International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 16 (2014), pp. 1653-1659

http://www. irphouse.com

Dynamic Estimation of Intermediate Fragment Size in a Distributed Database Query

Parnika bhat[1], Rajinder Singh Virk Student[2]

[1,2]Department of Computer Science
Guru Nanak Dev University

Abstract

Distributed database is a collection of databases that are distributed over various nodes and are logically interconnected by a communication network. In distributed database system the objective of Query Optimization is to execute the query in minimum time. That is the total cost of processing a query must be minimum, for efficient working of distributed database. Total cost involves I/O cost, CPU cost, Communication cost. Distributed Query Optimization divides the relations in to fragments and allocates it to different nodes. Query is divided into sub queries and then sub query allocation is performed optimally. Cost depends on the size of intermediate fragments, reallocation of fragments and communication speed in transferring fragments from one node to another. Query optimization can estimate the size of intermediate fragments in two ways: static and dynamic. In static case fixed or predetermined set of values are given as input file and in case of dynamic query optimization it's the code that generate the size of intermediate fragments. In the proposed work i will be using an existing simulator DQO (Dynamic Query Optimizer) which stochastically optimize process of subquery allocation to different nodes of distributed database. In this work a new component will be augmented that will estimate the size of intermediate fragments dynamically.

Index Terms— Distributed Databases, Query Optimization, Fragmentation, Genetic Algorithms, Total Cost, Communication Cost

INTRODUCTION

A database is an organized collection of data. This data is managed by special software known as Database Management System (DBMS). DBMS is responsible for querying, updating, defining the data. Distributed Database Systems have been

developed to meet the increase in amount of data and distributed nature of organizations in distributed enterprises. A distributed database is a collection of databases that can be stored at different computer network sites. It is under the control of a central database management system (DBMS).

In distributed databases query processing is of important concern. Data retrieval from different sites in a distributed database is called distributed query processing. 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.

In distributed databases as databases are located at geographically different locations, so a simple query that needs to access databases from various locations can be decomposed into sub queries. Sequencing of those sub queries is an important issue in distributed databases. Sequencing of sub queries should be done in such a way that there is minimum operating cost for processing that query. That is there is need to optimize the query. So query optimization comes into picture. Query optimization is to determine the most efficient way to execute a given query by considering the possible query plans. A query plan is an ordered set of steps used to access data from database.

Distributed Database Query Optimization deals with designing the data distribution (allocation and fragmentation); query Processing and analyzing the algorithms with a goal to achieve minimum total cost. The cost of a distributed execution strategy can be expressed with respect to either the total time or the response time. In query optimization a state space is searched which constitutes all the access plans that compute the same result to a query. Every access plan is associated with a certain cost, which is given by a cost function. That access plan is chosen which is having minimum cost associated with it. Query optimization is an NP hard problem. [5]

For query optimization there are various algorithms. Broadly they are classified into:

- Deterministic Search Algorithms
- Genetic Algorithms
- Randomized Algorithms

Genetic algorithms generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Genetic algorithms propagate their best solutions for a given problem from generation to generation, improving them further and further [14]

This algorithm can be easily adapted to query optimization problem where character strings can be the query execution plans and selection, crossover, and mutation are applied on those execution plans. In this the fittest solution will be that has a minimum execution cost associated with it. [2]As it is known that the problem of finding an optimal plan is NP hard. So a randomized technique can also be used to create an optimal plan. In a randomized algorithm random choices are made to make a selection from state space. It moves from state to state with the goal of finding a state

with the minimum cost. It is simple to implement and faster than deterministic algorithm. But its disadvantage is it might not find the correct answer at all or it may take a very long time to find the correct answer.

The efficiency of a distributed database design depends on cost, response time, and availability.[3] The cost here include: data storage costs, network communication costs which include cost of retrieval and update messages, data transmission cost for intermediate processing, and final responses, and local data processing costs. A Query execution plan selected for processing the query must minimize the Query Processing Costs. The Query Processing Cost consists of Local Processing Cost and Communication Cost. In this work an effort shall be made to minimize the Local processing Cost as well as Communication cost.

II. DISTRIBUTED COST MODEL

The total time is the sum of all time components, while the response time is the elapsed time from the initiation to the completion of the query. A general formula for determining the total time can be specified as follows [2]:

Total_time = T_{CPU}* #insts + $T_{I/O}$*#I/Os + T_{MSG}*#msgs + T_{TR}*#bytes

The first two components measure the local processing time, where T_{CPU} is the time of a CPU instruction and $T_{I/O}$ is the time of a disk I/O. There are well established mathematical model to calculate local processing cost and communication cost.

