x1 y1, x1 y2, x3 y2, x3 y1, x1 y1).

Polygon2=(x3 y1, x2 y1, x2 y2, x3 y2, x3 y1).

Figure 1 shows the partitioning of database into poly 1 and poly 2. F1, F2, F3, F4, F5 represents the features or layers of spatial database. Linear line across partition 1 and partition 2 represents boundary between two polygons.

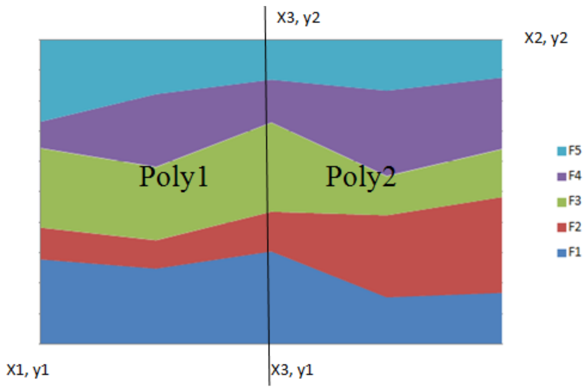

Fig 1: Partitioning of Bounding Box of spatial data

2.1.2 SQL queries for finding intersection of layers with polygon:

After obtaining polygon1 and polygon2, intersection of each Layer with both polygons can be determined. Layers intersecting one Polygon are taken in one server and layers intersecting other polygon are taken in another Server.

SQL Query for finding out whether one layer (e.g. Layer1) intersecting Polygon1 and Storing result in a table (e.g. Table1) for partition 1:

SELECT ST_Intersects (Layer1.the_geom, ST_GeomFromText ('POLYGON ((x1 y1, x1 y2, x3 y2, x3 y1, x1 y1))', 4326)),* INTO schema.Table1 FROM schema.Layer1;

SQL Query for finding out whether one layer (e.g. Layer1) intersecting Polygon2 and Storing result in a table (e.g. Table1) for partition 2:

SELECT ST_Intersects (Layer1.the_geom, ST_GeomFromText ('POLYGON ((x3 y1, x2 y1, x2 y2, x3 y2, x3 y1))', 4326)),* INTO schema.Table2 FROM schema.Layer1;

Likewise SQL queries can be performed for all features or all layers which divide the whole data into two and the data which lies on boundary get replicated on both the servers.

SQL query for finding out data that lies on boundary of two Polygons:

SELECT count (*) FROM schema.Table1 as t1 JOIN schema.Table2 as t2 ON t1.gid = t2.gid where t1.st_intersects=t2.st_intersects;

In bounding box wise partition fragmentation of query is not required because all table names remain same, only size of tables differs. Therefore same query can be redirected to both the servers and the processing takes place in parallel.

2.2 Feature wise partition

Feature wise partition can be done according to need of an organization. One can store most frequently used features together on one server in order to maximize performance and minimize time to access and switching time.

One approach to feature wise partition can be that all utilities get stored on one server and all other feature on another server. For feature wise partition different table can be placed on different servers because one table in PostgreSQL is equivalent to one feature. In feature wise partition, query fragmentation is required which is done by sql parsing.

3. GENERAL DESCRIPTION OF CONCEPTS USED

Various concepts are utilized for designing of approach of parallel query processing .Some important tools and concepts are described in this research paper. One of them is Pgpool-II-3.0.4 and other is Join Processing in Multi Database System which are described here:

3.1 Pgpool-II-3.0.4

Pgpool-II-3.0.4 is an open source tool available for Parallel Query Processing. Pgpool-II is a middleware that works between PostgreSQL servers and a PostgreSQL database client. [5]. Pgpool distributes the data on both the server and then redirect same query on both the server and get the combination of result via dblink.sql but it does not support query fragmentation.

3.1.1 Feature of pgpool of our interest is Parallel Query:

Using the parallel query function, data can be divided among the multiple servers, so that a query can be executed on all the servers concurrently to reduce the overall execution time. Parallel query works the best when searching large-scale data [6].

3.1.2 Tools for implementing parallel query by Pgpool

PgpoolAdmin: The pgpool Administration Tool is management interface for pgpool to monitor, start, stop pgpool and change setting for pgpool. PgpoolAdmin-3.0.3 is suitable for all pgpool-II-3.0 versions [7].

Systemdb: User defined rules for data partitioning and merging of result via Dblink.

Dblink.sql: Its Sql file in usr/local/postgresql/contrib.dblink.sql. It is used to query local Postgresql and remote Postgresql [8]. But it lacks SQL server’s linked server approach or open query that allows for synchronized join between linked servers/databases and is not useful in cases where lots of data needs to with local data.

3.2 Join Processing in Multi Database System [12]

In this approach user will input the query that will change into modified query. Modified query will separate the Local and remote references according to qualification. Then part of query get executed on local server and part of query get executed on HP-SQL remote server and then transferring of some component from remote server to local server take place because in order to perform join both relations involved in the join need to be present at the same site before join operation take place.

This idea of join processing in multi database system is taken into consideration by taking both databases as postgreSQL-8.4.

