ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 11, November 2016

Query Optimization to Minimize Cost and Latency for Crowdsourcing Systems

[1]Salma Eram, [2]Jayashree S Patil

[1]M. Tech Student, Department of CSE, G.Narayanamma Institute of Technology and Science, Telangana, India.

[2]Associate Professor, Department of CSE, G.Narayanamma Institute of Technology and Science, Telangana, India.

Abstract— In recent time's crowdsourcing systems, such as CrowdDB, Qurk and deco provides a SQL-like query language as a declarative merge with the crowd. Declarative crowdsourcing is designed for hiding the complexities and also to decrease the burden of the user in dealing with the crowd systems. Hence the user is only enforced to submit a sql-like query so that the system takes the responsibility of compiling the query, generating the execution plan and deciding in the crowdsourcing marketplace. A given query can have many alternative execution plans and the difference in crowd sourcing cost among the best and the worst plans may have several orders of magnitude. To run over this issue, there is a method called query optimization which evaluates the cost and latency for crowd-sourcing systems. Now the cost and latency are taken into the consideration as query optimization objectives for generating the query plans so that it provides a good balance between the cost and latency. A scalable algorithm called Genetic Algorithm is used to optimize two types of operators: selection queries and join queries. GA is applied to problem for obtaining optimal solution .The final solution is obtained from genetic algorithm as a best query plan with minimum cost and latency. Performance is evaluated for both the operators in terms of cost and latency.

Keywords: Query Optimization, Crowd-sourcing, Genetic Programming

1. INTRODUCTION

Crowd-sourcing help developers to incorporate human computation into different tasks, which are difficult for computer algorithms to perform well such as tagging images, categorizing products and extracting sentiments from Tweets; To take advantage of crowdsourcing in practically, developers have to write a code using low-

level APIs, which turns to some common test even for simple applications such as, to accommodate existing data with crowd-sourced data; by resolving the inconsistencies in crowd-sourced data might lead to improve the data quality and to optimize crowd-sourcing workflows for monetary cost and latency. CrowdDB, Qurk and Deco are the various crowd-sourcing systems which provide an SQL-like query language as a declarative interface to the crowd. To cover the complexities while dealing with the crowd an SQL like declarative interface is designed. It also provides the crowd-sourcing system as an interface which is more familiar to the database users. Therefore, for a given query, a declarative system must first compile the query, to generate the execution plan, and post the human intelligence tasks (HITs) to the crowd according to the plan, it collects the answers, handle errors and resolve the inconsistencies in the results.

By leveraging human intelligence a crowd-source has established a variety of opportunities for many challenging problems. For example, applications such as image tagging, natural language processing, and semantic-based information retrieval can exploit crowd-based human computation to supplement existing computational algorithms. Naturally, human workers should have knowledge, experience, and perception to solve problems in crowd-sourcing. Therefore it is not clear that which problems can be better solved by crowd-sourcing than by solving traditional machine-based methods. Therefore, a cost sensitive quantitative analysis method is needed.

Query optimization is an operation of frequent relational database management systems. The query optimizer workout to regulate the most active way to evaluate a given query by considering the possible query plans. Practically, query optimizer doesn't directly communicate with the user once queries are submitted to database server, firstly it is parsed by the parser; then the query is moved to the query optimizer where optimization occurs. By the reason of database structures are complicated, in most cases, and especially for not very simple queries, the needed data for a query can be gathered from a database by bring it in different ways, through different data-structures, and in different orders. Each different way normally requires different processing time. Processing times of the same query may have high difference, next to minutes, hours, depending on the way it is being selected. An automated process uses

plan of query optimization to find out the way to measure a user query in a less amount of time. Hence there must be system that helps to analyze the query, optimize it, find query evaluation plans and finally predict potential query plan for execution over crowd sourced data.

Problem Definition

Declarative query is used to improve the usability of the system, which requires the system to have the capability to optimize and provide an "optimal" query execution plan for each query. Since a declarative crowd-sourcing query can be executed in different ways, the choice of execution plan has a significant effect on overall performance, which includes the number of questions being asked, the types/difficulties of the questions and the monetary cost incurred.

Problem Solution

To run over these issues it is important to design an efficient query optimizer that selects the "best" query plan from all potentially good query plans based on a cost and latency optimization objectives. A proposed approach uses genetic algorithm along with the query optimization technique to find the most efficient query plan for answering a query.

2. RELATED WORK

Genetic Algorithms (GA)

Genetic algorithms (GA) first described by John Holland in 1960s and further refined by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s To extract nature optimization strategies, GA uses Darwinian Evolution and translate them in order to find out the global optimum in defined phase space by mathematical optimization theory. A set of methods or techniques, inheritance or crossover, mutation, natural selection, and fitness function are defined by GA to find approximate solutions to difficult problems. Such methods are basis for evolutionary biology which is applied to computer science.

GA is widely used and accepts method for very difficult optimization problem. One of the most important optimization problems in computer science, query optimization for large join query is genetic algorithm. GA is used to solve a wide range of problem such as data mining, optimization, games, evolving behavior in biological communities etc.

