ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

Query Processing In Distributed Database Through Data Distribution

Abhijeet Raipurkar[1], G.R. Bamnote[2]

Final Year M.E. student, Dept. of CSE, PRMIT & R, Badnera (M.S), India[1]

Professor & Head, Dept. of CSE, PRMIT&R, Badnera (M.S), India[2]

ABSTRACT: **Query processing in a distributed system requires the transmission of data between computers in a network. The arrangement of data transmissions and local data processing is known as a distribution strategy for a query. Two cost measures, response time and total time are used to judge the quality of a distribution strategy. Various algorithms are used that derive distribution strategies which have minimal response time and minimal total time, for a special class of queries. The optimal algorithms are used as a basis to develop a general query processing algorithm. The integration of a query processing subsystem into a distributed database management system is used for analyzing query response time across fragmentations of global relations. Distributed query processing is an important factor in the overall performance of a distributed database system. The database queries access the applications on the distributed database sites thus the main problem incurred is the minimization of the total operating cost i.e. communication cost and processing In order to optimize queries accurately, adequate information must be accessible to determine which data access techniques are most effective e.g. table and column cardinality, organization information, and index availability.**

Keywords: Fragmentation, optimize queries, table and column cardinality, index

I.INTRODUCTION

Distributed database plays an important role in today era where information dependency is more and all sorts of people need access to companies' databases. In addition to a company's own employees, company's customers, potential customers, suppliers, and vendors wants to access the information. It is possible for a company to have all of its databases concentrated at one mainframe computer site with worldwide access to this site provided by telecommunications, networks, including the Internet. Although the management of such a centralized system and its databases can be controlled in a well-contained manner and this can be advantageous, it poses some problems as well. For example, if the single site goes down, then everyone is blocked from accessing the databases until the site comes back up again. Also the communications costs from the many far PCs and terminals to the central site can be expensive. One solution to such problems, and an alternative design to the centralized database concept, is known as distributed database.

The idea is that instead of having one, centralized database, data is going to be spread among the cities on the distributed network, each of which has its own computer and data storage facilities. All of this distributed data is still considered to be a single logical database. When a person or process anywhere on the distributed network queries the database, it is not necessary to know where on the network the data being sought is located. The user just issues the query, and the result is returned. This feature is known as location transparency. This can become rather complex very quickly, and it must be managed by sophisticated software known as a distributed database management system or distributed DBMS.

A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed, different users can access it without interfering with one another. Collections of data (e.g. in a database) can be distributed across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Replication and distribution of databases improve database performance at end-user worksites.

To ensure that the distributive databases are up to date and current, there are two processes: replication and duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be very complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources. Duplication on the other hand is not as complicated. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, changes to the master database only are allowed. This is to ensure that local data will not be overwritten. Both of the processes can keep the data current in all distributive locations.

II. BACKGROUND OF PRESENT SYSTEM

A distributed database system is the combination of two different technologies used for data processing: Database Systems and Computer Networks. The main component of a database is the data which is basically collection of facts about something. This something may be the business data in case of a business corporation, strategic data in case of a military database etc. Distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. Retrieval of data from different sites in a DDB is known as distributed query processing. Distributed processing performs computations on numerous CPUs to attain a single result. [2] In query processing, the database users generally specify what data is required rather than specifying the procedure to retrieve the required data. Thus, an important aspect of query processing is query optimization. Now in query optimization, the optimizer of the database system finds a good way to execute the queries. [3] Query processing is more complex and difficult in distributed environment in comparison to centralized environment as large number of parameters affect the performance of distributed queries, relations may be fragmented and/or replicated, and considering many sites to access, query response time may become very high [2][1] The distributed query optimization has several problems related to the cost model, larger set of queries, optimization cost, and optimization interval. The area of query optimization is very large within the database field.[10] The goal of Distributed Query Processing (DQP) is to execute such queries efficiently in order to minimize the response time and the total communication cost associated with a query. [2]. Therefore it seems logical to look at potential benefits in relation to their costs. There are three major activities [6] in the processing of distributed database system, in the first phase the database is fragmented, in second phase some complex mechanism is used to allocate the database fragment to the different sites and in the third phase the execution of task takes place. It is believed that an effective database fragmentation improves the performance of the database. No doubt fragmentation increases the complexity of physical database design but it significantly impact performance and manageability [7].

