© 2016 IJSRST | Volume 2 | Issue 6 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Engineering and Technology

Query Optimization for Declarative Crowdsourcing System

Nilesh. N. Thorat[1], A. B. Rajmane[2]

[1]ME CSE Student , [2]Associate Professor

Department of Computer Science and Engineering, Ashokrao Mane Group of Institution, Vathar tarf, Maharashtra, India

ABSTRACT

Crowdsourcing is a distributed problem-solving, production model that has emerged in recent years. crowd sourcing is designed to hide the complexities as well as relieve the user from burden of dealing with the crowd data.. The user is requested to pass sql queries to the crowd system to generate the execution plan. Passed query is executed based on the alternative execution query plans in crowd sourcing. Here, CROWDOP a cost-based query optimization approach for declarative crowd sourcing systems is implemented. This considers both cost and latency in query optimization and provides balance between both of them. For this CrowdOp utilizes three types of queries: join queries, selection queries, and complex selection-join queries. At the end results are compared and evaluated.

Keywords : Crowdsourcing, query optimization, human intelligence tasks (HIT).

I. INTRODUCTION

Crowdsourcing has attracted growing interest in recent years as an effective tool for harnessing human intelligence to solve problems that computer cannot perform well, such as document translation, handwriting recognition, audio transcription and photo tagging. Various solutions have been proposed for performing common database operations over crowdsourced data, such as selection (filtering), join, sort/rank and count. Recent crowdsourcing systems, such as CrowdDB, Qurk and Deco, provide an SQL-like query language as a declarative interface to the crowd. An SQL like declarative interface is designed to encapsulate the complexities of dealing with the crowd and provide the crowdsourcing system an interface that is familiar to most database users. Consequently, for a given query, a declaratives system must first compile the query, generate the execution plan, post the human intelligence tasks (HITs) to the crowd according to the plan, collect the answers, handle errors and resolve the inconsistencies in the results.

Declarative querying improves the usability of the system, it requires the system to have the capability to optimize and provide a “near optimal” query execution plan for each query. Since a declarative crowdsourcing query can be evaluated in many ways, the choice of execution plan has a significant impact on overall performance, which includes the number of questions being asked, the types/difficulties of the questions and the monetary cost incurred. It is therefore important to design an efficient crowdsourcing query optimizer that is able to consider all potentially good query plans and select the “best” plan based on a cost model and optimization objectives.

To address this challenge,we propose a novel optimization approach CROWDOP to finding the most efficient query plan for answering a query,supporting cost-based query optimization. Like in traditional databases, optimization mechanisms in crowdsourcing systems can be broadly classified into rule-based and cost-based systems. A rule-based optimizer simply applies a set of rules instead of estimating the cost to determine the best query plan. CrowdDB[3] is an example system that employs a rule-based query optimizer based on several rewriting rules such as predicate push-down, join ordering, etc. While rule-based optimization is easy to implement, it has limited optimization capability and often leads to ineffective execution plans. CROWDOP, in contrast, adopts cost-based optimization that estimates the cost of alternative query plans for evaluating a query and uses the one with the lowest estimated cost. Optimizing multiple crowdsourcing operators. CROWDOP considers three commonly used operators in crowdsouring systems: FILL solicits the crowd to fill in missing values in databases; SELECT asks the crowd to filter items satisfying certain constraints; and JOIN leverages the crowd to match items according to some criteria. Considering the existing crowdsourcing database systems, Deco[7] focuses on optimizing FILL operator, Qurk[10] on optimizing JOIN operator, and the two recent crowdsourcing algorithms, CrowdScreen[8] and CrowdFind[9], are designed for optimizing SELECT operator. CROWDOP supports cost-based optimization for all three operators. Instead of optimizing the cost of each individual operator independently, CROWDOP optimizes the overall cost of all operators involved in a query and derives the “best” query evaluation plan. Two key performance concerns in crowdsourcing systems are monetary cost (how much people pay for crowdsourcing) and latency (how long people wait for results). A good query optimizer should consider the tradeoff between these factors and perform a multi-objective optimization. Neither single objective solution, i.e., minimizing the cost but incurring heavy latency nor reducing latency but incurring high cost, is desirable. We examine recent crowdsouring works and find most query optimizers only search for query plans with the minimal monetary cost. The only approach taking latency into account is CrowdFind that studies the tradeoff between cost and latency for finding a limited number of items. CROWDOP incorporates the cost-latency tradeoff into its optimization objectives. It is capable of finding the query plan with low latency given a user-defined budget constraint, which nicely balances the cost and time requirement of users. We summarize our contributions to study 1)cost-based query optimization that considers cost-latency tradeoffs and supports multiple crowdsourcing operators. 2) We formalize query optimization objectives to minimize the latency under user-defined cost budget. 3) We develop efficient algorithms for optimizing selection, join and complex queries.

