European Journal of Academic Essays 1(3): 89-93, 2014
ISSN: 2183-1904
www.euroessays.org

Implementation of query optimization techniques in distributed environment through genetic algorithm

Mishra Sambit Kumar [1], Pattnaik Srikanta (Dr.)[2], Patnaik Dulu(Dr.)[3]

[1.] Associate Professor
Department of Computer Sc.&Engg.
Ajay Binay Institute of Technology, Cuttack, Odisha, India
sambit_pr@rediffmail.com

[2.] Professor, S.O.A. University, Bhubaneswar, Odisha, India
srikantapatnaik@hotmail.com
[3.] Principal, Government College of Engineering, Bhawanipatna, Odisha, India
patnaik_d@yahoo.com

Abstract

In distributed databases, usually the cost of optimization increases while consulting the underlying data sources in the query optimization process. The query optimizer may handle the data sources and the optimization techniques adopted are required to be implemented to all relevant cost information with minimal communication. It is understood that query optimization issues depend upon relations i.e. cardinality, size of a tuple and fraction of tuples participating in a join with another relation. Also it is seen that the query optimization issues depend upon attribute comprising with cardinality of domain, actual number of distinct values and Common assumptions. The query optimization task involves with selecting the plans and sub plans, evaluating the size of plans and sub plans along with cost of plans and sub plans. In many cases the query execution cost is measured in conceptual units. But In a distributed database, the query execution costs must be divided into multiple dimensions. In this paper it is intended to discuss the approaches of query optimization and enumerate the query optimization techniques in distributed environment. As it has been seen that selecting the optimal execution strategy for a query is NP-hard in the number of relations, genetic algorithm may be applied in this case to evaluate plan select value, CPU time, cost of query plans and sub plans including I/O cost. The purpose of doing so is to measure the cost of data sets while dealing with large data sets in the distributed environment.

Key words: Query optimization, query plan, tuple, cardinality, instruction cost, CPU cost, NP-hard, I/O cost.

1. Introduction

The distributed database technology includes schema integration, data transformation, distributed query processing and query optimization. The distributed query processors usually need three basic requirements.
(i)Need of Query Processing: In a large scale distributed system, both data access and computation may be carried out at various sites.
(ii)Need of Cost Factors: In a centralized DBMS, query execution cost is a single dimensional factor measured in conceptual units.In a distributed database, costs must be divided into multiple dimensions under the control of single logical database.

(iii)Need of Cost Estimation: The implementation mechanism of query optimization process is motivated by the necessity of the cost estimation in
the part of the query optimizer. Usually the optimization algorithms are divided into three steps.
Step 1: Selecting the sub plans that require cost estimates.

Step 2: Evaluate the size of plans and sub plans.

Step 3: Calculate the costs for plans or sub plans. Evaluate execution plan for the query.

Corresponding Author: Mishra Sambit Kumar

To minimize the retrieval of large datasets, cost measurement of the required data sets is essential.
While optimizing multiple queries at a time in the distributed environment, it is observed that it is efficient with more number of similar queries also the decision space is quite large.
Query optimization issues depend upon relation, i.e. issues involved with cardinality, size of a tuple and fraction of tuples participating in a join with another relation. Also the query optimization issues depend upon attribute comprising with cardinality of domain, actual number of distinct values and Common assumptions i.e. independence between different attribute values and uniform distribution of attribute values within their domain.
In general there may be five requirements for distributed queries.
(i)Conserve resources : Since there is no restriction on how many rows (or objects) a query may return, resources must clearly be conserved to cater for the potentially unlimited set of results. Conservation of resources is achieved on the server by never holding onto any database resources.
(ii)Response in a reasonable amount time: The query should take about the same time to execute as a local query (i.e. within the same order of magnitude). That is, it must respond with the first row(s) of information with the same latency as a local query.
(iii)Robustness : There are two forms of robustness. The first form of robustness relates to the returned results. Whilst the client is iterating over the result set, they do not want other users to interfere with the returned results. The second form of robustness relates to remote resources. If the network connection is lost for any reason, relying on obscure error conditions to handle server resource cleanup is undesirable.

Therefore, server resources are not held open across client requests.
(iv)Not limited partial result set : This requirement simply states that any method for retrieving information must not limit the number of rows returned. All rows of query data should be returned from the server.
(v)Intuitive to Use : The client developer should find retrieving data similar to or better than current local query approaches. In addition, the client using the query method should not be required to make allowances for the fact that it is a remote query. They should be able to use the same client code for local and remote queries.

2. Review of Literature

Lanzelotte et.al[1] have discussed in their paper about the inadequateness of the enumerative strategies while optimizing complex queries due to the large number of execution plans. To resolve that particular problem, random strategies were being used. Usually the transformational approach characterizes that particular kind of strategies.
Generally, heterogeneity and autonomy of data sources characterize data integration systems. Sources might be restricted due to the limitation of their query interfaces or certain attributes must be hidden due to privacy reasons. To handle the limited query capabilities of data sources Manolescu et.al[2] have introduced dependant join operator which is asymmetric in nature.
Oszu et.al[3]have discussed in their paper that the goal of query optimization is to find an execution strategy for the query that is close optimal. An execution strategy for a distributed query may be described with relational algebra operations and communication primitives for transferring data.