4. METHODOLOGY

Methodology for parallel query processing in homogenously distributed spatial databases uses three instances of spatial database i.e. Postgresql. Query gets fired from local database to two other instances of remote databases and then accumulative result from two remote databases get transferred to local database and get displayed there on Geoserver.

Whole methodology for communication between three spatial databases is divided into five parts i.e. Query fragmentation, query redirection to remotely distributed databases, query execution on remote databases, perform union after transferring result to local database and displaying result on Geoserver-2.0.

4.1 Query fragmentation

Query fragmentation includes division of query into two partitions so that one part of query will goes to one server and other part of query will goes to another server.

For Fragmentation of query concept of SQL Parsing is used.

SQL Parsing: Parsing of SQL statement is done by “String Tokenizer” method in java and only Table name is parsed in the query.

This scheme is applicable for feature wise fragmentation only among all types of fragmentation methodologies. In all other type of fragmentation techniques table name remains same on both the servers, only size differs.

In horizontal fragmentation technique partition is done on the basis of predicate. Tuples of a table which satisfy the condition get stored in one partition and those which don’t satisfy the condition get stored in other fragment. In vertical fragmentation techniques some set of columns of a table which are commonly used together get stored together in one fragment. In bounding box fragmentation technique fragmentation is done on the basis of geometry or spatial coordinates. In Feature wise fragmentation some set of features gets stored in one fragment on the basis of query and a single table is considered as one feature. Therefore only in feature wise fragmentation different tables are present in fragments which is the only fragmentation technique is used in this research for fragments in remote databases.

4.2 Query Redirection to remotely distributed databases:

Distributed Databases means two different servers in which partitions of database are stored and from a third server let’s say Local Host Query get fired to both remote servers containing partitioned data.

For Query redirection to distributed databases following methodology is used:

4.2.1 LookUp Table Search:

After Query Fragmentation Tables name that query demands is obtained. Then redirection query to that particular server on which the data query asking for resides is done by LookUp Table maintained at Local Host.

LookUp Table: It contains meta information i.e. which table is stored on which server together with their Credential (Host, Username, password, port) is maintained at Local Host.

4.2.2 Connection to servers by extracting credentials from LookUp table:

Extraction of server name at Local host for both part of query is done from LookUp table. Now establishment of connection to specified servers by fetching parameter from LookUp table results in query’s parts redirection.

4.3 Query Execution on remote databases:

A part of query gets executed on one server and another part of Query gets executed on another Server. Result is obtained in the form of result set .Let say Resultset1 and Resultset2.

4.4 Transferring result to Local database and perform Union

4.4.1 Result set transfer:

Transferring of result-set from remote server is done with the help of Local host server's Object by creating a table (which is mentioned by User in Query) or executing the create SQL query with the help of Statement object of Local host.

4.4.2 Performing Union Operation:

Two tables which contain result of user’s query to server1 and to server 2 is obtained. After this any Operation can be applied which contains data from both tables or Join operation can be performed. Then storage of final Result in 3rd table is done. (Drop the table if already exists).

4.5 Displaying result on server

Finally the combined result from both the servers gets populated on Geoserver.

For populating result Geoserver installation is done which mainly includes Geoserver start-up, workspace creation and store creation. As a result of this, PostGIS get connected to Geoserver and layer from PostGIS storing combined result from both the servers can be viewed as a map on Geoserver.

Figure 2 shows overall structure of query processing in homogenously distributed spatial databases

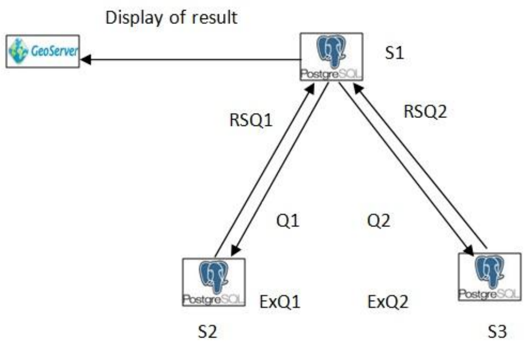

Fig 2: Query processing in homogenously distributed spatial databases.

Annotation used in Figure 2:

S1, S2, and S3: Represents site 1, site2 and site3 respectively.

Q1: one part of query.

Q2: Another part of query.

ExQ1: Execution of query Q1.

ExQ2: Execution of Query Q2.

RSQ1: Result Set of Query Q1.

RSQ2: Result Set of Query Q2.

In this approach, a query gets fired from S1 which gets further partitioned into Q1 and Q2. Q1 and Q2 get transferred to S2 and S3 respectively and they get executed on their respective servers. Then result set RSQ1 and RSQ2 get transferred to S1as a result of query execution on both servers S2 and S3. Finally result gets displaced on Geoserver.

5. EXPERIMENT AND RESULTS

For the experiment software requirement and platform considered was Ubuntu 11.03, Postgresql 8.4, PostGIS 1.52 and Pgadmin III on all the servers.

Feature Wise Fragmentation technique is adopted for this experiment. For feature wise partition 2010 TIGER/LINE DATA [11] was considered. All utilities of database get stored on one server and all other feature on another server. A Feature is equivalent to a table in spatial database. Table 1 shows the Feature Wise Fragments of Tiger Data stored on server 1 and server 2.