II. METHODS AND MATERIAL

A. Literature Review

1. Davidson, Khanna, Milo,Roy[1], concluded that Group-by and top-k are the most basic constructs in database queries. The criteria used for grouping and ordering certain types of data – for example unlabeled photos clustered by the same person ordered by age – are difficult to evaluate by machines. While evaluating top-k and group-by queries with the help of crowd the answer may be either type or value questions. Suppose that two data elements are given, then answer to a type question is “yes” if the elements have same type, so they belong to same cluster or identical group i.e. two data elements ordered based on answer to value question. Results from crowd source are fetched using predefined assumption but it may be incorrect. They introduced efficient algorithms for top-k and group-by ,problems in crowd source systems,which gives results with high probability.

2.Ju Fan, Meiyu Lu, Beng Chin Ooi, Tan, Zhang[2], concluded that , the web is full featured data in terms of HTML tables . If these HTML tables are integrated gives rise to a knowledge repository but semantic correspondences between web table columns need to be checked, it can be carried out with help of conventional schema matching but they won’t produce good result as sometime it may be incomplete. They proposed system with two solutions for web table matching which solves semantic correspondences and schema matching. First, concept-based approach is designed which deals with mapping of each column of web table to best concept, which solves problems for columns which are disjoint ,due to incomplete values of columns. Second, hybrid machine crowd sourcing framework deals with incomplete column with concept matching tasks to the crowd under a given budget and utilizes the crowdsourcing result to help the algorithm to produce the best matches for the rest of the columns.

3.Franklin,Tim Kraska, Ramesh, Reynold Xin[3] designed CrowdDB system which performs a computationally difficult functions, such as matching, ranking, or aggregating results based on fuzzy criteria. CrowdDB takes input from human with help of crowd source system for providing information that is missing from the database which cannot easily got answers database systems as well as search engines. CrowdDB have resemble with traditional database system with some big change. Traditional database systems does not take human input for query processing. From an implementation point of view human-oriented query operators are needed to integrate as well as cleanse crowdsourced data. Performance as well as cost depends on a number of new factors including worker affinity, training fatigue, motivation and location.

4.Chien-Ju Ho, Jabbari and Vaughan[4], concluded that Crowdsourcing markets is a tool to collect data from very different workers. Workers use labels for classification of common tasks but it may be error prone, at a particular time it can be treated as spam also. The solution to this problem can be obtained by collecting labels for each instance from multiple workers. With the help of online primal-dual techniques, classification tasks of task assignment and label assisgnment for workers can be carried out in heterogeneous way. They show that adaptively assigning workers to tasks can lead to more accurate predictions at a lower cost when the available workers are diverse.

5.H. Park and J. Widom[5] designed comprehensive system named "Deco" which deals with answering the query depending on stored relational data together with data obtained from the crowd . The basic objective is to fetch best query plan with the help optimized query on the basis of user estimated monetary cost.Novel techniques are used in Deco's query optimizer system which include cost model that can easily differentiate between "free" existing data versus paid new data. Cardinality estimation algorithm deals with changes to the database state during query execution. Plan enumeration algorithm uses common subplans repeatedly in a setting that makes reuse challenging.

