International Journal of Information Technology and Management

Vol. X, Issue No. XV, May-2016, ISSN 2249-4510

QUERY OPTIMIZATION IN DISTRIBUTED RDBMS

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. X, Issue No. XV, May-2016, ISSN 2249-4510

Query Optimization in Distributed RDBMS

MD. Mustfa Alam[1] Dr. Saket Bihari Singh[2]

[1]Research Scholar, Magadh University, Bodhgaya

[2]Professor, Dept. of Mathematics, A. M. College, Gaya

Abstract – Database queries have become increasingly complex in the age of the distributed DBMS (DDBMS). This poses a difficulty for the programmer but also for the DDBMS. Query optimization is a difficult enough task in a non-distributed environment. Anyone who has tried to study and understand a cost-based query optimizer for a relational DBMS (such as DB2 or Sybase SQL Server) can readily attest to this fact. When adding distributed data into the mix, query optimization becomes even more complicated. This paper focuses query optimization in distributed RDBMS.

Keywords: Database Queries, Distributed DBMS, RDBMS

----------X----------

1. INTRODUCTION

Distributed and parallel processing is an efficient way of improving the performance of Database Management Systems (DBMSs) and applications that manipulate large volumes of data (Valduriez, Ozsu, 1999.). A distributed database management system (DDBMS) supports the formation (creation) and maintenance of distributed databases, where data are stored at different sites connected through a network. An objective of DDBMSs is to present an easy and unified interface to the users so that they can access the databases as if there were a single database (Liu, Yu, 1993). Another important objective of DDBMS is to process distributed queries efficiently in addition to providing availability and reliability. Distributed database systems (DDBS) and distributed computing systems (DCS) differ in the resources to be shared. DCS share hard disks and printer set, while DDBS distribute data, where the data as well as operations on the data items are equally important (Upadhyaya, Lata, 2008).

Since data are geographically distributed in such a 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 (Wang, Chen, 1994). The local processing phase involves local processing such as selections and projections; the reduction phase uses a sequence of reducers (i.e, semi joins and joins) to reduce the size of relations; and the final processing phase sends all resulting relations to the assembly site where the final result of the query is constructed. Clearly, a straightforward approach to processing a distributed query would be to send all relations directly to the assembly site, where all joins are performed. This naive method, however, is unfavorable due to its high transmission overhead and because little parallelism is exploited. 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 (Liu, Yu, 1993).. The optimizers of R* (Hass, 1982) and Distributed INGRES (Stonebraker, Neuhold, 1977), take both local processing costs and communications costs into account. In R*, a join between two relations is performed at a single site by using the nested-loop method or the merge-scan method. For a general query, R* exhaustively enumerates all possible sequences of joins with all possible join methods and allocates joins at each possible site. Since each join is performed at only a single site, existence of multiple processors at different sites is not considered for improving performance through parallel execution. In contrast, Distributed-INGRES uses the "fragment and replicate" query processing strategy (Epstein, *et al.,* 1978). The strategy requires one of the relations referenced by a query to be fragmented and other relations to be replicated at the sites that have a fragment of the fragmented relation. The query is decomposed into the same number of sub queries as the number of sites and each sub query is processed at one of these sites. Its main feature is that it allows parallel processing.

Many other algorithms (Chen, Li, 1989, Pelagatti, Schreiber, 1979), (Wong, Katz, 1983), (Gavish, Segev, 1986), (Yu, *et al.,* 1984, Yu, *et al.,* 1986, Yu, *et al.,* 1987, Yu, *et al.,* 1989) also take advantage of fragmentation to process queries. For example, the algorithm given in (Yu, *et al.,* 1984), called the

Fragment and Replicate Strategy (FRS) algorithm is based on the same principle as that of (Epstein, *et al.,* 1978). However, it takes into account not only the amount of data transferred and processed at individual sites, but also the presence or absence of fast access paths (e.g., indexes) to reduce response time. When all the relations referenced by a query are unfragmented, the FRS algorithm cannot be used. Another strategy is to partition one of the referenced relations into a number of fragments and distribute the fragments to a number of sites so that the query can be processed in parallel at different sites (Yu, *et al.,* 1984). In (Yu, *et al.,* 1984), a Partition and Replicate Strategy (PRS) algorithm is given to determine which relation and which copy of the relation is to be partitioned into fragments, how many fragments are to be produced, and where these fragments are to be sent for processing.