Table 1. Feature wise fragmentation of 2010 TIGER/ Line Data

Features stored on server1	Features stored on server2
arealm	linearwater
areawater	pointlm
check	primaryroads
counties	prisecroads
facesah	rails
facesmil	roads
-	states

Query fragmentation is done on the basis of table name extraction from the query by SQL parsing. Queries get fired from Local Host and then redirection of queries to servers is done in accordance with feature present in query.

SQL Query on 2010 TIGER/LINE DATA taken for experiment:

Select name10, statefp from tiger.states as s1, tiger.areawater as s2 where s1.mtfcc10='G4000' and s2.mtfcc='H2030';

Features states and areawater of 2010 TIGER/LINE DATA are on different servers. A query is fired from Local Host having instance of PostgreSql which get parsed and table names are extracted from the query. Therefore after Query Fragmentation table name states and areawater are obtained.

Now separate query for server1 and server2 are query 1 and query 2 respectively.

Query1= select name10 from tiger.states as s1 where s1.mtfcc10=‘G4000’;

Query2= select statefp from tiger.areawater as s2 where s2.mtfcc=‘H2030’;

Now Query Redirection is done and lookup table is searched for redirection. By fetching credential from lookup table entries, part of query having table name “states” is redirected to server1 and another part of query containing table name “areawater” is redirected to server2.

Then Query Execution takes place at both the server. Table 2 shows execution plan of query1 and query2.

Table 2. Execution Plan of Queries

Parameters	Sequential scan on areawater	Sequential scan on states
Cost	0.00 – 37792.51	0.00-3.30
Rows	484603	104
Width	3	9
Actual Time	0.025- 1265.499	0.010-0.101
Actual Time Rows	485571	104
Loops	1	1
Filter	(mtfcc)::text = 'H2030'::text	(mtfcc10)::text = 'G4000'::text

By applying this approach execution of query on both the servers done in parallel and then transferring of result set done to third server. Therefore parallel execution reduces total run time. Total run time for execution of queries in parallel was 693.697 ms excluding time taken for redirection of query on both servers and transferring result back to main server.

Query redirection time and transferring result back can vary depending on various factors like network traffic, network speed, latency etc. which may affect the performance. Types of queries fired on particular database also affect the performance. Therefore fragmentation techniques based on query information can be taken into consideration [2] for storing fragments of data on different servers.

6. CONCLUSION

This approach can be used for benchmarking spatial databases. Different types of benchmarking like functional, performance and Database benchmarking can be done by collecting results from application of this approach and different types of fragmentation techniques and storage methods can be considered for storing data at different sites.

It can also be used by many Organizations having large scale of data distributed over different servers and by implementation of Query Fragmentation and only transferring of result set rather than whole data results in performance gain. It will also solve overhead of data having only at one site and it will also result in distribution of load on one server only.

This research work is useful in many Organizations like E-Governess, Municipal Corporation in order to fire query on distributed Database (Postgresql-8.4) and in order achieve optimization of performance. Further work can be done by implementing the same approach considering heterogeneously distributed spatial databases according to the need of organization.

7. REFERENCES

[1] Stefano Ceri, Giuseppe Pelagatti, “Levels of Distributed Transparency, Distributed Database Design,, Translation of Global Query to Fragment Query,” in Distributed Databases: Principles and Systems, pg-37-126.

[2] Monika Yadav, Neha Arora, “Fragmentation Techniques for Distributed Storage of Spatial Data”, Proceedings of IRF International Conference, 5th & 6th February 2014, Pune India.

[3] PostGIS Reference, [Online] Available: http://postgis.refractions.net/documentation/manual-1.3/ch06.html

[4] Spatial Functions, [Online] Available: http://postgis.refractions.net/docs/ST_Intersects.html

[5] Features of pgpool, [Online] Available:http://pgpool.projects.postgresql.org/

[6] Pgpool tutorial, [Online] Available: http://pgpool.projects.postgresql.org/pgpool-II/doc/tutorial-en.html.

[7] PgpoolAdmin features and installation, [Online] Available: http://pgpool.projects. postgresql.org/pgpoolAdmin/doc/en/install.html

[8] Dblink description, [Online] Available: http://www.postgresonline.com/journal/archives/44-Using-DbLink-to-access-other-PostgreSQL-Databases-and-Servers.html

[9] Geoserver-2.1 User Manual, [Online] Available:http://docs.geoserver.org/stable/en/user/

[10] Geoserver Information, [Online] Available: http://workshops.opengeo.org/

[11] 2010 TIGER/Line Shapefiles Technical Documentation. [Online] Available: http://catalog.data.gov/dataset/tiger-line-shapefile-2010-series-information-file-for-the-2010-census-block-state-based-shapefi.

[12] S.Bandhopadhyay, DataManagement (New Dimension and Perspectives), Join Processing in multidatabase.

[13] P.Bernstein, N.Goodman, E.W ong, C.Reev and J.Rothnie, Query Processing in a System for Distributed Databases.