Design Optimization using Crowdsourcing

Mr. N. Bhaskar
Assistant professor
Department of Information Technology
KCG College of Technology Karapakkam, Chennai

Ms. K. Kalpana
B. Tech Student
Department of Information Technology
KCG College of Technology Karapakkam, Chennai

Ms. M. Divya
B. Tech Student
Department of Information Technology
KCG College of Technology Karapakkam, Chennai

Abstract

Crowdsourcing is the disperse problem solving model in the domain of Data mining that has emerged in recent years. Crowdsourcing is designed to relieve the user from the burden dealing with the crowd. The Motivation of the project is the registered users or any group of audience from social media who are registered can rate the designs from Number of designs provided by the designers and filter by the admin based on different contexts and the filtered rated designs are displayed to the end users or customers. The project is mainly for the particular boutique shop. The project is based on query optimization: cost-based and latency based query optimization.
Keywords- crowdsourcing, data mining, query optimization, cost, latency

I. INTRODUCTION

Crowdsourcing is functional way to address tasks by utilizing hundreds of thousands of ordinary workers (i.e., the crowd). Furthermore, access to crowd resources has been made easier due to public crowdsourcing platforms, such as Amazon Mechanical Turk(AMT), CrowdFlower and Up work . As a result, crowdsourcing has become an active area of research in the recent years. In this paper, we describe cost-based query optimizer and latency based query optimizer.

Our goal is to find the best query plan from different levels of context to answer a query, where “best” means the least estimated monetary cost(how much people pay for crowdsourcing) and latency(how long people wait for results) on different contexts across all possible query plans. The project is supposed to be used by the designers, registered users, end users and any audience those who are all in the area of fashion industry in the crowdsourcing environment.

II. EXISTING SYSTEM

In the existing system, the designs of clothes are displayed in outsourcing environment so the user can access, browse through open source and get the clothes or product without any ratings or different contexts. So cost and latency can be a burden to the users they have to choose from the large set of design collections and get their product from online sellers.

III. DOMAIN DESCRIPTION

Data mining is a process of obtaining raw data and converting it into information for decision making process. Data is collected and analyzed to answer the questions. The collected data is taken for processing or organizing. The organized data is cleaned up to avoid error, inaccuracy of data and duplication. Once the data is cleaned, they are analyzed. The techniques that are used for analysis are called are exploratory data analysis. As soon as the data is analyzed, it can be reported in many formats as required by the user. The tasks performed by data analysis are retrieve values, filter, find, sort, classify, and find anomalies, cluster, associate and correlate.

Data mining is a process of obtaining raw data and converting it into information for decision making process. Data is collected and analyzed to answer the questions. This survey paper reviews some major mining techniques and key challenges. Data mining is a term from computer science. Sometimes it is also called KDD. Data mining is about finding new information in a lot of data. Crowdsourcing is also one of the part in Data mining.

IV. PROPOSED SYSTEM

In the proposed system, the designs of clothes are filtered and displayed to the customers based on different contexts and it is not an outsourcing it is implemented in the crowdsourcing environment. Only registered users can give ratings, based on ratings and contexts users can choose their designs based on their own requirements. The proposed system has advantages

