International Journal of Computer and Communication Technology

Volume 6 | Issue 3 Article 7

July 2015

OPTIMIZED INTENTION FOR CONTINUOUS QUERIES IN THE ACTIVE DATA AGGREGATION NETWORK WITH COST MODEL

INDUMATHI. R
CSE. Dhanalakshmi Srinivasan College of Engineering, Coimbatore, indu_be@yahoo.co.in

OPTIMIZED INTENTION FOR CONTINUOUS QUERIES IN THE ACTIVE DATA AGGREGATION NETWORK WITH COST MODEL

INDUMATHI.R

CSE. Dhanalakshmi Srinivasan College of Engineering, Coimbatore.
Email:indu_be@yahoo.co.in

Abstract- Within an RDBMS streams of changes to the data and reporting when the result of a query defined over the data changes. These queries are referred to as the continuous queries since they continually produce results whenever new data arrives or existing data changes. To make online decision we require monitoring the continuous queries. Here the aim is to introduce the low-cost, scalable technique to answer continuous aggregation queries using a network of aggregators of dynamic data items. There is significant work in systems that can efficiently deliver the relevant updates automatically. And also to provide for getting the optimal set of sub queries with their incoherency bounds which satisfies client query's coherency requirement with least number of refresh messages sent from aggregators to the client. For optimal query execution divide the query into sub-queries and evaluate each sub-query at a judiciously chosen data aggregator. The main purpose is to response the client with the least number of tasks with the help of random query selection. Random query selection means for the user submitted query the relevant queries, sub queries are generated. A query cost model which can be used to estimate the number of messages required to satisfy the client specified incoherency bound.

Keywords-*Continous Queries, Query Planning, Random Query Selection, Cost Model,*

1. INTRODUCTION

Data mining is the process of finding correlations or patterns among dozens of fields in large relational databases. While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Data mining consists of five major elements: (1)Extract, transform, and load transaction data onto the data warehouse system. (2)Store and manage the data in a multidimensional database system. (3) Provide data access to business analysts and information technology professionals.(4)Analyze the data by application software. (5)Present the data in a useful format, such as a graph or table.

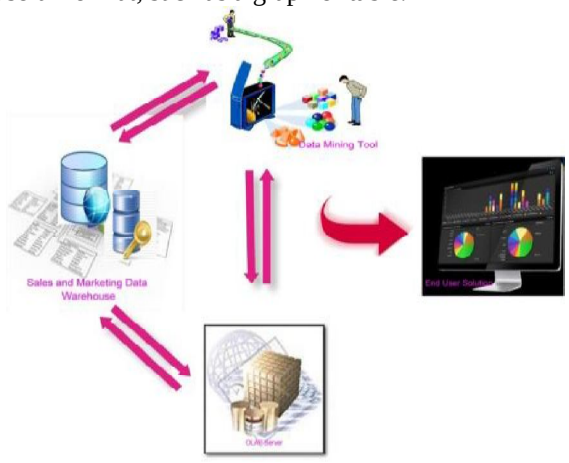

Figure 1 - Integrated Data Mining Architecture

The figure1 represents a fundamental shift from conventional decision support systems. Rather

than simply delivering data to the end user through query and reporting software, the Advanced Analysis Server applies users' business models directly to the warehouse and returns a proactive analysis of the most relevant information. These results enhance the metadata in the OLAP Server by providing a dynamic metadata layer that represents a distilled view of the data. Reporting, visualization, and other analysis tools can then be applied to plan future actions and confirm the impact of those plans.

The more powerful the data mining queries, the greater the utility of the information being gleaned from the data, and the greater the pressure to increase the amount of data being collected and maintained, which increases the pressure for faster, more powerful data mining queries. This increases pressure for larger, faster systems, which are more expensive. All database systems must be able to respond to requests for information from the user—i.e. process queries. Obtaining the desired information from a database system in a predictable and reliable fashion is the scientific art of *Query Processing*. Getting these results back in a timely manner deals with the technique of *Query Optimization.* A database query is the vehicle for instructing a DBMS to update or retrieve specific data to/from the physically stored medium. The actual updating and retrieval of data is performed through various "low-level" operations. There are three phases that a query passes through during the DBMS' processing of that query: Parsing and translation, Optimization and Evaluation. Most queries submitted to a DBMS are in a high-level language such as SQL. During the parsing and translation stage, the human

readable form of the query is translated into forms usable by the DBMS. These can be in the forms of a relational algebra expression, query tree and query graph.