Local Processing Costs

$$LPC_y^q = \sum_s S_{yS}^q (\text{IOC}_s \sum_r I_{ry}^q M_{ry}^q + \text{CPC}_s \sum_r I_{ry}^q M_{ry}^q) \quad \text{(a)}$$

Where M_{ry}^q = No. of memory blocks of relations 'r' accessed by sub query 'y' of q. IOC_s = Input Output Cost Coefficient of site s in milisec per 8k bytes CPC_s= CPU Cost coefficient of site s. So equation (a) represents local processing. Local processing costs for a join may be given as

$$LPC_y^q = \sum_s S_{yS}^q (IOC_s \sum_p \sum_r I_{ryv[p]}^q M_{ryv[p]}^q + \sum_s S_{yS}^q (IOC_s \prod_r I_{ry}^q M_{ry}^q + CP \prod_r I_{ry}^q M_{ry}^q)$$

Where 'ρ_p' 'Selectivity Factor' & is defined as the ratio of resultant different values of a field to the domain of that field ($0 <= \rho_p <= 1$). $M_{ryv[p]}^q$ is the size of an intermediate relation. $v_{[p]}$ represents 'left previous operation' of a join for p=1 & 'right previous operation' of a join for p=2.

Communication Costs

$$COMM_y^q = \sum_p \sum_s \sum_v S_{yv[p]S}^q S_{yv}^q C_{sv} \left(\sum_r I_{ryv[p]}^q M_{ryv[p]}^q \right) \quad \text{(b)}$$

Where C_{sv} (is the communication cost coefficient between site s and v)

C_{sv} = 0 if (s = v) (i.e. if the previous operations and current join operation is done at the same site) [8]

III. RELATED WORK

(Areerat et al, 2009) In this paper authors had proposed a new join order algorithm called Exhaustive Greedy (EG) algorithm to optimize intermediate result sizes of join queries. Exhaustive search and greedy algorithm are combined and modified to identify good join orders. Exhaustive search algorithm can guarantee the optimal solution as it produces the entire join trees of a join graph. Greedy algorithm used to reduce search space by generating only one query tree in a polynomial time. In order to determine join order selection at subsequent steps, this EG algorithm also updates join graphs to reflect new size of join nodes and new join selectivities of edges associated with the join nodes at each step. In addition, most intermediate result sizes of join queries estimated by the EG algorithm are comparable to the results estimated by the Exhaustive Search algorithm (ESU)that is modified to update join graphs, it is named as ESU Algorithm. Exhaustive algorithm is performed at the beginning to find all the edges in the original join graph then each edge is set as a starting route of the each candidate route. The remaining join orders of each candidate route then are performed by greedy algorithm.

(Ali A. Amer and Hassan I. Abdalla, 2012) In case of distributed database system, fragmentation, replication and allocation are three main processes that affect its performance. A database can be fragmented horizontally, vertically or in both ways. Optimal allocation of fragments leads to optimal solution in case of dynamic distributed environment. If the fragments are allocated across the different nodes in such a way that communication cost is reduced it will result in efficient working of distributed database systems. The authors have proposed an efficient model for data re-allocation by changing the data access pattern over different nodes in DDBMS. In this model a set of prefixed query frequency values are used for the distribution of fragments over different sites. Data fragments are re-allocated based on two information communication costs between sites and update cost values for each fragment. The re-allocation of fragments is done depending on the maximal update cost value for each fragment. Fragment priority (FP) procedure is used for allocation of fragments to an anticipated site which avoid replication of fragments thereby reducing the computational complexity. This model provides an effective solution for dynamic fragments re-allocation problem in case of a distributed relational database systems by reducing the frequency of fragment transmission from one node to other over the network, hence improving the overall performance of DDBS.

(Shahidul Islam Khan and Dr. A. S. M. *et al*, 2010) In distributed database systems there are three processes by which data is distributed among various sites, these are: fragmentation, allocation, and replication. The reliability and performance of a database system can be improved by effective data processing. Fragmentation process requires empirical knowledge of type of queries submitted to the centralized system and their frequencies. For the initial stage of a database design this fragmentation process is not suitable. In this paper the author had proposed a

horizontal fragmentation technique is capable of taking proper fragmentation decision at the initial stage by using the knowledge gathered during requirement analysis phase without the help of empirical data about query execution. It allocates the fragments properly among the sites of DDBMS. As fragmentation is done synchronously with allocation there are no added complexities for allocating fragments to different sites. Hence, by avoiding frequent remote access and high data transfer among the sites, DDBMS can be improved significantly.