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

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

Fig. 1: user wants to look for designs

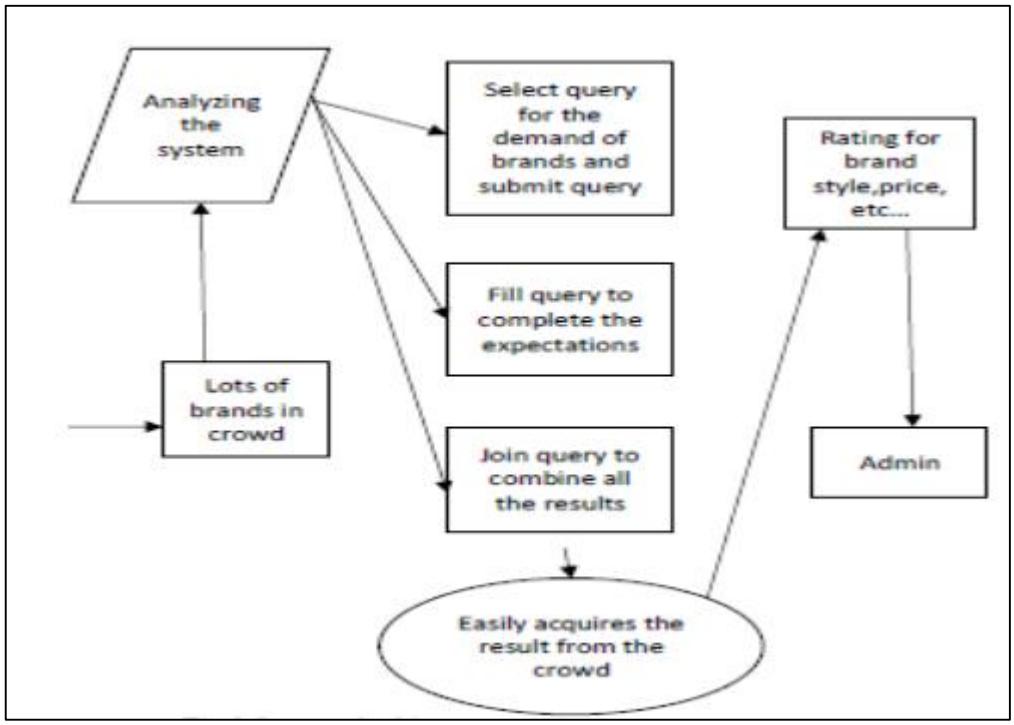


Fig. 2: System Architecture

V. METHODS AND MATERIALS

For the paper, we consider the following three query types.

A. Selection Query

A selection query applies one or more human-recognized selection conditions over the tuples in a single relation. Selection query has several applications in real crowdsourcing platform, such as filtering of data and finding items. For that purpose we are using two algorithms called Crowd source (filtering) and Crowd find (finding).

B. Join Query

A join query leverages human intelligence to combine tuples from two or more relations according to certain join conditions. One typical type of join query is entity resolution that represents the same real-world entity. Other applications include subjective classification and schema matching .suppose if we have different contexts like clothes material, clothes style, and colors of the clothes, size, and patterns of the cloth. If customer wants a one different cloth material but he/she wants the design from the other design from another dress, for this kind of scenario JOIN queries are used we are using NESTED LOOP join query for this purpose.

C. Complex Query

We focus our discussion on latency-bounded cost minimization for complex query optimization. For the case where the latency constraint is not imposed, we can optimize the query plan by traditional databases: heuristic rules like pushing down selections and determining the join ordering and then invoke the above used techniques for optimizing selections and joins operations.

To achieve minimum latency, we are using recent technique and it is a tool called R programming, where R is an integrated suite of software facilities for data manipulation.

VI. TASK SETTINGS

The requester also needs to determine some task settings based on the requirements. There are two main factors that the requester needs to consider.

A. Pricing

The requester needs to price each task. Usually task prices vary from a few cents to several dollars. Usually, high prices can attract more workers, thereby reducing the latency but paying more does not always improve answer quality.

B. Timing

The requester can set time constraints for a given task. For each work or task, the requester can set the certain time bound (e.g., 30 seconds) to answer it, and the worker must answer or give response for the query within the time bound. The requester can also set the expiration time of the tasks, i.e., the maximum time that the tasks will be available to workers in the platform (e.g., 48 hours).

VII. CROWDSOURCING PLATFORMS

A. Amazon Mechanical Turk (AMT)

AMT is a widely used crowdsourcing platform. AMT focuses on micro-tasks, e.g., labeling an image. A requester can group multiple micro-tasks as a Human Intelligence Task (called HIT). The requester can also set some requirements, e.g., the price of a HIT, the time constraint for answering a HIT, the expiration time for a job to be available on AMT, and the qualification test. A requester can build HITs from several different ways.

1) The requester user interface

AMT provides many templates, such as categorization, data collection, sentiment analysis and image tagging. After designing the worker interface, requesters need to upload the task files. The user interface is very easy to use, even for untrained requesters.

2) AMT Command Line Tools (CLT)

AMT predefines a set of commands in CLT, such as loadHITs, getResults, grant- Bonus, blockWorker and so on. Requesters can utilize CLT to easily build HITs by specifying three files: the data file, the task file to specify user interface for the tasks, and the property file to add the title, description, keywords, reward and assignments for the HIT. The CLT is suitable when a requester has a relatively small number of assignments.