To increase the speed of long-running sql queries a Query optimization is required. The *query* optimizer tries to consider the possible *query* plans to determine the most useful way to execute a given *query*. The optimizer may not choose the best answer on its own when there is a difference in the amount of time spent for computing the best query plan and the quality of the choice. To balance these two, different qualities of database management systems (DBMS) have been used. Hence the query optimizer is used to evaluate the resource footprint of various query plans based on cost and use this as the basis for plan selection. For each possible query plan an estimated "cost" is assigned, and choose the plan with the least cost.

There are basic steps in query processing, which includes Query Parser, Query Optimizer, Query Execution / Evaluation.

- Query Parser test the validation of the query and then converts it into an internal form.
- Query Optimizer checks all identical evaluation plan then chooses the one with least cost. Cost is estimated based on number of tuples or size of tuples. It also evaluates latency based on processing time.
- Query executor takes a query evaluation plan, executes that plan and returns optimal query with minimum cost and latency.

Under Query optimizer an efficient algorithm called as genetic algorithm is used to generate a best execution plan with minimum cost and latency.

Apply GA on select query and join query after parsing the initial query plan.

Procedure:

1. Working principle of GA is to create the population (set of solutions) of randomly generated individuals (solutions to the problem).
2. After creating the population, fitness of every individual in population is evaluated.
3. Then by taking the loop for all relations in query, it selects 2 or more individuals from current population (based on fitness) and performs crossover on selected individuals to form new population.
4. After completing the crossover mutation is performed.
5. Then the fitness of every individual for new population is evaluated and terminates when satisfactory fitness level

and optimal solution has been reached for population or go for next iteration.

A given query can be executed in many ways which leads to alternative execution plans but the difference in the best and the worst plans may have several orders of magnitude in terms of crowd-sourcing cost. Execution plan has a significant impact on overall performance which includes the number of questions being asked, types/difficulties of the questions and the monetary cost incurred.

3. PROPOSED APPROACH

In traditional databases, crowd-sourcing systems have optimization mechanisms which can be broadly classified into rule-based and cost-based. A rule-based optimizer defines the set of rules instead of calculating the cost to generate the best query plan. CrowdDB [1] is an example of a rule-based query optimizer which calculates based on several rewriting rules such as predicate push-down, join ordering, etc. While rule-based optimization has limited optimization capability and often leads to ineffective execution plans hence it is easy to implement. In cost-based optimizer, it calculates the cost of alternative query plans for executing a query and uses the one with lowest estimated cost.

So, the proposed approach designs an efficient crowdsourcing query optimizer selects the "best" query plan from all potentially good query plans based on a cost and latency optimization objectives. Proposed approach is used to find the most efficient query plan for answering a query by applying genetic algorithm along with query optimization method. Following are the advantages for proposed approach, Generates good query plans that should have good balance between the cost and latency optimization objective.

1. Supports select and join operator.
2. Query optimization objectives to minimize the latency under user-defined cost budget.
3. To develop efficient algorithm for optimizing selection, join queries.

Proposed approach considers two commonly used operators in crowd-souring systems: SELECT asks the crowd to filter items satisfying certain constraints and JOIN asks the crowd to match items according to some criteria. Considering the existing crowdsourcing systems, CrowdDB focuses

on optimizing SELECT operator, Qurk focuses on optimizing JOIN operator but proposed approach focuses on both select and join operator.

This paper proposes on an algorithm called genetic algorithm. When we provide query as an input then the main focus will be on query optimizer. Firstly query parser parses the query wherein it translates into internal form from a given dataset. Then, the query optimizer takes good query plans and estimates the cost and latency of each query plan. Cost estimation is measured based on number of tuples and latency is generally measured as total elapsed time for answer query. Query executor executes the query plans of select and join operator by using GA then resultant is obtained as best query plan with minimum cost and latency. Finally performance is evaluated for two operators in terms of cost and latency.

4. CONCLUSION AND FUTURE WORK

A query can be executed in many ways but the difference in crowdsourcing cost between the best and the worst plans may have several orders of magnitude. To overcome this issue, the cost and latency query optimization objectives are taken into the consideration for generating the query plans .In this paper we propose an algorithm called genetic algorithm wherein it selects all possible query plans and considers only the optimal query plan which has minimum cost and latency. It also supports select and join operator. Performance is evaluated for both the operator in terms of cost and latency.

In the future we would like to study how to incorporate correlations between select/join conditions into the optimizer for complex queries, and we also plan to extend proposed system to support more advanced SQL operators, such as sorting and aggregation.

REFERENCES

[1] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, “CrowdDB: Answering queries with crowdsourcing,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 61–72.

[2] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller,Human-powered sorts and joins, PVLDB, 5(1):13–24, 2011.

[3] J. Gao, X. Liu, B. C. Ooi, H. Wang, and G. Chen, “An online cost sensitive decision-making method in crowdsourcing systems,” in Proc. ACM

SIGMOD Int. Conf. Manage. Data,pp. 217–228, 2013.

[4] Dr. P.K.Butey, Prof. Shweta Meshram & Dr. R.L. Sonolikar, Query Optimization by Genetic Algorithm. In Proc. of the 2012:44-51, ISSN: 2229-7421, International Science Press

[5] A. Swami and A. Gupta, Optimization of large join queries. In Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data, pages 8-17, Chicago, IL, June 1988.

[6] A. Marcus, E. Wu, S. Madden, and R. C. Miller, "Crowdsourced databases: Query processing with people," in Proc. 5th Biennial Conf. Innovative Data Syst. Res., 2011, pp. 211–214.