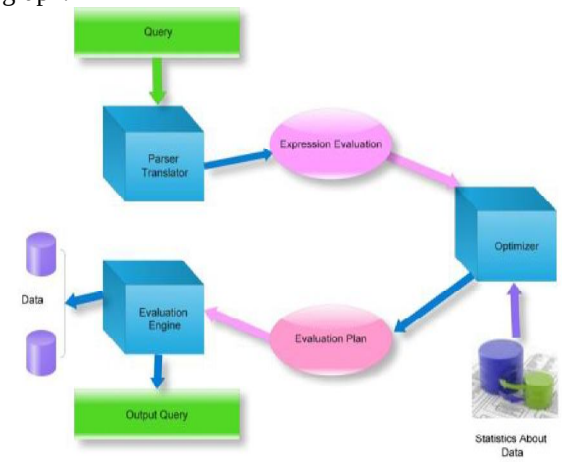

Figure2: Query Processing

2. RELATED WORKS

Data Stream Management Systems (DSMSs) were developed to be at the heart of every monitoring application. Monitoring applications typically register hundreds of Continuous Queries (CQs) in DSMSs in order to continuously process unbounded data streams to detect events of interest. DSMSs must be designed to efficiently handle unbounded streams with large volumes of data and large numbers of CQs, i.e., exhibit scalability[6]. This need for scalability means that the underlying processing techniques a DSMS adopts should be optimized for high throughput. These systems are built around three design principles that aid in the real-time. Querying of complex data sources: query interfaces tailored to the application's specific data types, optimized data collection processes, and allowing queries to provide feedback to the collection process.

Various mechanisms for efficiently maintaining incoherency bounded aggregation queries over continuously changing data items are proposed in the literature [5,7,9]. This paper work distinguishes itself by being sub-query based evaluation to minimize number of refreshes. In [7], authors propose using data filters at the sources; instead here assign incoherency bounds to sub-queries which reduce the number of refreshes for query evaluation. In proxy-based caching approaches store content at various locations outside the site infrastructure and can improve Web site performance by reducing content generation delays, firewall processing delays, and bandwidth requirements[2]. In cache at the page level, which does not guarantee that correct pages axe served and provides very limited reusability. The benefits of existing proxy-based and back end caching techniques, without their respective limitations.

In traditional database systems optimize for performance on one-shot query processing, emerging large-scale monitoring applications require continuous tracking of complex data analysis queries over collections of physically-distributed streams. In readily incorporates several complex analysis queries thus giving the first known low-overhead tracking solution for such queries in the distributed-streams model[8]. The optimized for tracking high-speed streams, and result in very low processing and communication costs, and significant savings over naive updating schemes.

The method of executing queries over dynamic data dissemination network is practical since it can be implemented using a mechanism similar to URL-rewriting [4] in CDNs. Just like in a CDN, the client sends its query to the central site. For getting appropriate aggregators (edge nodes) to answer the client query (web page), the central site has to first determine which data aggregators have the data items required for the client query. If the client query cannot be answered by a single data aggregator, the query is divided into sub-queries (fragments) and each sub-query is assigned to a single data aggregator.

3. PROPOSED SYSTEM

The Web suffers from performance and scalability issues. Content distribution networks (CDNs) solved the problem for static content using caches at the edge nodes of the networks. CDNs continue to evolve to serve more and more dynamic applications. A dynamically generated web page is usually assembled using a number of static or dynamically generated fragments. The static fragments are served from the local caches whereas dynamic fragments are created either by using the cached data or by fetching the data items from the origin data sources. One important question for satisfying client requests through a network of nodes is how to select the best node(s) to satisfy the request.

The content requested proximity to the client and load on the nodes are the parameters generally used to select the appropriate node. In dynamic CDNs, while selecting the node(s) to satisfy the client request, the central site (top-level CDN node) has to ensure that page/data served meets client’s coherency requirements also. Such dynamic data dissemination networks can be used to disseminate data such as stock quotes, temperature data from sensors, trafficinformation, and network monitoring data. In this paper we propose a method to efficiently answer aggregation queries involving such data items.