3) AMT APIs

There are many APIs in AMT, including the creation of HITs, block/unblock the workers, collection of the finished answers and statistics collection for the requester and workers. There are three steps to use the APIs to publish tasks:

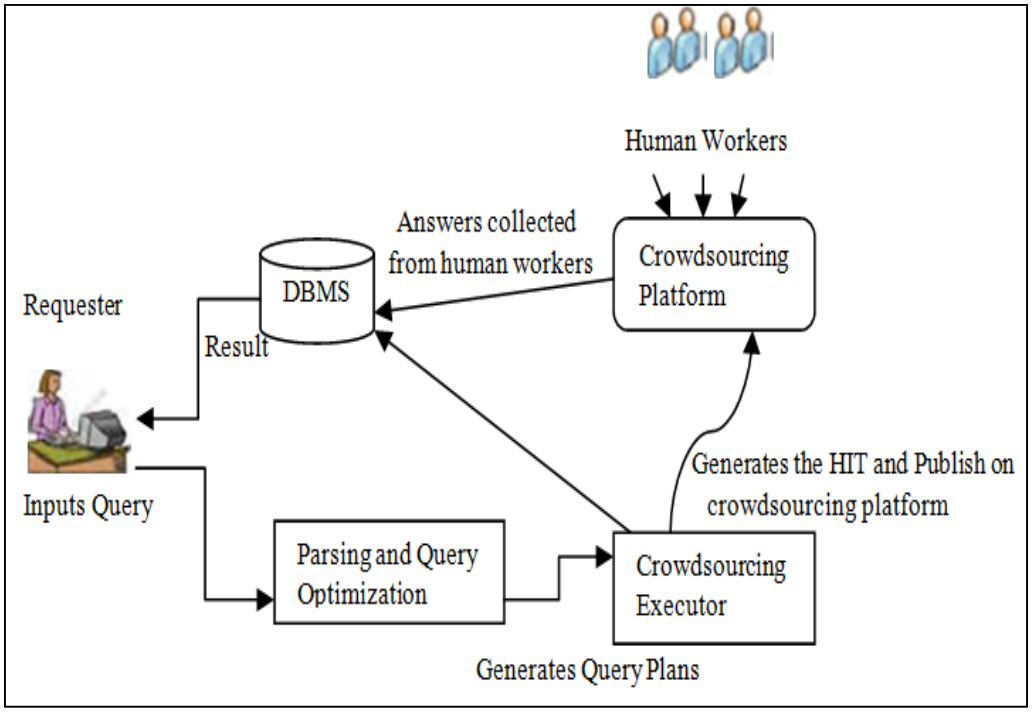
1) download the SDK for a specified language, e.g., python and java;
2) specify the title, description, reward, the content of tasks and the detailed properties for the HIT;
3) Publish the HITs to the platform.

Requesters can build their own server to manage the tasks and embed their tasks into AMT using inner HTML. When a worker requires a task, AMT transforms the requirement to the requester's server and then the requester can decide how to assign tasks to the server. When a worker submits an answer to AMT, AMT also transforms the result to the requester. A worker can browse HITs on AMT. Each HIT has some information, e.g., the description of the task, the price, the keywords, the qualification test if required, and the requester's id. After a worker submits the answers of HITs to the platform, s/he can find the total earnings and the Status of the submitted HITs on the platform.

B. CrowdFlower

CrowdFlower has similar functionalities with AMT, but they still have some differences. First, CrowdFlower has a quality-control component, and it leverages the Gold-Injected method to block low-quality workers. Second, besides publishing the tasks on its own platform, CrowdFlower also publish the tasks on other platforms.

VIII. WORKFLOW OF CROWDSOURCING



IX. EXAMPLE OF CROWDSOURCING SYSTEM

Crowdsourcing system has some following application.

- Voting System: In this type of crowdsourcing system, a user is required to select an answer in given choices. The answer to the most of users select is considered to be accurate.
- Information Sharing System: Website is used to share information between internet users. Some crowdsourcing systems aim at sharing various types of information among the crowd.
- Creative System: In creativity mode, the contribution of human work cannot be replaced by an advanced technology.

X. CROWDSOURCED OPTIMIZATION AND SYSTEMS

There are several crowdsourcing systems that integrate crowdsourcing into relational database management systems (RDBMS) and enable RDBMS to process computer hard queries. The basic workflow of query processing consists of query parser, query plan generation, optimization, and execution. Given a query, a parser is first applied and multiple plans can be generated. Then the query optimization selects the best query plan, and finally, the plan is executed with both machines and the crowd. Existing crowdsourcing database systems focus on query model, query operators, and query optimization techniques. Next we discuss different existing crowdsourced systems.

A. CrowdDB

To utilize the crowdsourced operators, CrowdDB extends SQL and defines a new query language, called Crowd- SQL, that is used to define which table or attribute should be crowdsourced. In query processing, CrowdDB introduces three crowd operators.

1) CrowdProbe

Collect missing information of attributes or new tuples from the crowd. The typical user interface of CrowdProbe is a form with several fields for collecting information from the crowd.

2) CrowdJoin

Implement an index nested-loop join over two tables, where at least one of which is crowdsourced. In particular, the inner relation must be a Crowd table and the user interface is used to crowdsource new tuples of inner relation which can be joined with the tuples in outer relation.

