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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Analysis of Query Using Join and Semi-join

Gurminder Singh Boparai[1], Satish Kumar[2]

Research Scholar, Department of Computer Science and Engineering, G.I.M.E.T, Amritsar, Punjab[1]

Assistant Professor, Department of Computer Science and Engineering, G.I.M.E.T, Amritsar, Punjab[2]

Abstract: Database is the collection of files or table and Database Management System is used to manage the overall activities of the database. We have two approaches for storing and managing database namely Centralised Database management system and Distributed Database management system. Query Processing and its optimization is one of the major key areas of Distributed Database Systems. Query optimization is much more difficult in Distributed Database as compared to the Centralised Database System. This literature review paper is based in the query optimization in Distributed Database System. It mainly focused on to analyse the performance and behaviour of the use of semi-joins and joins. Various factors are considered like CPU cost, I/O cost, Total Cost, Response time and Total time etc. Minimizing the amount of data transmission is important to reduce the query processing cost. To minimize this communication cost, the exhaustive enumerative technique has used to bring dynamism in determining the sub operations (join or semi join) to various sites.

Keywords: OSP, semi join, distributed database, enumerative

I. INTRODUCTION

Distributed Database is defined as the collection of logically interrelated data distributed over several sites. It is developed to meet the organisational structure of distributed enterprises and to develop the efficient techniques for processing complex queries in cost efficient manner. An objective of DDBMSs is to present an easy interface to the users so that they can access the databases as if there were a single database. Another important objective of DDBMS is to process distributed queries efficiently in addition to providing availability and reliability. To process the data located at different locations is distributed data processing. Distributed data processing is needed because of changing business requirements, which have made distributed data processing cost [1].

A distributed database is more popular because it improves system performance, reliability, availability and modularity in distributed system. The data distribution problem and query processing are the critical issues in distributed database. Database system performance is effective depends on join operator. The allocation of operations or sub queries involved in a particular query to the various sites of a network is one of the important components of distributed query processing and query optimization. The query is broken into various sub operations like selection or projections join and semi join and these operations performed at many different sites of network in different sequences. OSP and OAP are the two components of query optimization. OSP requires the optimal sequence of operations for example Join order sequence. OAP requires optimal placement of these operations to different site [1].

Join is the primary target of query optimizer because of the high evaluation costs. Many algorithms have been proposed for the exploitation of join operations in distributed database. The important task of these algorithms is to reduce the size of data transmitted through the communication network. Sending all the relations to one site and executing join there is one straightforward approach but due to high transmission cost it is not good. Significant research effort have been made in order to reduce cost, there are two approaches join sequence and semi join strategies.

II. DISTRIBUTED QUERY PROCESSING

Distributed query processing is retrieval of data from different sites in a distributed database Query processing is much more difficult in distributed environment than in centralized environment because a 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. This reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. The fragments that accessed by queries are needed to be allocated to the DDBs sites so as to reduce the communication cost during the applications execution and handle their operational processing The main objective of query processing is to transform the high level query (relational calculus) into efficient execution strategy expressed in equivalent low level query (into relational algebra+ communication operators) on local databases. It must achieve both correctness and efficiency. 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
- 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. semi joins 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.