Incoherency of a data item at a given node is defined as the difference in value of the data item at the data source and the value at that node.

Incoherency bound, a data aggregator gets data updates from the data source or some higher level data aggregator so that the data incoherency is not more than the data incoherency bound. In a hierarchical data dissemination network a higher level aggregator guarantees a tighter incoherency bound compared to a lower level aggregator. Thus data refreshes are pushed from the data sources to the clients through the network of aggregators.

The problem of selecting the optimal plan ensure that each data item for a client query is disseminated by one and only one data aggregator. Although a query can be divided in such a way that a single data item is served by multiple; but in doing so the same data item needs to be processed at multiple aggregators, increasing the unnecessary processing load. By dividing the client query into disjoint sub-queries it ensure that a data item update is processed only once for each query.

The query incoherency bound needs to be divided among sub query incoherency bounds such that, besides satisfying the client coherency requirements, the chosen is capable of satisfying the allocated sub-query incoherency bound. Here it can prove that the number of refreshes depends on the division of the query incoherency bounds among sub-query incoherency bounds. The individual data items involved. The data dissemination cost is dependent on data dynamics and incoherency bound associated with the data. We model the data dynamics using a data synopsis model, and the effect of the incoherency bound using an incoherency bound model. These two models are combined to get the estimate of the data dissemination cost.

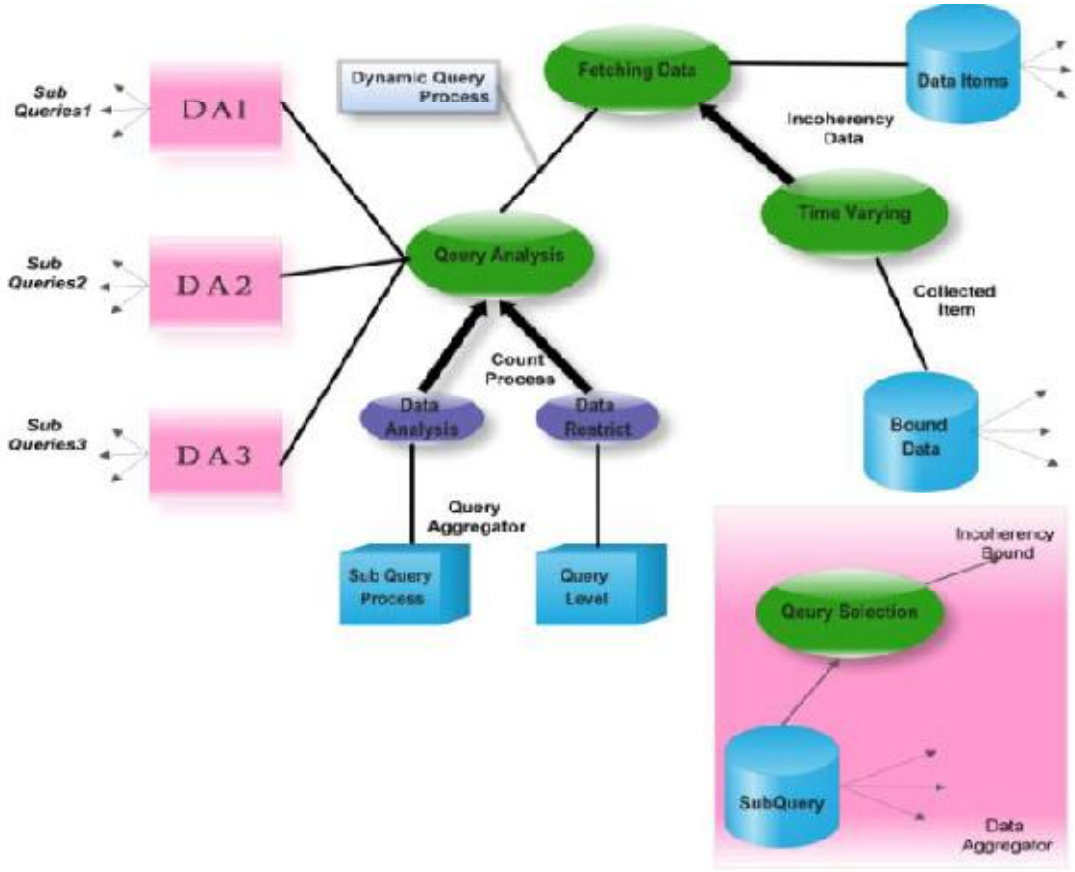

Figure 3: Architectural view of the system

4. ARCHITECTURE OF THE SYSYTEM

There are several types of web services available now a day, the services are provided by the different kind of service provider. The client can view or access the number of services from the service provider. And they can search as per their want to search.

