International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 6, June - 2013

Cost Estimates & Optimization of Queries Distributed Databases

Ridhi Kapoor
Department of Computer Science & Engineering, Guru Nanak Dev University Amritsar, Punjab, India

Abstract--Query optimizers are critical to the efficiency of modern relational database systems. If a query optimizer chooses a poor query execution plan, the performance of the database system in answering the query can be very poor. This paper describes the focus given on computing and analyzing the performance of joins, semi joins and Cartesian product via comparative analysis of Processing cost, Communication cost and Total cost in distributed database system. A component in the database management system called the Query Optimizer decides how to pick an efficient execution plan. For this the optimizer deploys cost-based optimization. Approximate execution costs are calculated for various plans, and one with low cost is chosen. The execution cost is a weighted function of the system resources needed to execute the query. Examples of such system resources are the CPU time or the number of I/O operations. In order to come up with reasonable cost estimates, the optimizer needs to estimate the size of sub-queries. This is important, for instance, when choosing the join order of the relations. To estimate the sizes of sub-queries, the optimizer needs to know the selectivity of the query predicates.

Index Terms **—Cost Based Query Optimizers, Distributed Databases, Query Optimization, Response Time, Selectivity, Total Time.**

I. INTRODUCTION

A DDB query is answered by joining tables [8]. In a distributed database, tables reside on different nodes of a computer network; to join tables, data must be moved between nodes. Consequently, the cost of a distributed query includes a processing cost (the joins) and a transmission/communication cost [1]. In a query, the order of joins is not specified. Distributed query optimization involves finding an efficient order for the required joins. Before moving a large table across the network, it may be possible to reduce its size by restricting it to just those rows that are related to the table to which it will be joined. However, to effect this reduction, another table must be sent across the network and an additional join performed. As the number of tables in a query increases, the number of possible join schedules grows at least exponentially; an exhaustive search for the minimum cost schedule is not feasible. Optimizer needs the selectivity of a query, i.e., the number of records that qualifies to a query, in order to generate an efficient query execution plan. The query optimizer can generate several execution plans for the same query. To choose the execution plan having the response time close to the optimal, the optimizer is based on a cost model.

Distributed Database [2]: - A database that consists of two or more data files located at different sites on a computer network.

Query consists of operations on tables. Most commonly performed operations are

Select (σ): Returns tuples that satisfy a given predicate

Project (π): Returns attributes listed

Join (⋈): Returns a filtered cross product of its arguments

Set operations: Union, Intersect, and Difference

Query Processing:-It is defined as the activities involved in parsing, validating, optimizing and executing a query. The main aim of query processing is to transform a query written in high level language(eg.SQL) into efficient and correct strategy expressed in low level language(implementing low-level language).

High level user query -> Query Processor ->low-level data manipulation commands.

Query Optimization:-It refers to the process by which the best execution strategy for a given query is found from a set of alternatives. Query optimization is a part of query processing. The main aims of query optimization are to choose a transformation that minimizes resource usage, reduce total execution time of query and also reduce response time of query.

Cost Based Query Optimization:-It assigns an estimated "cost" to each possible query plan, and chooses the plan with the smallest cost [1]. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations and CPU requirements, and other factors determined from the dictionary. The search space can become quite large depending on the complexity of the SQL query.

II. DISTRIBUTED COST MODEL

One of the hardest problems in query optimization is to accurately estimate the costs of alternative query plans. Optimizers cost query plans using a mathematical model of query execution costs that relies heavily on estimates of the cardinality, or number of tuples, flowing through each edge in a query plan [3]. Cardinality estimation in turn depends on estimates of the Selection factor of predicates in the query. An optimizer cost model includes cost functions to predict the cost of operators, and formulas to evaluate the sizes of results.

Cost function (in terms of time)→ I/O cost + CPU cost + Communication cost [7].

- **Total Time/Cost→** It is the sum of all times/cost.

Generally—

Total cost = CPU cost + I/O cost + communication cost

CPU cost = unit instruction cost ∗ no. of instructions

I/O cost = unit disk I/O cost ∗ no. of disk I/Os

Communication cost = message initiation + transmission

$$Tcost(queries) = Ncost + Ccost$$

Where, Tcost is Total cost, Ncost is Network Processing Cost and Ccost is the Communication Cost in Distributed Processing environment.

- Ncost (Processing Cost) can be calculated as:-

Ncost= (local*localc) + (remote1*remote1c) + (remote2*remote2c) + (remote3*remote3c) + (remote4*remote4c) +…. + (remoteN*remoteNc).
Where localc, remote1c, remote2c, remote3c… remoteNc is the processing cost of local, remote1, remote2, remote3 …remoteN site resp.

- Ccost (Communication Cost) can be calculated as:-

Ccost=(local*clocal)+(remote1*cremote1c)+(remote2*cremote2c)+(remote3*cremote3c)+(remote4*cremote4c)+…. + (remoteN*cremoteNc).
Where clocal, cremote1c, cremote2c, cremote3c… cremoteNc is the communication cost of local, remote1, remote2, remote3 …remoteN site resp.