2. REVIEW OF LITERATURES:

Distributed Query Optimization:

Distributed query optimization requires evaluation of a large number of query trees each of which produce the required results of a query. This is primarily due to the presence of large amount of replicated and fragmented data. Hence, the target is to find an optimal solution instead of the best solution.

The main issues for distributed query optimization are −

- Optimal utilization of resources in the distributed system.
- Query trading.
- Reduction of solution space of the query.

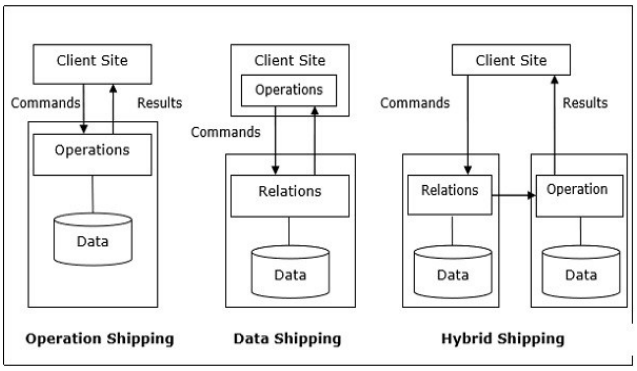
Optimal Utilization of Resources in the Distributed System

A distributed system has a number of database servers in the various sites to perform the operations pertaining to a query. Following are the approaches for optimal resource utilization −

Operation Shipping − in operation shipping, the operation is run at the site where the data is stored and not at the client site. The results are then transferred to the client site. This is appropriate for operations where the operands are available at the same site. Example: Select and Project operations.

Data Shipping − in data shipping, the data fragments are transferred to the database server, where the operations are executed. This is used in operations where the operands are distributed at different sites. This is also appropriate in systems where the communication costs are low, and local processors are much slower than the client server.

Hybrid Shipping − this is a combination of data and operation shipping. Here, data fragments are transferred to the high-speed processors, where the operation runs. The results are then sent to the client site.



Query Trading

In query trading algorithm for distributed database systems, the controlling/client site for a distributed query is called the buyer and the sites where the local queries execute are called sellers. The buyer formulates a number of alternatives for choosing sellers and for reconstructing the global results. The target of the buyer is to achieve the optimal cost.

The algorithm starts with the buyer assigning sub-queries to the seller sites. The optimal plan is created from local optimized query plans proposed by the sellers combined with the communication cost for reconstructing the final result. Once the global optimal plan is formulated, the query is executed.

3. REDUCTION OF SOLUTION SPACE OF THE QUERY

Optimal solution generally involves reduction of solution space so that the cost of query and data transfer is reduced. This can be achieved through a set of heuristic rules, just as heuristics in centralized systems.

Following are some of the rules −

- Perform selection and projection operations as early as possible. This reduces the data flow over communication network.
- Simplify operations on horizontal fragments by eliminating selection conditions which are not relevant to a particular site.
- In case of join and union operations comprising of fragments located in multiple sites, transfer fragmented data to the site where most of the data is present and perform operation there.
- Use semi-join operation to qualify tuples that are to be joined. This reduces the amount of

data transfer which in turn reduces communication cost.

- Merge the common leaves and sub-trees in a distributed query tree.

4. COMPONENTS OF DISTRIBUTED QUERY OPTIMIZATION:

There are three components of distributed query optimization:

Access Method — in most RDBMS products, tables can be accessed in one of two ways: by completely scanning the entire table or by using an index. The best access method to use will always depend upon the circumstances. For example, if 90 percent of the rows in the table are going to be accessed, you would not want to use an index. Scanning all of the rows would actually reduce I/O and overall cost, whereas, when scanning 10 percent of the total rows, an index will usually provide more efficient access. Of course, some products provide additional access methods, such as hashing. Table scans and indexed access, however, can be found in all of the "Big Six" RDBMS products (i.e., DB2, Sybase, Oracle, Informix, Ingres, and Microsoft).