A query normally has many possible execution strategies, and choosing a suitable one for processing a query is known as query optimization. A query is expressed by using a high-level language such as SQL (Structured Query Language) in relational data model.[1] The main function of a relational query processor is to transform a high-level query into an equivalent lower-level query (relational algebra), and the transformation must achieve both correctness and efficiency. Execution strategy for the given query is implemented by lower-level query. Since data is geographically distributed in distributed relational database system, the processing of a distributed query is composed of the following three phases:

- local processing phase,
- reduction phase, and
- final processing phase

The local processing phase basically involves local processing such as selections and projections.

The reduction phase uses a sequence of reducers (i.e, semijoins and joins) to reduce the size of relations.

The final processing phase sends all resulting relations to the assembly site where the final result of the query is constructed.

A straightforward approach of processing a distributed query would involve sending all relations directly to the assembly site, where all joins are performed. The allocation of the data influences the performance of the distributed systems given by the processing time and overall costs required for applications running in the network [8]. In distributed query processing, partitioning a relation into fragments, union of the fragments to form a whole relation, and transferring a relation/fragment from one database to another database are common operations.

The problem of finding the optimal partition structure for a given set of privacy constraints and query workload can be shown to be intractable. Heuristic search techniques based on Greedy Hill Climbing to come up with nearly optimal solutions.

III. QUERY PROCESSING

DDBS adds to the conventional centralized DBS some other types of processing expenses, because of the additional design (hardware & software) to handle the distribution. These expenses present as the cost of data transfer over the network. Data transferred could be, intermediate files resulting from local sites, or final results need to be sent back to the original site that issued the query. Therefore, database designers are concerned about query optimization, which target minimizing the cost of transferring data across the network.

Critera for measuring the cost of a query evalution strategy for centralized DBMSs number of disk accesses (# blocks read / written) and for distributed databases, additionally the cost of data transmission over the network and Potential gain in performance from having several sites processing parts of the query in parallel.

Join queries in distributed database are ship whole, fetch as needed, semi joins and bloom joins and each of these join strategies are having their own advantages and disadvantages. Main considerations of query processing in distributed databases are:

- Communication cost
- If there is several copies of a relation, decide which copy to use
- Amount of data being shipped
- Relative processing speed at each site
- Site selection

A. Objectives of Query Processing

In the centralized database system, there are many ways for executing the queries. Its expected cost is mainly the CPU cost and I/O price. It aims at making queries cost lowest. In the distributed database system, the query optimization includes two parts: the query strategy optimization and local processing optimization. And the query strategy optimization is more important between them. There will be several strategies in the same query due to that the data are stored in different sites. The system resource and response time while each strategy costs is also different. So the expected cost should include corresponding communication cost besides
CPU cost and I/O price. Namely the expected cost is the sum of CPU cost, I/O price and communication cost. The formula of communication costs can be estimated roughly as follows:

$TC(x)=C0+x*C1$

Here,

X stands for data transmission amount, usually its unit is b (bit) for computing.

C0 is short for the time which is going to take on the initial communication between two sites, which is determined by the communication system and almost a constant, its unit is s (second).

C1 is for the transmission cost of the unit cost (the reciprocal of data transmission speed), namely the unit data transfer times, and its unit is s/b.

An objective of DDBMS is to process distributed queries efficiently and also providing availability and reliability. [1] In a distributed execution environment, we consider two time consumption estimates namely: total time or response time. [5]

The Total cost is the sum of the time consumed by each processor, regardless of concurrency.

Local processing time: (CPU + I/O) time

Communication time: fixed time to initiate a message + time to transmit the data

The response time is the elapsed time between the initiation and the completion of a query.