6.A. D. Sharma, H. Garcia-Molina,A. Parameswaran, and A. Halevy[6], proposed a system named CrowdFind which deals with problem of searching some items which satisfy fixed properties within data set for humans. Suppose that a human wants to identify total no of travelling photos from a travel website, since the data for this constraints may be very large, also monetary cost and latency would be also large. They proposed optimal algorithm which has comparison capacity between statistic cost versus actual time to evaluate the query. They study the deterministic as well as error-prone human answers, along with multiplicative and additive approximations. Lastly, They study how they may design algorithms with specific expected cost as well as time measures.

B. Proposed Work

In this paper, we focus on studying the cost-latency optimization problems while assuming the accuracy issue have been adequately addressed. The basic idea of the algorithm is to first solve the latency bounded cost minimization problem, for given query Q and a latency constraint L, finds the query plan with latency bounded by L and minimum cost. Declarative crowdsourcing is designed to hide the complexities and relieve the user the burden of dealing with the crowd. we proposes CROWDOP, a cost-based query optimization approach for declarative crowdsourcing systems.

Advantages

- ✓ Considers both cost and latency.
- ✓ Generates query plans that provide a good balance between the cost and latency.
- ✓ Supports different crowd sourcing operators.

Following techniques are used for proposed work

- ✓ Fuzzy criteria is used to perform computationally difficult functions, such as matching, ranking, or aggregating results for missing fields in dataset.
- ✓ OPTIFRAMEWORK algorithm is used to accomplish Cost Minimization objective.
- ✓ OPTISELECT algorithm is used to optimize select query cost.
- ✓ OPTIJOIN algorithm is used to optimize join query cost.
- ✓ LATENCYOPTI algorithm is used to optimize cost for complex query.

C. Proposed Architecture

The proposed architecture consists of six modules.

1. Data Extraction

This module is for generating dataset and extracting into the data table. This process efficiently reads all the attribute values from the dataset and loads it into the specified data table for classification process. Here the operator called FILL initially finds all the NULL attributes and replaces the column value with the predefined match. So that this extracted data can be efficiently processed further.

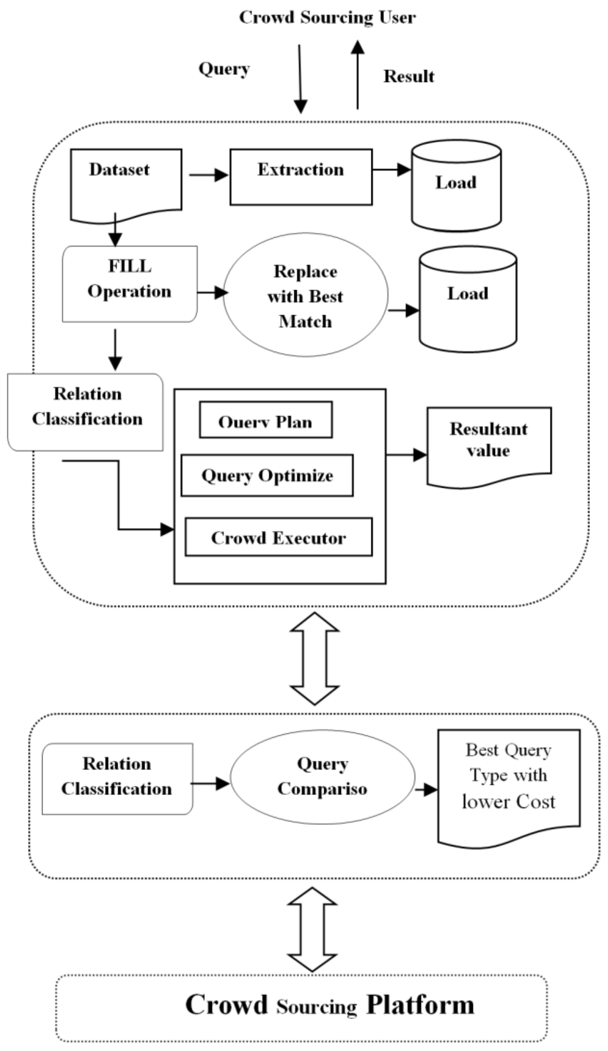


Figure 1: System Architecture

2. Relation Classification

This process is for classify the extracted dataset into multiple tables. From the loaded dataset table this classification process separates the columns and groups them into new table. Separated tables must contain at least one common attributes to process the queries.