Oszu et.al[4] have discussed in their paper that the selection of the optimal strategy generally requires the prediction of execution cost of the alternative candidate ordering prior to actually executing the query. The execution cost is
expressed as a weighted combination of I/ O, CPU, and communication costs.

Ioannidis et.al[5] have discussed in their paper about the generation of codes for the selected
Query execution plans which may then be executed in either compiled or interpreted mode to produce the query result.

Kossmann et.al[6] have discussed in their paper about the particular use of indices by
the optimizer to execute a query and in which order the operations of a query may be executed. The optimizer enumerates alternative plans, estimates the cost of every plan using a cost model, and chooses the plan with lowest cost.

Ibraraki T et.al[7] have discussed in their paper that selecting the optimal execution strategy for a query is NP-hard in the number of relations. For complex queries with many relations, this incurs a prohibitive optimization cost. Therefore, the actual objective of the optimizer is to find a strategy close to optimal and to avoid bad strategies.

G. Graefe et.al.[8] have discussed in their paper that although distributed query processing is a well-studied problem modern architectures pose new challenges and opportunities for fine-grained parallelization at all levels ranging from intra-operator level up to the workload level.

O. Gorlitz et.al[9] have discussed in their paper that the optimization may be split into conventional local optimization for choosing access strategies and the global optimization where join ordering and site selection are the main tasks. For site selection two fundamental options exist: data shipping where the data is transfered from the storing site to the site executing the query and query shipping where the evaluation of
the sub query is delegated to the storing site.

3. Problem Statement

3.1. Example

SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO
AND DUR > 37

Strategy 1

∏ENAME(σDUR>37∩EMP.ENO=ASG.ENO (EMP X ASG))

Strategy 2

∏ENAME(EMP X ENO (σDUR>37 (ASG)))

Here in this case Strategy 2 avoids Cartesian product. To minimize the cost function, the I/O cost, CPU cost and communication cost of the query plans must be evaluated. The query plans may have different weights in different distributed environments.

Total cost of query plans= 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

Elapsed time between the initiation and the completion of a query may be evaluated by calculating the response time, CPU time, I/O time and communication time.
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 = unit msg initiation time*no. of sequential msg + unit transmission time * no. of sequential bytes.

3. Complexities of relational operations

Assume the relations of cardinality n with sequential scan .
Table 1. Complexities (Relational operations)

Operation	Complexity
Select	O(n)
Project (without duplicate elimination)	
Project (with duplicate elimination)	O(nlog n)
Group	
Join	O(nlog n)
Semi-join	
Division	
Set Operators	
Cartesian Product	$O(n^2)$

While moving inner relation to the site of outer relation, joining may not be possible in their arrival. For storing the total cost of outer tuples corresponding to the inner tuples may be required.
Total Cost = cost(retrieving qualified outer tuples) + no. of outer tuples fetched * cost(retrieving matching inner tuples from temporary storage) + cost(retrieving qualified inner tuples) + cost(storing all qualified inner tuples in temporary storage) + msg. cost * (no. of inner tuples fetched * avg. inner tuple size) / msg. size.

4. Materials and Methods

This problem may be represented as a genetic algorithm problem, where it is intended to determine the chromosome, genetic algorithm operators. The individual query plan may be represented as chromosome and sub plans/ tasks may be represented as gene. For the crossover, one point in the selected chromosome might be selected along with a corresponding point in another chromosome and then the tails might be exchanged. Mutation processes causes some bits to invert and produces some new information. Therefore the best individual may be used to proceed forward to the next generation. After performing operations, some chromosomes might not satisfy the fitness and as a result the algorithm discards this process and gets **q** (q<=n) children chromosomes. The algorithm then selects **n** chromosomes with the lower fitness value from the **q+n** chromosomes (**q** children and **n** parents) to be parent of the next generations. This process will be repeated until a certain number of generations are processed, after which the best chromosome is chosen.

5. Problem Formulation

5.1. Algorithm

Step 1 : Set maximum generation and relations.
Step 2 : Set maximum number of relations.
Step 3 : Assign maximum number of queries.
Step 4 : Assign the required length of query plan, where query plan can be termed as chromosome.
Step 5 : Evaluate population by considering number of queries and query plans.
Step 6 : Set probability for crossover, pc
Step 7 : Set probability of mutation, pm
Step 8 : Evaluate I/O operation time
Step 9 : Calculate plan select value by taking number of queries and query plans into consideration.
Step 10 : Evaluate CPU time during processing.
Step 11: Evaluate estimated cost of plan by considering plan select value, total number of queries along with number of relations.
Step 12: Evaluate total cost of plan by considering CPU cost, I/O cost and communication cost into account.CPU cost may be obtained by multiplying unit instruction cost with number of instructions.
Similarly I/O cost may be obtained by multiplying unit disk I/O cost with number of disk I/Os.