Distributed query processing is to translate a high-level query on a single logical distributed database (as seen by the users) into a low-level language on physically distributed local databases. In distributed query processing, the total cost should be minimized for executing a distributed query. [5] The query execution involves only a local processing cost when no relation is fragmented in a distributed DBMS. On the other hand, if relations are fragmented, a communication cost is incurred in addition with the local processing cost. The aim of distributed query processing is to minimize the total execution cost of the query which includes the total processing cost (sum of all local processing costs in participating sites) of the query and the communication cost. The local processing cost of a distributed query is evaluated in terms of the number of disk accesses (I/O cost) and CPU cost. The CPU cost is incurred when performing data operations in the main memory in participating sites. The I/O cost can be minimized by using efficient buffer management technique. In a distributed query execution, the communication cost is required to exchange data between participating sites. Hence, the communication cost depends on several factors such as the amount of data transfer between participating sites, the selection of best site for query execution, the number of message transfer between participating sites, and the communication network. In case of high-speed wide area networks (with a bandwidth few kilobytes per second), the communication cost is the dominant factor and the optimization of CPU cost and I/O cost can be ignored in such cases. The optimization of local processing cost is of greater significance in case of local area networks.[1][2]

IV.QUERY OPTIMIZATION

B. Semi Join Based Query Optimization

If the data with complete relationship were transmitted in the network, it must bring redundancy. When one relationships transmitted to other sites, not all the data participate in the join-operation or could be used. So, the data which don't participate in the join-operation or useless will not be in the network transmission. The basic principle of query optimization strategy based on semi-join operation just reduces the data quantity in relationship operation and the data transmission among sites.

The definition of semi-join operation is "Semi-join is the algebraic relationship operation derived from theconnection and projection operation."

If there were two relationships named R and S on the site 1 and site 2 respectively, and property A and B are the two properties respectively on them. So the semi-join in public property R.A = S.B can be expressed as:

$R \infty_{A=B} S = \Pi (R \infty_{A=B} S)$□ --------------------(1)

Formula (1) can be expressed as:

$R \infty_{A=B} S = R \infty_{A=B} (\Pi B(S))$ ----------------------------(2)
and The semi-join operation is not symmetry, namely
$R \infty S \neq S \infty R$. --(3)
The full connection operation can be expressed by the semi-join operation as the intermediate steps. With the semi-join operation, the full connection operation in public property R.A = S.B can be expressed as:
$R \infty_{A=B} S = (R \infty_{A=B} S) \infty_{A=B} S = (R \infty_{A=B} (\Pi_B(S)) \infty_{A=B}$ S…………….(4)

The connection process and cost estimates using the semi-join optimization method and its cost estimates are as follows :
1) With the public property B, $\Pi_B(S)$ can be projected between the relationship R and S on site 2.
2) Transmit the result of $\Pi_B(S)$ from site 2 to site 1, the cost is:
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □C0□+□C1*size(B)* val(B[S])----------------------(5)
size(B) stands for the length of property B, and val(B[S]) stands for the quantity of different values within property B in the relationship B.
3) Calculate the semi-join on site 1 and then set the results for R', then $R' = R \infty_{A=B} S$.
4) Transmit R' to site 2 from site 1, the cost is:
C0□+C1*size(R)*card(R ') ----------------(6)
card(R ') stands for the quantity of records in R'.
5) Do the connection operation on the site 2, then $R' \infty_{A=B} S$.
The total cost using the semi-join method for the query optimization is:
Csj=2C0□ C1(size(B)* val(B[S])□size(R)*card(R'))----(7)
If we transmit the relationship R from site 1, then executive the connection operation with S on the site 2, the total cost is:
Cnj=C0□ C1*size(R)*card(R) --------------------------- (8)
In the process of query optimization, there may be several methods using the semi-join algorithm to optimize the connection query. Then best method should be selected after calculating the cost of each available semi-join scheme. Next, lowest site cost can be selected and calculate the full connection's cost. Finally determine the optimal method after comparing the two schemes.

C. SDD-1 Algorithm

In the SDD-1 algorithm, it uses the semi-join algorithm to handle the connection among the relationship and cut them. When all of the relationship is the maximum contract, then transfer it to a site where the query can be executed. The query operation is not always ended on this site [6].

SDD-1 algorithm has three important characteristics as follows:

1) It uses the semi-join operation to handle strategy.2) The relationship of the whole sites is not repetitive and fragmented.3) During price estimation of the whole algorithm, the transmission cost to the starting site is not calculated.

SDD-1 algorithm consists of two parts:

- the basic algorithm and
- the post-optimality.