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

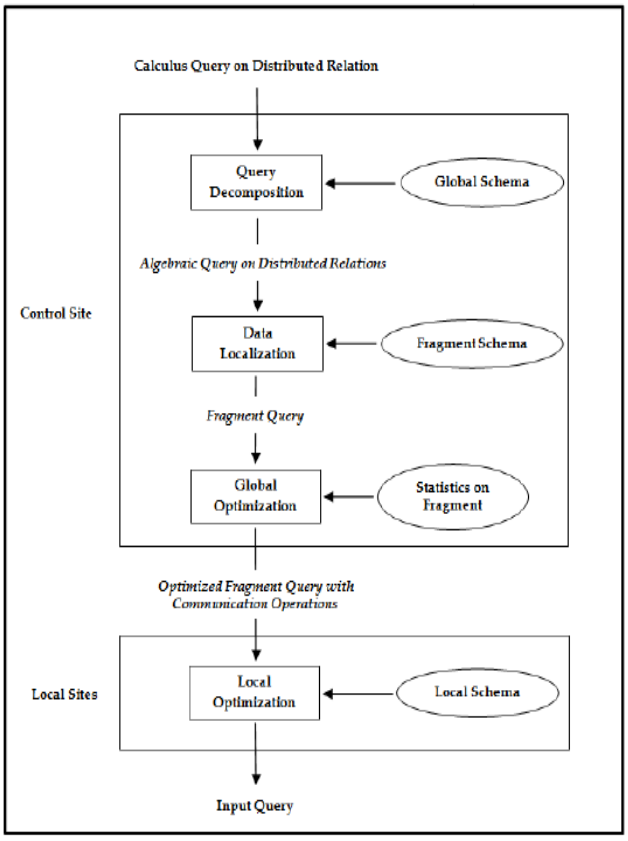

Fig 1: Distributed Query Optimization Model [3]

III. RELATED WORK

Lin Zhou, et al (2012) describes analysis of the query optimization process based on semi join operation combined with practical operation. Here semi joins operations are used to improve time response performance of a query and reduce communication cost. The goal of query optimization operations is to reduce the volume of data and the cost for process. In this paper, the connection process is expressed by semi joins and then cost estimation is done. They compare the cost and results between two different semi joins operation methods and prove that the query methods are can affect the execution speed of the system directly. [4]

Manik Sharma, et al (2012) focus is given on computing and analyzing the performance of joins and semi joins in distributed database system. The various metrics that are considered like query cost, memory used, CPU cost, I/O cost, bytes transferred from one site to another, total time and response time. In this paper, a experimental analysis is done in which query is implement and execute using three different joins and using semi join. They conclude that, In case of data transmission, semi join give is more useful than join. It reduces the amount of data transferred. In regard to cost, the use of semi join is beneficial if the cost to produce and send it to other site is less than the cost of sending the whole operand relation and of doing the actual join. In regard to total time, the query executed with semi join possess lesser total time when data transfer is used. Joins gives its best in data transmission when a relation having lower cardinality is transmitted to the location where relation of upper cardinality and larger tuple size is placed. Further one is able to conclude that semi joins are beneficial if the transmission cost is main consideration otherwise joins will be preferred. [5]

Xiao feng Li, Dong Li, et al (2010) is based on study of some common optimization algorithm based on multi relation semi join is put forward to apply to this situation that takes buffer zone of distributed database system as the final assembly station of intermediate result query. The experiment proves that query that query optimization algorithm based on multi relation semi join reduces the data volume of intermediate result and effectively decreases the overall cost of network communications. [6]

Pawandeep Kaur, et al (2013) presents the join query optimization in Distributed databases. One method for the join query is first to transfer the data from servers to client site and then insert the data into client database, after that join query is performed. Proposed method will directly perform the join query on the client site after fetching from servers site and it will not insert the data into client database. By the proposed method, insertion time of data into client database will be deducted. So, this method will optimize the join query in distributed databases. [7]

Sunita M. Mahan, Vaishali P. jadhav (2013) "tri-variate optimization strategies of semi join technique on distributed databases" (2013) mentioned the use of semi join operation. Beneficial semi join operation reduces the amount of data transmission required to perform the join sequences. The problem of finding an optimal strategy to minimize data transmission cost in distributed database systems, even with one join attribute is problem determining the optimal sequence of join operations in query optimization leads to exponential complexity. To deal with this problem, there is a need of develop a heuristic approach to solve the problem. [8]

Manik Sharma, et al (2013) focus is on design and analysis of DSS queries. The selected set of DSS queries are simulated by using exhaustive enumerative technique and genetic approach under serial and parallel processing environment. The simulation results show that an exhaustive enumeration approach provides optimized solution but takes huge time for complex DSS queries (Hours, Days, Month or Even Years), hence it is infeasible to implement this approach for optimizing a set of DSS queries. On the other hand genetic algorithms optimize DSS Queries very quickly but show loss in accuracy and quality of solution as compare to exhaustive enumerative approach. Further the parallel execution of the different sub operations of a DSS query significantly reduces the total cost of system resources. [9]

Abhijeet Raipurkar, G.R. Bamnote (2013) focus is given on Query processing in a distributed system requires the transmission of data between computers in a network. Two cost measures, response time and total time are used to judge the quality of a distribution strategy. They presented 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

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

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. [10]

In this simulator, a distributed query is broken down into various sub operations like Selections, Projection or Joins/Semi-joins etc. to be performed in a particular order[11]. Then these ordered sub-query operations may be performed at many different sites of the network in many different permutation & combinations of the sites of existing computer network. Experiments were conducted after running the simulator enumerative on an experimental set of Wisconsin benchmark Database Queries. Exhaustive enumerative approach (for sub-query operation allocation) is coded in PASCAL programming Language. The total cost of a distributed query is composed of I/O cost, CPU cost and communication cost. A detailed analysis of the results of one of the query is explained in the following section. Structure of Wisconsin Database Tables (Bi) [12], communication and I/O coefficient's matrices are shown below.