- **Response Time** →Elapsed time between the initiation and the completion of a query

Response time = CPU time + I/O time + communication time
CPU time = unit instruction time *no. of sequential instructions
I/O time = unit I/O time *no. of sequential I/Os
Communication time [5] = unit msgs initiation time * no. of sequential msgs + unit transmission time * no. of sequential bytes.

III. Selectivity

The selectivity factor is defined as the ratio of number of rows of result to the cardinality of the base table.

Selectivity factor of each operation for relations

For joins

Join Selectivity Factor is defined as the ratio of the number of rows participating in the Join to the total number of rows in the Cartesian product of relations.

$$SF(R, S) = card(R\ S)/card(R) * card(S)$$

For semi joins

Fraction of R-tuples that join with S-tuples. An approximation is the selectivity of A in S

$$SF_{\triangleright<}(R \triangleright< A\ S) = SF_{\triangleright<}(S.A) = card(\pi A(S))/card(dom[A])$$

For Cartesian product

It is also called CROSS PRODUCT or CROSS JOIN. It combines the tuples of one relation with all the tuples of the other relation.

$$Card(R \times S) = card(R) * card(S)$$

IV. Related Work

Doshi, Pankti, and Vijay Raisinghani (Advances in Computing, Communication, and Control. Springer Berlin Heidelberg, 2013. 1-13)"k-QTPT: A Dynamic Query Optimization Approach for Autonomous Distributed Database Systems." has discussed Query processing in a distributed database system requires the transmission of data between sites using communication networks. Distributed query processing is an important factor in the overall performance of a distributed database system. In distributed query optimization, complexity and cost increases with increasing number of relations in the query. Cost is the sum of local cost (I/O cost and CPU cost at each site) and the cost of transferring data between sites. Mariposa, Query Trading (QT) and Query Trading with Processing Task Trading (QTPT) are the query processing algorithms developed for autonomous distributed database systems. However, they incur high optimization cost due to involvement of all nodes in generating optimal plan. We present our solution *k*-QTPT, to reduce the high optimization cost incurred by QTPT. In *k*-QTPT, only *k* nodes participate in generating optimal plans.

Golshanara, Ladan, Seyed Mohammad Taghi Rouhani Rankoohi, and Hamed Shah-Hosseini(Knowledge and Information Systems (2013): 1-32)"A multi-colony ant algorithm for optimizing join queries in distributed database systems."In this paper, for the first time, a multi-colony ant algorithm is proposed for optimizing join queries in a distributed environment where relations can be replicated but not fragmented. In the proposed algorithm, four types of ants collaborate to create an execution plan. Hence, there are four ant colonies in each iteration. Each type of ant makes an important decision to find the optimal plan. In order to evaluate the quality of the generated plan, two cost models are used—one based on the total time and the other on the response time. The proposed algorithm is compared with two previous genetic-based algorithms on chain, tree and cyclic queries. The experimental results show that the proposed algorithm saves up to about 80 % of optimization time with no significant difference in the quality of generated plans compared with the best existing genetic-based algorithm.

Mishra, Ms Anju, Ms Gunjan Nehru, and Mr AshishPandey(International Journal of Engineering 1.6 (2012))"Dynamic Programming Solution for Query Optimization in Homogeneous Distributed Databases"-The "multiple query optimization" (MQO) tries to reduce the execution cost of a group of queries by performing common tasks only once, whereas traditional query optimization considers single query at a time An optimal dynamic programming method for such high dimensional queries has the big disadvantage of its exponential order and thus we are interested in semi-optimal but faster approaches.

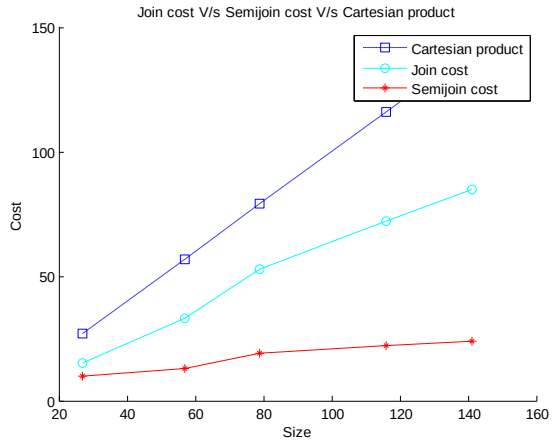
Stone, Paul D., et al. "Query Execution and Maintenance Costs in a Dynamic Distributed Federated Database." submitted to ACITA (2012) - The cost of query evaluation in a Dynamic Distributed Federated Databases (DDFD) depends on the topology connecting the database nodes together. Different topologies provide opportunities to adopt a variety of query optimization strategies and topology also influences the efficiency of these strategies. It describes a number of strategies to optimize join queries and then derive cost estimation formulae. The costs of maintaining these topologies are also formulated and compared. Costs formulas are defined at a coarse grained level, using a small number of parameters and derive only the dominant or average behaviors of the queries and topologies considered. This approach is intended to provide insights to likely optimization candidates, leading to further refinement of the models.