5.2. Experimental analysis

Maximum generations=100
Numberofqueries=100
Number of relations=100
Size of Chromosome (Plan in a query)=5

Probability of crossover, Pc= 0.07

Probability of mutation, Pm=0.002

Table 2. Query plan along with cost

Sl.No.	Query Plan	Est_Cost	Total_Cost
1	7	0.018344	0.36288
2	15	0.018344	0.36688
3	18	0.018419	0.36838
4	24	0.018569	0.37138
5	25	0.018144	0.37188

6. Discussion and future direction

We have already discussed the optimization technique in terms of cost evaluation of query plans along with algorithm. The information may be available through standard set of rules which allows querying the primary database about statistics, or by caching statistics from before query executions. Usually the communication costs remain constant for the duration of optimization and execution of the query. The tasks/ sub plans may have some cost associated with them which
reflects both the CPU and I/O cost required to process them. It is seen that the cost of an access plan is the cost of processing its component tasks.
In a distributed environment, a global access plan for a set of queries corresponds to a plan that provides a way to compute the results of all queries.In this case the access plan may be constructed by choosing one plan for each query and then merging them together. It is clearly understood that this type of problem is NP-hard . A nondeterministic algorithm needs only one plan for each query and check whether the cost of the global access plan obtained by merging the guessed local access plans is less than or equal to combining the access plans which can be easily done in polynomial time and therefore the checking steps takes only polynomial time.

7. Conclusion

As we have seen that each query may have a number of possible solution plans, and also each plan of a query may contain a set of tasks, where each task may have associated cost, therefore the cost value may represented by a positive integer number. Alternative plans of a query , and other queries in the query set , may contain the same task. It is required to determine a set of tasks, with minimal total cost which may contain all the tasks of at least one plan of each query. The evaluation of query plans, evaluating cost of query plans and specific implementations of query optimization is very much essential for distributed database systems. In this case the optimizer must have to consult the data sources to evaluate the cost of that operation. The process implemented in this case indicates that, when the physical database design is known to the optimizer, this query optimization algorithm works with accuracy.

8. References

[1] Lanzelotte, R.S.G., Zaït, M., Gelder, A.V.: Measuring the effectiveness of optimization. Search Strategies. In: BDA 1992, Trégastel, pp. 162–181 (1992).

[2] Manolescu, I., Bouganim, L., Fabret, F., Simon, E.: Efficient querying of distributed resources in mediator systems.In:Meersman, R., Tari, Z., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 468–485. Springer, Heidelberg (2002).

[3] Oszu, M. T. and Valduriez P., "Distributed and Parallel Database Systems," in Trucker A. (Ed),
The Computer Science and Engineering
Handbook, CRC press, pp. 1093-1111, 1997.

[4] Oszu M. T. and Valduriez P., Principles of
Distributed Database Systems, Prentice Hall
International, NJ, 1999.

[5] Ioannidis Y. E., "Query Optimization," in Trucker A. (Ed), The Computer Science and Engineering Handbook, CRC press, pp. 1038- 1054, 1996.

[6] Kossmann D. and Stocker K., "Iterative Dynamic Programming: A New Class of Query Optimization Algorithm," *ACM TODS*, March 2000.

[7] Ibraraki T. and Kameda T. "Optimal Nesting for Computing N-Relational Joins," ACM Transactions on Database Systems, vol. 9, no. 3, pp. 482-502, 1984.

[8] G. Graefe. Parallel Query Execution Algorithms. In Encyclopedia of Database Systems , pages 2030{2035. 2009.

[9] O. Gorlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID descriptions. In COLD'11 , 2011.

Authors' Profile

Er. Sambit Kumar Mishra has obtained B.E. and M.Tech. in Computer Science & Engineering from Amaravati University, Maharastra and Indian School of Mines, Dhanbad respectively. He is now serving in ABIT, Cuttack as Associate Professor in the department of Computer Science & Engineering. He has submitted his Ph.D thesis for evaluation. He has 13 number of publications in reputed International Journals. Recently he is also reviewer of peer reviewed international Journals, e.g . International Journal

of Scientific and Engineering Research (IJSER) and International Journal of Information Technology and Computer Science(IJITCS), European Journal of Academic Essays.

Prof. (Dr.)Srikanta Pattnaik has obtained B.E. degree in Electronics & Telecommunication Engineering from UCE, Burla and M.E. & PhD from Jadavpur University. Presently he is working as Professor at S.O.A. University, Bhubaneswar. He is also Editor in Chief of many reputed International Journals like International Journal of Information and Communication Technology, International Journal of Computational Vision and Robotics , SPRINGER-VERLAG etc.

Prof.(Dr.)Dulu Patnaik is presently working as Principal, Government College of Engineering, Bhawanipatna. He obtained his M.E. from N.I.T., Rourkela and Ph.D from Indian School of Mines, Dhanbad. He is also Editor and Chief Editor of many reputed International Journals.