Cardinality: 10,000, Tuple Size: 40bytes, Table Size: 100 Kilobytes, Block Size: 4Kb, Table Size: 100 Blocks

TABLE I: Table as per Relations Bi's Design

Unique (0 – 9999)	Twos (0 – 1)	Tens (0-10)	Hundreds (0 – 99)	FiveHundreds (0 – 499)	Thousands (0 – 999)
7	0	1	11	4	999
111	1	4	2	3	4
9998	1	9	4	444	111
777	0	3	98	499	45
10,000 Tuples ..	.....	.....	.....	.....	.....etc

TABLE II: Relation Bi's Statistics

Relations Bi	Cardinality	Tuple Size (bytes)	Relation Size (Kilobytes)	Relation Size (Blocks) 1Block = 4 kb
B1-B7	10,000	40	400	100

In this paper, communication coefficients matrices, I/O Coefficient's matrices, number of blocks and number of sites have taken. The below tables show the different matrices we have taken in our experiment. Table III show the communication between the six sites, I/O and the CPU coefficients. Table IV represents the allocation of the data to which site. Table V represents the intermediate fragments used in various operations.

TABLE III: CPU,I/O, Communication Coefficients

Communication Co-efficient	S1	S2	S3
S1	0	2	2
S2	2	0	3
S3	3	2	0
IO Coefficients	1	1.1	1.2
CPU Coefficients	1.1	1	1

TABLE IV: Data Allocation Matrix

Data Allocation coefficients's A_{ij}	S1	S2	S3
B1	1	1	0
B2	0	0	0
B3	0	0	1

IV. EXPERIMENTAL QUERY

The Query we considered for our experiment operates on 4 base relations. There are 12 operations and the fragments generated in solving the Query are 15. The query tree which represents the execution of the experimental query is shown in figure 2. The different no. of operations and fragments gives the ides of the execution of the query we have taken

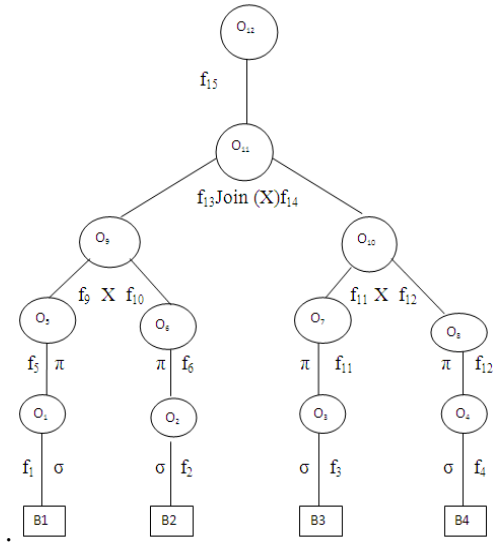

Fig 2: Query Tree

A. *The Query Tree Description*

No. Of Operations: 12(O_1 – O_{12})
(Tree Nodes)
No. Of Selections: 4 (O_1 – O_4)
No. Of Projections: 4 (O_5 – O8)
No. Of Joins: 3 (O_9 – O_{11})
Final Operation 1 (O_{12})
Send the query result to the query originating site
No. Of Intermediate Fragments: 15 (Tree Edges, Including 7 Base Relations)

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

TABLE V: Fragment Operation Matrix

sub queries→ / ↓fragments	SELECTIONS & PROJECTIONS								JOINS			
	1	2	3	4	5	6	7	8	9	10	11	12
f1	1	0	0	0	0	0	0	0	0	0	0	0
f2	0	1	0	0	0	0	0	0	0	0	0	0
f3	0	0	1	0	0	0	0	0	0	0	0	0
f4	0	0	0	1	0	0	0	0	0	0	0	0
f5	0	0	0	0	1	0	0	0	0	0	0	0
f6	0	0	0	0	0	1	0	0	0	0	0	0
f7	0	0	0	0	0	0	1	0	0	0	0	0
f8	0	0	0	0	0	0	0	1	0	0	0	0
f9	0	0	0	0	0	0	0	0	1	0	0	0
f10	0	0	0	0	0	0	0	0	1	0	0	0
f11	0	0	0	0	0	0	0	0	0	1	0	0
f12	0	0	0	0	0	0	0	0	0	1	0	0
f13	0	0	0	0	0	0	0	0	0	0	1	0
f14	0	0	0	0	0	0	0	0	0	0	1	0
f15	0	0	0	0	0	0	0	0	0	0	0	0
f16	0	0	0	0	0	0	0	0	0	0	0	1