For example they can login to the Gmail, twitter or some other else. And the user can give the numerous queries they want to be search. The service provider keeps the distinct queries submitted by the user. And also they keep the track of pages viewed. The service provider can update the query either by static or dynamic. And also they update the pages viewed by the users. For that submitted query the answers cannot get by the single aggregator. The submitted query were split and searched by the number of aggregator. The results from the distinct aggregator were collected and response to the client. The above concept purpose is to response the client with the least number of tasks with the help of random query selection. Random query selection

means for the user submitted query the relevant queries, sub queries are generated. In query cost execution, the costs of refreshes are updated.

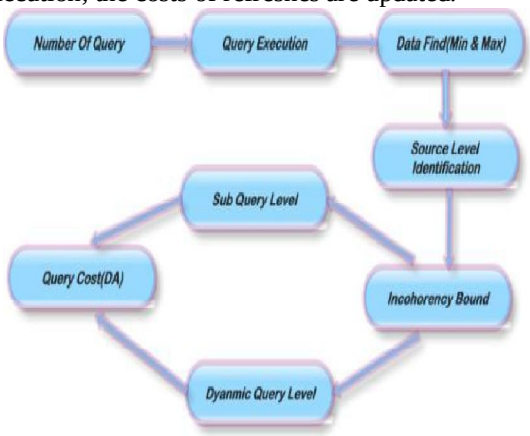

Figure 4: Evaluation Of query

The algorithm is to execute the continuous query and to calculate the cost (i,e) to estimate the number of refresh messages sent from aggregators to the client. The outline of Greedy algorithm for delivering the queries is follow:

Algorithm **GreadyBFS**
Get input data
Complete a path tree for each origin query by doing the following.
Initialize the query table
Step 1:Create the list of tentative query labels
Update the node table
Step 2: Finalize the current query
Step 3: Find the next current query
if there is a valid current query then Go to Step 1
Otherwise, done
Output the completed query table

First, get a set of maximal sub-queries corresponding to all the data aggregator in the network. The maximal sub-query for a data aggregator is defined as the largest part of the query which can be disseminated by the DA.[3] For the given client query and the relation consisting of data aggregators, data-items, and the data incoherency bounds ,maximal sub queries can be obtained for each data aggregator by forming sub-query involving all data items in the intersection of query data items and those disseminated by the DA.

For executing an incoherency bounded continuous query, a query plan is required which includes the set of sub-queries, their individual incoherency bounds and data aggregators which can execute these sub-queries[3]. The need to find the optimal query execution plan which satisfies client coherency requirement with the least number of refreshes. The mechanism to:

Task 1: Divide the aggregation query into sub-queries; and

Task 2: Allocate the query incoherency bound among them.
while satisfying the following conditions :

condition 1. Query incoherency bound is satisfied.

condition 2. The chosen DA should be able to provide all the data items appearing in the sub-query assigned to it.

condition 3. Data incoherency bounds at the chosen DA should be such that the sub-query

Algorithm for **QueryExecutionWithCalculatingCost:**
Begin
If have_input_data <> 1 Then
get_query
get_links
ReDim query _table(num_query, 4)
ReDim all_origins_ query _table(num_ query *
num_zones, 5)
first_origin = 0
End If
have_input_data = 1
calc_costs
For node_count = 1 To num_ query
If nodes(node_count, 2) = "z" Then
origin_zone = node_count
initialize_node_table
debug_output
origin_done = "no"
Do Until origin_done = "yes"
find_neighboring_nodes
update_node_table
debug_output
finalize_current_node
debug_output
find_current_node
Loop
output_origin_node_table
if first_origin = 0 then
first_origin = 1
End if
End If
Next
End

5. COST MODEL FOR AGGREGATION QUERIES

Consider a query over two data items P and Q with weight w0 and w1 and now is to estimate the dissemination cost: The query sumdiff will be:

$$R_{data} = w_p R_p + w_q R_q = w_p \sum | p_i - p_{i-1} | + w_q \sum | q_i - q_{i-1} |$$