(Syam Menon, 2005) A distributed database system comprises of number of databases (or fragments of databases) stored at multiple sites. All these databases work together in a way that users don't realize the presence of multiple databases; it appears as a single large database. Database is partitioned horizontally or vertically (or both) to obtain database fragments. For any distributed database system to work well, these fragments have to be dispersed over the available sites in such a way as to minimize the total volume of data transmitted and, consequently, the total cost of transmissions. In a distributed database system the fragments need to be allocated judiciously at various sites across the communications network. It is difficult to allocate fragments to most appropriate sites. In this paper the problem of fragment allocation is solved by new integer programming formulations. This approach handles the problem with storage and processing capacity constraints. In the presence of capacity restrictions this approach gives effective solution. The cost of local processing is assumed to be negligible relative to the cost of communication. Even for relatively large problems, reformulations are very effective, proved by experiments conducted over different parameter values. The use of this programming formulation has reduced the use of heuristic technique. Effective distribution of the database fragments, affect both performance and cost. In this using new formulations, for the problem of the fragments are allocated at minimum cost. Even in case of allocating reasonably large number of fragments the formulations are seen to be extremely effective as compared to previous formulations. But the limiting factor of this paper is that none of the approaches presented have been implemented and tested on a real distributed database system.

(Lisbeth Rodríguez and Xiaoou Li, 2011) To improve the response time of query in distributed database two processes are required, these are: Vertical and Horizontal partitioning of distributed database. In current database management system horizontal partitioning has received strong attention than vertical partitioning. Efficient vertical partitioning solution requires monitoring of user queries. In this paper authors had developed a system called DYVEP (DYnamic VErtical Partitioning) that dynamically partitions the distributed database into vertical fragments. Without any intervention from DBA it fragments and re-fragments a database. To demonstrate acceptable query response time, a Benchmark database TPC-H is used to carry out experiments. Vertical Partitioning algorithm is used by DYVEP to determine whether the vertical partitioning scheme (VPS) is better that the one in place. Results of experiments shows this system adaptively perform vertical partitioning within efficient query response time. In the future, the results can be extended to Multimedia database systems, to clearly observe the effect of proposed system.

(B.M. Monjurul Alom, Frans Henskens and Michael Hannaford, 2009) In query processing in distributed systems the main problem is determining the sequence and the sites for performing the set of operations if the query is subdivided into sub queries that require operations at geographically distributed databases, such that the operating cost for processing the query is minimized. For this authors had proposed a technique to process the query with minimum intersite data transfer. The proposed technique is used to determine which relations are to be partitioned into fragments, and where the fragments are to be sent for processing. The technique generally fragments the relations that exist in the predicates (the WHERE condition) of the query. It chooses more than one relation to remain fragmented which exploits parallelism, while replicating the other relations (excluding the fragmented relations) to the sites of the fragmented relations. Thus the communication costs and local processing costs can be reduced due to the reduced size of the fragmented relations and the response time of queries can be improved.

IV. CONCLUSIONS AND FUTURE SCOPE

Communication cost is one of the major concerns in distributed system. Previously research has been conducted in area of optimal allocation of data to appropriate sites to reduce communication cost and improve performance in distributed systems. In this work effort shall be made to estimate the size of intermediate fragments dynamically. For this purpose a stochastic stimulator will be used for optimal allocation of fragments in a distributed environment, where this optimality depends on knowing communication cost in transferring data fragments from one site to the other. Genetic Algorithm technique will be used, as GA helps to reach best solutions much faster. In future, the work can be extended to multimedia database system. As the multimedia database is a dynamic system, so the advantages of estimating the intermediate fragments dynamically would be understood much clearly.

V. REFERENCES

[1] Sangkyl Rho, Salvatore T. March, "A Nested Genetic Algorithm for Distributed Database Design", IEEE, 1994

[2] Salvatore T. March and Sangkyu Rho, "Allocating Data and Operations to Nodes in Distributed Database Design", 1995

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

[4] Shahidul Islam Khan and Dr. A. S. M. Latiful Hoque, "A New Technique for Database fragmentation in Distributed Systems", 2010,

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

[6] Rho Sangkyu, T. March Salvatore, "A Comparison of Distributed Database Design Models", Seoul Journal of Business Vol. 8 No. 1, 2002

[7] Areerat *et al*, "Exhaustive Greedy Algorithm for Optimizing Intermediate

Result Sizes of Join Queries", IEEE, 2009
[8] Lisbeth Rodríguez and Xiaoou Li, "A Dynamic Vertical Partitioning Approach for Distributed Database System", IEEE, 2011
[9] G.R.Bamnote and Himanshu Joshi, "Distributed Database: A Survey" International Journal Of Computer Science And Applications, 2013
[10] Ali A. Amer and Hassan I. Abdalla, "A Heuristic Approach to Re-Allocate Data Fragments in DDBSs", International Conference on Information Technology and e-Services, IEEE, 2012
[11] Azzam Sleit, "A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems", American Journal of Applied Sciences, 2007
[12] Syam Menon, (2005) "Allocating Fragments in Distributed Databases" Transactions on parallel and distributed systems, IEEE, 2005
[13] Sangkyl Rho, Salvatore T. March, (1994) "A Nested Genetic Algorithm for Distributed Database Design", IEEE