3) CrowdCompare

This operator is designed to implement two functions, Crowd-Equal and Crowd-Order, defined in the CrowdDB's query model. The interface of the operator crowdsources two tuples and leverages the crowd to compare these tuples. Crowd-Equal compares two values and asks the crowd to decide whether they have the same value. Crowd-Order asks the crowd to give an order according to a predefined attribute. CrowdDB also proposes rule-based optimization techniques for processing queries with multiple operators.

B. Qurk

Qurk uses a SQL-based query language with user defined functions (UDFs) to enable crowdsourced data management. To facilitate users to implement the UDFs, Qurk has several pre-defined task templates that can generate the UIs for posting different kinds of tasks to the crowd. Typical crowdsourcing tasks include:

1) Filter: produce tuples that satisfy the conditions specified in the UDF.
2) Sort: rank the input tuples according to the UDFs specified in the order-by clause.
3) Join: compare input tuples and perform join according to the UDF
4) Generative: allow workers to Generate data for multiple fields.

In query processing, Qurk focuses on implementing join and sort.

1) CrowdJoin

Similar to Crowd Qurk also implements a block nested loop join and crowdsources the tuples from two tables for evaluating if they satisfy join conditions. In particular, Qurk studies the techniques for batching multiple comparisons to reduce the cost.

2) CrowdSort

Qurk implements two basic approaches to execute sort. The comparison-based approach solicits the crowd to directly specify the ordering of items. This approach may be expensive for large datasets due to the quadratic comparison. Another task type, rating, is used to reduce the cost using a well-defined interface.

Qurk has two important components for cost optimization: task cache and task model. Task cache maintains the crowdsourced answers from previous tasks, and task model trains a model to predict the results for the tasks based on the data that are already collected from the crowd. So if one task can get necessary information from task cache or task model, it will not be published to the crowdsourcing platform. Once the task cannot get useful information from task cache and task model, it will be pushed to the task complier. The complier generates and publishes the tasks to the crowdsourcing platform. Statistic manager determines the number of tasks, assignment and the cost for each task.

C. Deco

Deco separates the user view and system view. The logical relations are specified by a schema designer and queried by an end-user. Raw schema is stored in the RDBMS and it is invisible to the schema designer and users. Deco focuses on crowdsourcing missing values or new tuples based on the defined fetch rules. Deco designs fetch rules that allow the schema designers to specify how data are collected from the crowd. Given a fetch rule: get the value of attribute A2 given the value of attribute A1, Deco presents the values of attributes in A1 and asks the crowd to give the values of attributes in A2. For instance, given "China! Capital", Deco collects the capital of China from the crowd. In particular, if A1 is empty, the system fetches new values of attributes in A2. As inconsistency may exist in the collected data, Deco can also specify resolution rules such as deduplication and majority voting to resolve inconsistencies in the collected data. Deco also supports other operators, such as Dependent Left Outer Join, Filter and Scan.

Based on the defined operators, given a complicated query, a fundamental query optimization problem is How to find the best query plan to the query, which has the least estimated monetary cost across all possible query plans. To solve the problem, first defines the monetary cost. Considering the fact that the existing data in the database can be leveraged, the cost is formally defined as the new data that needs to be obtained from the crowd. In order to find the best query plan with the minimum cost, there are two problems addressed in.

1) Cost Estimation

How to estimate the cost of a query plan. As a query plan is executed, the database may collect new data from the crowd, which may affect the cost estimation of subsequent processes. By considering this effect, proposes an iterative approach to estimate the cost for a query plan.

2) Optimal Query Plan Generation

Simply enumerating all possible query plans is computationally expensive, and considers to reuse the common sub-plans in order to reduce the redundant computation. Then the best query plan, i.e. with least estimated cost, can be returned.

XI. CONCLUSION

The paper concludes with the optimized designs are displayed to the customers based on different contexts if the user or customers meet their requirements on filtered designs they can choose designs by various contexts and make choose designs by designers. By this paper we can achieve minimum cost and latency.

XII. FUTURE WORK

In future work, we may have implement how the chosen design will be delivered to the customers and what are the security problems while delivering to customers and how to overcome the security issues in this Crowdsource platform.

REFERENCES

[1] 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.
[2] 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.
[3] Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi. CrowdOp: Query Optimization for Declarative Crowdsourcing Systems, in knowledge and data engineering, vol. 27, no.8, august 2015.
[4] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries inProc. 16th Int. Conf. DatabaseTheory, 2013, 225–236.
[5] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In CIKM, pages 1203 – 1212, 2012.
[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In SIGMOD Conference, pages 61–72, 2011.
[7] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In CIDR, pages 211–214, 2011