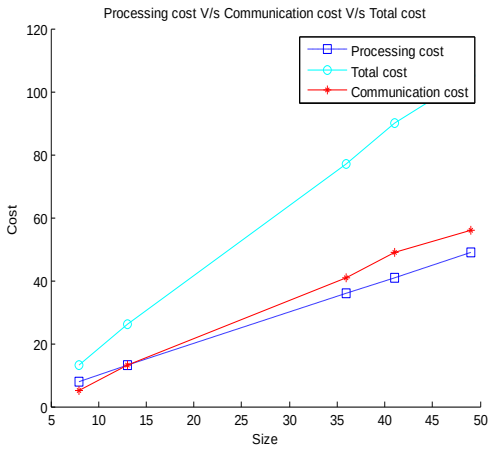
V. Results and Analysis

I have performed query optimization using the MATLAB to choose a good execution strategy for a given query [5]. It can

be observed from previous studies that MATLAB is usually better in choosing a good execution strategy than a traditional query optimization approach that uses a crisp cost model. Using MATLAB we have explored method to establish a good cost model [6][7]for Distributed Environment and by investigating existing algorithms based on cost model. The mathematical equations are easily modeled using MATLAB interface which has proved helpful to save the time in cost calculations. In this study DDBMS is developed using 1 Local and 4 Remote sites and the databases are selected randomly by the system in order to make comparative analysis of effect of JOINS, SEMI-JOINS and CARTESIAN PRODUCT on query cost estimation and makes clear justification that which is the best query operation to be applied and also gives the clear view of effect of query operations on Processing Cost, Communication Cost and Total Cost.

Graphical Results





➢ JOINS v/s SEMI JOINS v/s CARTESIAN PRODUCT

One of the interesting questions is when the query has to be executed with Join, Semi Join and Cartesian product. The selection of Join, Semi Join and Cartesian product in distributed system directly depends upon the data transmission from one site to another. In this study the major fact that came out is that joins and semi-joins are more useful than Cartesian product when small proportion of table is to be retrieved and cost is to be reduced and also semi joins are found more useful than join when the data transmission from one site to another is more.

VI. CONCLUSIONS

Accurate cost estimation is very important in distributed databases, because errors can have a huge impact on actual execution cost. The cost-based approach generally chooses an execution plan that is as good for large queries with multiple joins or multiple indexes [10][11]. The cost-based approach also improves productivity by eliminating the need to tune database statements on our own. The most important task for the CBO is to design an execution plan for an SQL statement. The CBO takes an SQL statement and tries to weigh different ways (plan) to execute it. It assigns a cost to each plan and chooses the plan with the smallest cost. The cheapest plan is the one that will use the least amount of resources (CPU, Memory, I/O, etc.) to get the desired output.

VII. REFERENCES

[1] Mishra,Ms Anju,Ms GunjanNehru, and Mr AshishPandey. "Dynamic Programming Solution for Query Optimization in Homogeneous Distributed Databases." International Journal of Engineering 1.6 (2012).

[2] Liu, Mengmeng. "Efficient optimization and processing for distributed monitoring and control applications." Proceedings of the on SIGMOD/PODS 2012 PhD Symposium. ACM, 2012.

[3] Xu, Zichen, Yi-Cheng Tu, and Xiaorui Wang. "PET: reducing database energy cost via query optimization." Proceedings of the VLDB Endowment 5.12 (2012): 1954-1957.

[4] Golshanara, Ladan, Seyed Mohammad Taghi Rouhani Rankoohi, and Hamed Shah-Hosseini. "A multi-colony ant algorithm for optimizing join queries in distributed database systems." Knowledge and Information Systems (2013): 1-32.

[5] Bausch,Daniel, Ilia Petrov, and Alejandro Buchmann. "Makingcost-basedqueryoptimizationasymmetry-aware." Proceedings of the Eighth International Workshop on Data Management on New Hardware. ACM, 2012.

[6] Chandramouli, Badrish, et al. "Accurate latency estimation in a distributed event processing system." Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, 2011.

[7] P.Stone,P.,P.Dantressangle,G.Bent,A.Mowshowitz, A.Toce,andB.Szymanski.Relationalalgebra-coarsegrained query cost modelsfor ddfds. In Proceedings of the Fourth Annual Conference of ITA, 2010.

[8] Catania, Barbara, and Lakhmi Jain. "Advanced Query Processing: An Introduction." Advanced Query Processing. Springer Berlin Heidelberg, 2013. 1-13.

[9] G.Bent.Hyperd: Analysis and performance evaluation of a distributed hypercube database databases. In Proceedings of the Sixth Annual Conference of ITA

[10] A. Toce, A. Mowshowitz, P. Stone, P.

Dantressangle, and G. Bent. Hyperd: A hypercube topology for dynamic distributed federated databases.In Proceedings of the Fifth Annual Conference of ITA, 2011.

[11] Görlitz, Olaf, and Steffen Staab. "Federated data management and query optimization for linked open data." New Directions in Web Data Management 1. Springer Berlin Heidelberg, 2011. 109-137.

[12] Stone, Paul D., et al. "Query Execution and Maintenance Costs in a Dynamic Distributed Federated Database." submitted to ACITA (2012).