The aggregator uses the information that client is, P & Q is an individual query value. User Compare Query Data Analysis[3],

$$R_{query} = \sum | w_p (p_i - p_{i-1}) + w_q (q_i - q_{i-1}) |$$

Thus to represent the relationship between *Rquery* and *sumdiff* values of the individual data items using a correlation measure associated with the pair of data items.

$$R_{query}^2 \propto (w_p^2 R_p^2 + w_q^2 R_q^2 + 2\rho w_p R_p w_q R_q)$$

ρ=Correlation Measure.

$$\rho = \frac{\sum (p_i - p_{i-1})(q_i - q_{i-1})}{\sqrt{\sum (p_i - p_{i-1})^2}\sqrt{\sum (q_i - q_{i-1})^2}}$$

Compare Query between P & Q,

$$R_{query}^2 = \frac{(w_p^2 R_p^2 + w_q^2 R_q^2 + 2\rho w_p R_p w_q R_q)}{(w_p^2 + w_q^2 + 2\rho w_p w_q)}$$

The value of the normalizing factor R_{query}:

$$1/\sqrt{w_p^2 + w_p^2 + 2\rho w_p w_q}\ .$$

The value of the incoherency bound(C) has to be adjusted by the same factor. The query cost model we first summarize the model to estimate the number of refreshes required to disseminate a data item at certain incoherency bound.The number of data refreshes is inversely proportional to the square of the incoherency bound (1/*C2*).The data dynamics was modeled as a random walk process.

$$Data\ dissemination\ cost \propto 1/C^2$$

The number of update messages for a data item is likely to be higher if the data item changes more in a given time window. The estimated dissemination cost for data item *S*, disseminated with an incoherency bound *C*, is proportional to *Rs/C2*. Using this result the query cost model is developed.

6. SUMMARY OF THE PERFORMANCE

Following features of the query planning algorithm improve performance:

_ Dividing the query into sub-queries and executing them at specifically chosen data aggregators.

_ Deciding the query plan using data *sumdiff* based mechanism specifically by maximizing sub-query gains.

_ Including more dynamic data as part of a larger sub-query.

7. CONCLUSION

This paper presents a cost based approach to minimize the number of refreshes required to execute an incoherency bounded continuous query. For optimal execution it is to divide the query into sub-queries and evaluate each sub-query at a chosen aggregator. In this method, the query execution can be implemented using schemes similar to that used in CDNs. The query cost model can also be used for other purposes such as load balancing various aggregators, optimal query execution plan at an aggregator node, etc.

REFERENCES

[1]. "SINA: Scalable Incremental Processing of Continuous Queries in Spatiotemporal Databases" - Mohamed F. Mokbel, Xiaopeng Xiong and Walid G. Aref , Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398

[2].D. VanderMeer, A. Datta, K. Dutta, H. Thomas and K.Ramamritham. Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web. ACM Transactions on Database Systems (TODS) Vol. 29, June 2004.

[3]. Rajeev Gupta and Krithi Ramamrithram, Query Planning for continuous Aggregation Queries over a Network of Data Aggregators, IEEE transaction on knowledge and data mining Vol 24,Issue:6

[4] S. Rangarajan, S. Mukerjee and P. Rodriguez. User Specific Request Redirection in a Content Delivery Network, 8th Intl. Workshop on Web Content Caching and Distribution (IWCW), 2003.

[5]. R. Gupta, A. Puri, and K. Ramamritham. Executing Incoherency Bounded Continuous Queries at Web Data Aggregators. WWW 2005.

[6].Mohamed A.Sharaf Alexandros Labrinnidis, Panos k. Chrysanthis,"Scheduling Continuous Queries In Data Stream Management Systems" in VLDB Endowment.

[7]. C. Olston, J. Jiang, and J. Widom. Adaptive Filter for Continuous Queries over Distributed Data Streams. SIGMOD 2003.

[8]. G. Cormode and M. Garofalakis. Sketching Streams through the Net: Distributed Approximate Query Tracking. VLDB 2005.

[9]. S. Zhu and C. Ravishankar. Stochastic Consistency and Scalable Pull-Based Caching for Erratic Data Sources. VLDB 2004.