V. ANALYSIS AND DISCUSSION OF ANALYTICAL RESULTS

In this section various experiments using join are analysed and observations are made to study the effect of fluctuations on the exhaustive enumerative solution. It has been shown earlier that the total cost consists of CPU, I/O and communication cost. The objective of this section is to study the variations of cost in different scenarios. The results are shown below:

A. Scenario:

The size of each fragment, called size (Fj), must be defined since it plays a major role when computing the communication cost. In this scenario, fragment size is considered as parameter. The fragment size in case of join [13] is more than in case of semi joins

B. Estimated Intermediate fragment sizes with Semi Joins:

Operation 1: $(\sigma)\ B_1 \rightarrow f_5$; Size: 100 x 0.7($\rho_s$) = 70 blocks

Operation 2: $(\sigma)\ B_2 \rightarrow f_6$; Size: 100 x 0.7($\rho_s$) = 70 blocks

Operation 3: $(\sigma)\ B_3 \rightarrow f_7$; Size: 100 x 0.7($\rho_s$) = 70 blocks

Operation 4: $(\sigma)\ B_4 \rightarrow f_8$; Size: 100 x 0.7($\rho_s$) = 70 blocks

Operation 5: $\pi_{Unique}\ (f_5) \rightarrow f_9$; Size: 70 x 0.9 ($\rho_p$) = 63 blocks

Operation 6: $\pi_{Unique}\ (f_6) \rightarrow f_{10}$; Size: 70 x 0.9($\rho_p$)= 63 blocks

Operation 7: $\pi_{Unique}\ (f_7) \rightarrow f_{11}$; Size: 70 x 0.9($\rho_p$) = 63 blocks

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

Operation 8: π_{Unique} (f_8) → f_{12}; Size: 70 x 0.9(ρ_p) = 63 blocks

Operation 9: (f_9⋮ X ⋮f_{10}) → f_{13}; Size: 63 x 0.2(ρ_j) =13 blocks

Operation 10: (f_{11}⋮ X ⋮f_{12}) →f_{14}; Size: 63 x 0.2(ρ_j) = 13 blocks

Operation 11: (f_{13}⋮ X ⋮ f_{14}) → f_{15}; Size: 50 x 0.2(ρ_j) = 10 blocks

Operation 12: f_{15} →Final Result to Query Site.

C. Estimated intermediate fragment sizes with Semi Joins:

Operation 1: (σ) B_1 →f_5 ; Size: 100 x 0.7(ρ_s) = 70 blocks

Operation 2: (σ) B_2 →f_6 ; Size: 100 x 0.7(ρ_s) = 70 blocks

Operation 3: (σ) B_3 →f_7 ; Size: 100 x 0.7(ρ_s) = 70 blocks

Operation 4: (σ) B_4 → f_8 ; Size: 100 x 0.7(ρ_s) = 70 blocks

Operation 5: π_{Unique} (f_5) → f_9; Size: 70 x 0.9 (ρ_p) = 63 blocks

Operation 6: π_{Unique} (f_6) → f_{10}; Size: 70 x 0.9(ρ_p)= 63 blocks

Operation 7: π_{Unique} (f_7) → f_{11}; Size: 70 x 0.9(ρ_p) = 63 blocks

Operation 8: π_{Unique} (f_8)→ f_{12}; Size: 70 x 0.9(ρ_p) = 63 blocks

Operation 9: (f_9⋮ X ⋮f_{10}) → f_{13}; Size: 63 x 0.1(ρ_j) = 6 blocks

Operation 10: (f_{11}⋮ X ⋮f_{12}) →f_{14}; Size: 63 x 0.1(ρ_j) = 6 blocks

Operation 11: (f_{13}⋮ X ⋮ f_{14}) → f_{15}; Size: 50 x 0.1(ρ_j) = 5 blocks

Operation 12: f_{15} →Final Result to Query Site.