The basic algorithm evaluates the factors of execution strategy such as cost and profit according the reduced price formula. Then the program set cut by semi-join operation will be given. Finally, the most beneficial execution strategy will be decided. But the efficiency of this strategy may not be optimal.

Post-optimality is the process of amending the implementation of basic algorithm to get more rational operation.

A query graph is given. The basic algorithm is described as follows:
1) Evaluate the income of all semi-join programs in the query graph.
2) Choose the semi-join with maximum income, execute it and recount the income of all the semi-join.
3) Execute step 2 circularly until all of the semi-join value of whose income is greater than 0 has been executed.
4) Select the site with minimum communication cost as the last executing site. Transmit all the relationship to the site and connect them to get the ultimate result.

The post-optimality can update the algorithm in two ways:
1) If the site where the last semi-join operation cut the relationship is the last executing location, the last semi-join operation can be canceled.
2) Correct the semi-join flow chart from the circulation of the basic algorithm. The cost of the original semi-join operation may be high, but the income evaluation is very large. If there is this kind of situation, the semi-join operation can be executed after the relationship is cut short. Then, the order of semi-join operation can be corrected.
Although the SDD-1 algorithm can cut short the relationship effectively and reduce the communication cost, there are some disadvantages, such as the complexity of the algorithm. When the number of records is very large, the cost on query and search will increase rapidly. Moreover, the SDD-1 algorithm doesn't make full use of the distributivity of distributed database system. All of the semi-join operations are executed in order and it will increase the response time of query to a certain extent.

D. Partitioning Algorithm

Hill-climbing is a heuristic in which one searches for an optimum combination of a set of variables by varying each variable one at a time as long as the objective value increases. The algorithm terminates when no local step decreases the cost. The algorithm converges to local minima. An initial fragmentation of the database is considered which satisfies all the privacy constraints.
Initial Guess

The initial state is obtained using the weighted set cover. Refer [15] for details of the algorithm. Algorithm for Weighted Set Cover: - Assign a weight to each attribute based on the number of privacy constraints it occurs in. - Encrypt attributes one at a time starting with the one which has the highest weight till all the privacy constraints are satisfied.
Hill Climbing Step:
Then, all single step operations are tried out
(1) Decrypting an encrypted column and placing it at Server 1.
(2) Decrypting an encrypted column and placing it at Server 2.
(3) Decrypting an encrypted column and placing it at both servers
(4) Encrypting an decrypted column and placing it at both servers.

From these steps, the one which satisfies privacy constraints and results in minimum network traffic is considered as the new fragmentation and the process repeats. The iterations are performed as long as we get decomposition at each step which improves over the existing decomposition using the cost metric discussed before.

V. ANALYSIS OF EXISTING SYSTEMS

Joins and semi joins are primitive operations used to extract required information from one, two or multiple tables. Focus is given on computing and analyzing the performance of joins and semi joins in distributed database system. The various metrics that will be considered while analyzing performance of join and semi join in distributed database system are Query Cost, Memory used, CPU Cost, Input Output Cost, Sort Operations, Data Transmission, Total Time and Response Time.

Join is one of the most imperative operations in database theory that is used to extract information from two or more than two tables. Technically join operation is one of the special cases of Cartesian product. In join unlike Cartesian product before concatenation the tuples of the join tables are checked against specified condition. There are various types of joins like equi-join, self join, inner join, outer join etc. Independent of type all of these are used to extract data from two or more tables. The center of attraction in this study will be equi-join one of the most frequently used type of join.

A semi-join is one of the important operations in relation theory that is used to optimize a joins query. Semi join [3] is used to reduce the size of relation that is used as an operand. A semi-join from Ri to Rj on attribute A can be denoted as Rj ∞ Ri . Research shows that semi joins are very helpful in optimizing the join query by reducing the quantity of data exchanged. But one of the darken side of using semi join is that it increases the local processing cost as well as number of message. It returns rows that match an EXISTS sub-query without duplicating rows from the left side of the predicate when several rows on the right side satisfy the norms of the sub-query. The research has shown that Semi-join and anti-join transformation cannot be done if the sub-query is on an OR branch of the WHERE clause.

The objective of semi join in distributed database is to reduce the data transmission [2] from one site to another.