3. Query Plan Generation

Here the query plan is generated based on the query asked. We generate select, join and complex queries for the given query. Based on the chosen query type the query plan is generated. This query plan contains the well matched query to get the required resultant value.

4. Query Optimization

This module executes the query plan for each query type offered. For the selection query type it initially read the query and executes them to fetch the appropriate result. Likewise, the remaining join and complex queries are executed and viewed on the table.

5.Crowd Sourcing Executor

Crowd sourcing executor estimates the crowd of each query execution and stores it on a database for calculating cost and latency. This execution is done based on the query execution start time and the end time. For each query type this process is followed and cost is estimated.

6. Cost & Latency Calculation

This module shows the cost and latency calculation of each query execution. The execution cost and latency may differ based on the query type is used for. Finally it compares the best query type for executing the given query and returns- the expected result.

D. Implementation Steps

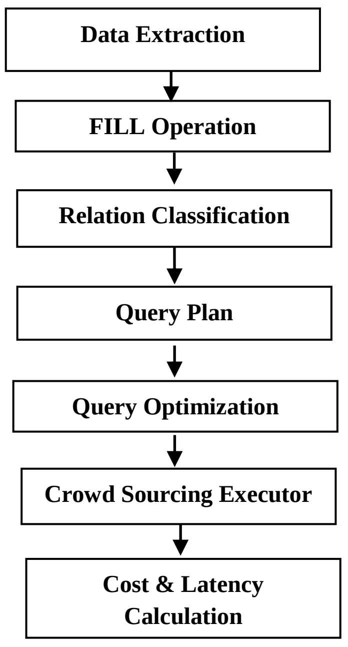


Figure 2: Implementation Steps

In first step we load the dataset which is obtained from website. The data set is in the text file format, first extract and then transform it to SQLSERVER database. In second step missing fields from datasets are filled with default values. In third step dataset is classified into different relation sets. In fourth step query is obtained from user for optimization purpose. In fifth step user query gets optimized for select ,join and complex join. In sixth step result gets displayed for user submitted query. In seventh step cost and latency for user query gets calculated and displayed to end user.

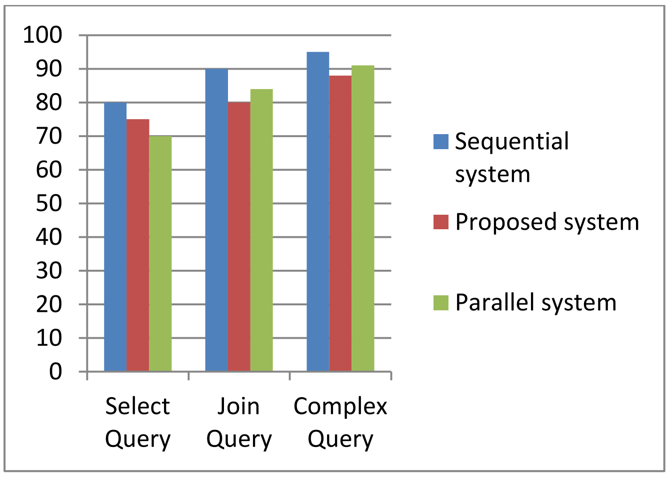
E. Scope

The main goal of the proposed work is to find best query execution plan based query optimization considering cost as well as latency constraint