- When fragment size in case of join is [70 70 70 70 63 63 63 63 13 13 10]. The cost calculated in this case is 360. When fragment size in case of semi join is [70 70 70 70 63 63 63 63 13 13 5]. The cost calculated in this case is 288.

- When fragment size in case of join is [70 70 70 70 63 63 63 63 13 13 10]. The cost calculated in this case is 360. When fragment size in case of semi join is [70 70 70 70 63 63 63 63 13 13 5]. The cost calculated in this case is 288.

- When fragment size in case of join is [70 70 70 70 63 63 63 63 13 13 10]. The cost calculated in this case is 360. When fragment size in case of semi join is [70 70 70 70 63 63 63 63 13 13 5]. The cost calculated in this case is 288.

- When fragment size in case of join is [70 70 70 70 63 63 63 63 13 13 10]. The cost Calculated in this case is 360. When fragment size in case of semi join is [70 70 70 70 63 63 63 63 13 13 5]. The cost calculated in this case is 288.

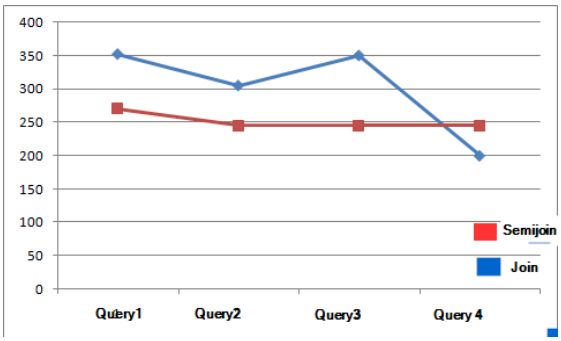

Fig 3: Result

The cost calculated in case of join is more than in case of semi joins. As the fragment size increases, total cost goes on increasing.

VI. CONCLUSION

The comparative analysis of the Query Processing using joins and semi joins in Distributed Database Management System is discussed. The use of joins and semi joins affects the cost of Query Processing in Distributed DBMS and the communication cost can be reduced using joins and semi joins.

REFERENCES

[1] M. Tamer Ozsu, Patrick Valduriez, "Principles of Distributed Database Systems", Third Edition, Springer, 2011

[2] B.M. Monjurul Alom, Frans Henskens and Michael Hannaford "Query Processing and Optimization in Distributed Database Systems" IJARCCE 2009

[3] Tewari, Preeti. "Query Optimization Strategies in Distributed Databases."International Journal of Advances in Engineering Sciences 3.3 (2013): 23-29.

[4] Lin Zhou ,Yan Chen,Taoying Li ,Yingying Yu "The semi join query optimization in distributed database system" CITCS 2012

[5] Manik Sharma, Dr. Gurvinder singh, Rajinder Virk "Analysis of joins and semi joins in a distributed database query" published in preceding of International journal of computer application (2012)

[6] Xiaofeng Li,Dong Li,Hong Zhi Gao,Lu Yao " study of query of distributed database based on relation semi join" ICCDA 2010

[7] Pawandeep Kaur, Jaspreet Kaur Sahiwal "join query optimization in distributed databases" Published in the preceding of International journal of scientific and research publication 2013

[8] Sunita M. Mahan, Vaishali P. jadhav "tri-variate optimization strategies of semi join technique on distributed databases" International journal of computer applications (2013) .

[9] Manik Sharma, Dr. Gurvinder singh, Rajinder Virk "Design and Comparative Analysis of DSS Queries in Distributed Environment" IEEE (2013)

[10] Abhijeet Raipurkar, G.R. Bamnote " Query processing in distributed Database Through Data distribution" IJARCCE 2013

[11] Ismail O. Hababeh, "A Method for Fragment Allocation Design in the Distributed Database Systems", The Sixth Annual U.A.E. University Research Conference, U.K.

[12] Kahlon, K.S. and Singh, Arjun, "Non-Replicated Dynamic Data Allocation in Distributed Databases", in International Journal of Computer Science and Security, Vol. 9, Sept., 2009, pp. 176-180.

[13] Chen, Ming-Syan, and Philip S. Yu. "Using join operations as reducers in distributed query processing." Proceedings of the second international symposium on Databases in parallel and distributed systems. ACM, 1990.