ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering &Technology (IJARCET)
Volume 2, Issue 8, August 2013

QUERY PLANNING FOR CONTINUOUS AGGREGATION QUERIES USING DATA AGGREGATORS

A. SATEESH[1], D. ANIL[2], M. KIRANKUMAR[3]

ABSTRACT:

Continuous aggregation queries are used to monitor the changes in data with time varying for online decision making. For continuous queries low cost and scalable techniques used a network of aggregators. Individual node cannot by itself determine its inclusion in the query result for this a different algorithmic challenges from aggregate and selection queries are presented. At specific coherencies each data item can serve for a set of data aggregators. Technique involves disseminating query into sub query and sub queries are executed on the chosen data aggregators. We build a query cost model which can be used to estimate the number of refresh messages which is required to satisfy the client specified incoherency bound. Performance results shows that by our method the query can be executed using less than one third the messages required for existing schemes. Our adaptive strategy employs distributed decisions made by the distributed servers independently based on localized statistics collected by each server at runtime. When comparatively static environment, propose two motionless tree construction algorithms relying on apriori system statistics. These static trees can also be used as early trees in a dynamic environment and apply our schemes to both single- and multi object distribution. Our extensive performance study illustrate that the adaptive mechanisms.

Index Terms— **Algorithms, Continuous Queries, Data Dissemination, Distributed Query Processing, Coherency, Performance**

I. INTRODUCTION

Applications such as auctions, Personal portfolio valuations for financial decisions, weather prediction website, espn cricinfo website, Stock Exchange Website, etc., make the extensive use of active data. For such appliances, data from one or more independent data sources may be aggregated to determine the data with less no of refreshes. Given the increasing number of such applications that make use of highly dynamic data, there is important concern in systems that can efficiently deliver the relevant updates mechanically. Consider a user who wants to road a portfolio of stocks in different accounts. Stock data values from possibly unlike sources are required to be aggregated to satisfy user's requirement. These aggregation queries are long running queries as data are continuously changing and the user is interested in notifications when confident conditions hold. Thus, responses to these queries are revived continuously. In these permanent query applications, uses are likely to tolerate some inaccuracy in the results i.e., the exact data values at the corresponding data sources need not be reported as long as the query results satisfy user specified accuracy requirements. Continuous queries are very essential in data mining as it proved to be very efficient in tracking data are space and time bounded. Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. Naturally a user desires to acquire the value of some aggregation function over disseminated data items, for example, to know value of portfolio for a client; or the AVG of temperatures sensed by a situate of sensors. In these queries a client specifies a coherency requirement as part of the query. Continuous Aggregated Queries enable members of an organization to cooperate by sharing their resources (both storage and computational) to host (compressed) data and perform aggregate queries on them, while protecting their autonomy. A framework with these properties can be useful in different application contexts. For instance, assume the case of a worldwide virtual organization with

users interested in biological data, as well as the case of a real association on an activity network. In both cases, even users who are not continuously interested in performing data analysis can make a part of their resources available for supporting analysis tasks needed by others, if their own ability of performing local asks is conserved. This is equivalent to the idea on which several popular applications for public resource computing are based. Members of a worldwide society offer their CPU, when it is inactive, to scrutinize radio telescope readings in search of nonrandom patterns. In order to make participants really independent, they should be forced no constraint on storage and computational resources to be shared on the reliability of their network connection. For answering the multi data aggregation query in circumstances, there are three alternatives for the client to get the query outcome. First, the client may get the data items d1, d2, and d3 separately. The query incoherency bound can be divided among data items in various ways ensuring that query incoherency is below the incoherency bound. Here show that getting data items independently is a costly option.

This strategy ignores the fact that the client is interested only in the aggregated value of the data items and various aggregators can disseminate more than one data item. Second, if a single Data Aggregator (DA) can disseminate all three data items required to answer the client query, the DA can construct a compound data item corresponding to the client query and disseminate the result to the client so that the query incoherency bound is not violated. It is noticeable that if we get the query result from a single DA, the number of refreshes will be (as data item updates may cancel out each other, thereby maintaining the query results within the incoherency bound). Further, even if an aggregator can refresh all the data items, it may not be able to convince the query coherency necessities. In such cases the query has to be executed with data from multiple aggregators.