III. RESULT AND DISCUSSION

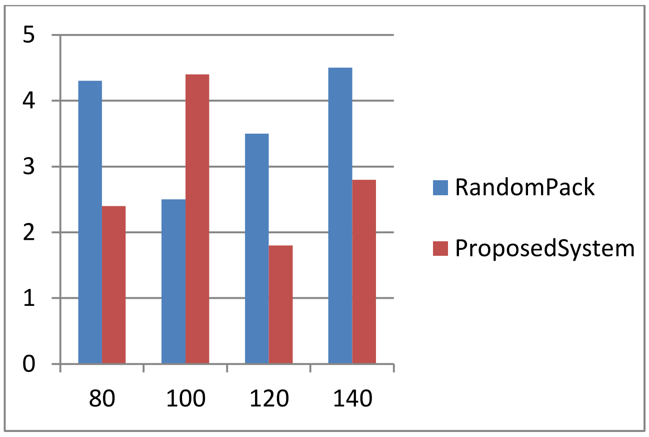
In the experiments, linear price function b + wx is used to evaluate cost and latency tradeoff between our proposed system with other existing systems.

For CSELECT and CJOIN, we set both base charge b and incremental charge w, while for CFILL, b and w are set to $0.01 (because filling a missing value is generally more expensive) and $0.002 (as some attributes have large domain)



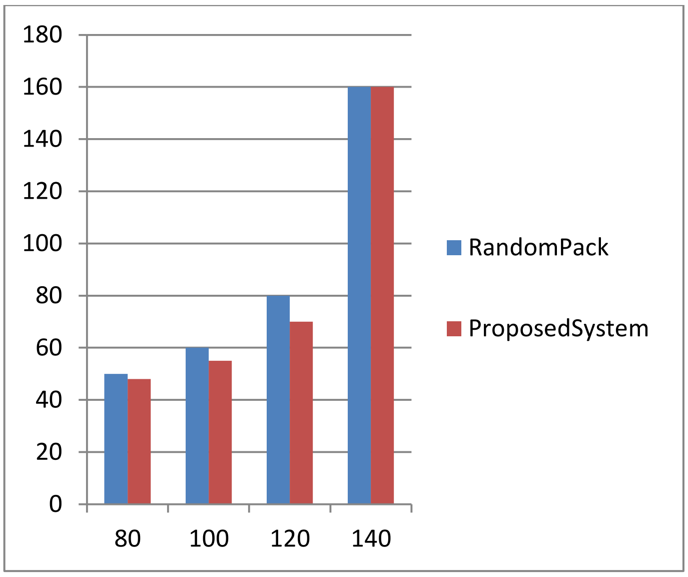
Graph 1: Query cost comparison

Following chart shows budget bounded latency. X axis represents user cost and y axis represents latency. For this purpose we used different number of selection conditions varying from 2 to 5,and note down respective cost.



Graph 2 : Cost bounded latency

Following chart shows cost on budget. X-axis shows user required budget and Y axis shows cost of query evaluation.



Graph 3: Cost on budget

IV. CONCLUSION

The best possible and best effective optimization algorithm is used for select, join, complex query. In the present time, simulated as well as real crowd experiments demonstrate the effectiveness of proposed query optimizer which produces best query plan which has a good balance between the cost and latency.

V. REFERENCES

[1] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, "Using the crowd for top-k and group-by queries," in Proc. 16th Int. Conf. Database Theory, 2013, pp. 225–236.

[2] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang, "A hybrid machine-crowdsourcing system for matching web tables," in Proc. IEEE 30th Int. Conf. Data Eng., 2014, pp. 976–987.

[3] 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.

[4] C.-J. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in Proc. 30th Int. Conf. Mach. Language, 2013, vol. 1, pp. 534–542.

[5] H. Park and J. Widom, "Query optimization over crowdsourced data," Proc. VLDB Endowment, vol. 6, no. 10, pp. 781–792, 2013.

[6] A. D. Sharma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, "Crowd-powered find algorithms," in Proc. IEEE 30th Int. Conf. Dta Eng., 2014, pp. 964–975.

[7] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, J.Widom. Deco: declarative crowdsourcing. In CIKM, pages 1203–1212, 2012.

[8] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis,A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In SIGMOD Conference, pages 361–372, 2012.

[9] A. D. Sharma, A. Parameswaran, H. Garcia-Molina, and A. Halevy.Crowd-powered find algorithms. In ICDE Conference, 2014.

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