Join Criteria — if more than one table is accessed, the manner in which they are to be joined together must be determined. Usually the DBMS will provide several different methods of joining tables. For example, DB2 provides three different join methods: merge scan join, nested loop join, and hybrid join. The optimizer must consider factors such as the order in which to join the tables and the number of qualifying rows for each join when calculating an optimal access path. In a distributed environment, which site to begin with in joining the tables is also a consideration?

Transmission Costs — if data from multiple sites must be joined to satisfy a single query, then the cost of transmitting the results from intermediate steps needs to be factored into the equation. At times, it may be more cost effective simply to ship entire tables across the network to enable processing to occur at a single site, thereby reducing overall transmission costs. This component of query optimization is an issue only in a distributed environment.

5. CONCLUSION

Distributed structures can be implemented to augment performance. A multi-site, multitable index structure could be created that would contain information on the physical location of tables, as well as the physical location of the data items within that table. This structure, however helpful from a performance perspective, would be difficult to maintain and administer due to its reliance on multiple sites.

REFERENCES:

B. Gavish and A. Segev, 1986. "Set Query Optimization in Distributed Database System," *ACM TODS,* vol. 11:3.

C. Liu and C. Yu, 1993. "Performance Issues in Distributed Query Processing," *IEEE,* vol. 4:8, pp. 889-905.

C. T. Yu, K. C. Guh, C. C. Chang, C. H. Chen, M. Templeton, and D. Brill, 1984. "An Algorithm to Process Queries in Distributed Network," in *IEEE Real-Time Syst. Symp.*.

C. T. Yu, K. C. Guh, W. Zhang, M. Templeton, D. Brill, and A. L. P. Chen, 1986. "Partitioning Relation for Parallel Processing in Fast Local Networks," in *International Conference on Parallel Processing*, pp. 1021-1028.

C. T. Yu, K. C. Guh, W. Zhang, M. Templeton, D. Brill, and A. L. P. Chen, 1987. "An Integrated Algorithm for Distributed Query Processing," in *IFIP Conference on Distributed processing* Amsterdam.

C. T. Yu, K. C. Guh, W. Zhang, M. Templeton, D. Brill, and A. L. P. Chen, 1989. "Partition Strategy for Distributed Query Processing in Local Fast Networks," *IEEE Trans. & Software Engineering,* vol. 15:6, pp. 780-793.

C. Wang and M.-S. Chen, 1994. "On the Complexity of Distributed Query Optimization," *IEEE* vol. 8, pp. 650662.

E. Wong and R. H. Katz, 1983. "Distributing a Database for parallelism," in *ACM SIGMOD* San Jose, CA.

G. Pelagatti and F. A. Schreiber, 1979. "A Model of an access Strategy in Distributed Database System," in *International Conference of Database Architecture*, Venice, Italy.

http://www.tutorialspoint.com/distributed_dbms/distributed_dbms_query_optimization_distributed_systems.htm

J. S. J. Chen and V. O. K. Li, 1989. "Optimizing Joins in Fragmented Database Systems on a Broadcast Local Network," *IEEE Trans. & Software Engineering,* vol. 15:1, pp. 26-38.

L. M. Hass, 1982. "R* : A Research Project on Distributed Relational DBMS," *Database Engineering,* vol. 5:4.

M. Stonebraker and E. Neuhold, 1977. "A Distributed Database Version of INGRES," in *Second Berkley*

Workshop on Distributed data Management & Computer Networks, USA, pp. 19-36.

P. Valduriez and T. Ozsu, 1999."Principle of Distributed Database Systems." Prentice Hall.

R. Epstein, M. Stonebraker, and E. Wong, 1978. "Distributed Query Processing in Relational Database System," in *ACM SIGMOD* Austin, USA.

S. Upadhyaya and S. Lata, 2008. "Task Allocation in Distributed Computing VS Distributed Database Systems: A Comparative Study," *IJCNS (International Journal of Computer Science and Network Security),* vol. 8:3, pp. 338-346.