Since reduced cost and advanced communication technology gives birth to the idea of Distributed Database Management Systems that turn out to be an integral part of many computer applications. Distributed Database [7] system is cluster of distributed computers that are coupled with one another with the help of some communication media (like twisted pair, coaxial cable, fiber optics, satellite etc.) on which a database is allocated and placed. It is obvious that a query may have different equivalent transformation that lead to different resource consumption. So in distributed database system one has to keep in mind the consumption of resources while selecting the execution strategy for the query. So while execution distributed query one has to keep in mind various factors like equivalent relational algebra's operations, placement of data and application programs, ordering of relation algebra's operations, bytes transferred from one site to another, Total_Time, Response_Time etc. Now let us understand the meaning and significance of Total Time and Response Time. In the distributed cost model [7] [5] total time which is computed by adding all the cost components (Local Processing Cost and Communication Cost) of a query, whereas Response Time is computed as an elapsed time from the starting to completion of query.

Mathematically the Total_Time and Response_Time are computed as follow:

Total_Time = TCPU * # Instructions + TIO * IO + +TMessage * #Messages +TTCost * #Bytes
Response Time
TCPU * # SInstructions + TIO *SIO + +TMessage * # S_Messages +TTCost * #SBytes

In distributed database system the analysis shows that join approach gives its best in data transmission when a relation having lower cardinality is transmitted to the location where a relation of upper cardinality and larger tuple size is placed. In regard to total time it is clear from above analysis that the query executed with semi join possess lesser total time when data transfer is more. The data transmission in a distributed query using semi join is always lesser than the data transmitted in distributed query using joins operation however data accessed using semi join may be larger than join operation. Semi joins implement more operation as compare to join, but it reduces the number of bytes transferred from one site to another to great extent. Semi joins are beneficial if the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

transmission cost is of main consideration, otherwise joins will be preferred.

VI.FUTURE SCOPE

The query optimization is one of the most important research directions in whether the centralized database or distributed database. Because of the complex establish environment and rich technology content of distributed database, there are still aspects deserving further study. Query optimization technology in the distributed database will be more and more perfect in future. The proposed work can be used to handle fuzziness is in the form of approximate values or linguistic variables, which can be applied only in queries. Though the fuzziness can be incorporated by storing the fuzzy value inside the database, it may not be the efficient method in the real time. Fuzzy databases are still not popular among the people because they are reluctant to replace their crisp data by fuzzy data before they are convinced. Various fuzzy database models, including relational and object-oriented databases, have been proposed over the past thirty years and tremendous gain is hereby accomplished

VII. CONCLUSION

Today's business environment has an increasing need for distributed database and client/server applications as the desire for reliable, scalable and accessible information is steadily rising. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users. However, there is some complexity when attempting to manage and control distributed database systems. A distributed database allows faster local queries and can reduce network traffic. With these benefits comes the issue of maintaining data integrity. Single big server could hardly handle requirement of high availability, data warehousing and fast data storage simultaneously. The distributed database satisfies them by separating functions at low cost.

Proposed work for query processing handles two important issues in data distribution i.e. minimizing query response time through partitions and handling fuzziness in database ny translating fuzzy queries into SQL.

REFERENCES

[1] A.Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. O (plogn)approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In Proc. 37th Ann. ACM STOC, pages 573581, 2005.
[2]Yan T, IacobesnM, Garcia-Mo Lina H,et al, Introduction of Query optimization of distributed database. Paris,FR: WAM Press, 1999.
[3]G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani,U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In Conference on Innovative Data Systems Research,2005.
[4] D.Chiang, L.R. Chow and N. Hsien, "Fuzzy information in extended fuzzy relational databases", Fuzzy Sets and Systems 92, pp.1-1(1997).
[5] G.J. Klir and B. Yuan [2001], "Fuzzy Sets and FuzzyLogic: Theory and Applications", Prentice Hall, Inc. Englewood Cliffs, N. J., U.S.A.
[6] Nilarun Mukherjee, Synthesis of Non Replicated dynamic Fragment Allocation Algorithmin Distributed Database System", Published in Proceeding of international conference on advances in Computer Science , 2010
[7] Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database System", Fifth Edition,Pearson Education, second Impression, pp 894, 2009.