II. RELATED WORKS

Value of a continuous weighted additive aggregation query, at time t, can be calculated as; where V_q is the value of a client query q involving n_q data items with the weight of the I th data item being Such a query encompasses SQL aggregation operators SUM and AVG besides general weighted aggregation queries such as portfolio queries, connecting aggregation of stock prices, weighted with number of distributes of stocks in the portfolio. Suppose the result for the query given by (1) needs to be continuously provided to a user at the query incoherency bound C_q. Then, the dissemination network has to ensure that whenever data values at sources change such that query incoherency bound is desecrated, the updated value should be refreshed to the client. If the network of aggregators can make sure that the ith data item has incoherency bound C_q, then the following condition ensures that the query incoherency bound Cq is satisfied X The client specified query incoherency bound needs to be translated into incoherency bounds for individual data items or sub queries such that (3) is satisfied. It should be noted that is a sufficient condition for satisfying the query incoherency bound but not essential. This way of converting the query incoherency bound into the sub query incoherency bounds is required if data are transferred between various nodes using only push-based mechanism.

We need a method for
a) Optimally dividing a client query into sub queries
b) Transfer incoherency bounds to them
c) The derived sub queries can be executed at chosen DAs
d) Total query execution cost, in terms of figure of refreshes to the client is minimizing.

The problem of choosing sub queries while minimizing query execution cost is an NP-hard problem. We give efficient algorithms to choose the set of sub queries and their corresponding incoherency bounds for a given client query. In difference, all related work in this area, propose getting individual data items from the aggregators which leads to large number of refreshes. For solving the above problem of best feasible dividing the client query into sub queries, we first need a

method to estimate the query execution cost for various alternative options. A method for calculate approximately the query execution cost is another important contribution. As we divide the client query into sub queries such that each Sub query gets executed at different aggregator nodes, the query execution cost (i.e., number of refreshes) is the sum of the execution costs of its constituent sub queries. The model of the sub query execution cost as a function of dissemination costs of the being data items involved. The data distribution cost is dependent on data dynamics and the incoherency bound associated with the data. We model the data dynamics using a data dynamics model, and the effect of the incoherency bound using an incoherency bounce model. These two models are combined to get the estimate of the data dissemination cost.

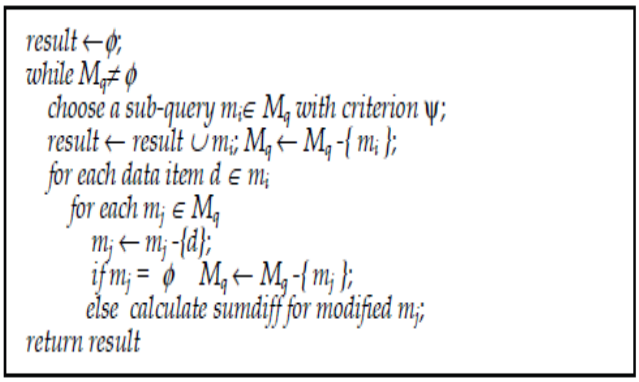


Fig 1: Greedy method for query plan selection

III. DATA DISSEMINATION COST MODEL

To estimate the number of refreshes required to disseminate a data item while maintaining a certain incoherency bound. There are two primary factors disturbing the number of messages that are needed to maintain the coherency constraint:

1) The coherency requirement itself and
2) Dynamics of the data.

A. Incoherency Bound Model

The number of dissemination messages will be proportional to the probability of greater than C for data value v(t) at the source/aggregator and u(t) at the client, at time t. A data item can be modeled as a discrete time random process where each step is correlated with its earlier step. In a push-based distribution, a data source can follow the following schemes:

a. Data source move forward the data value whenever it differs from the last pushed value by an amount more than C.

b. Client estimates data value based on server specified parameters. The source pushes the new data value whenever it differs from the (client). Estimated value by an amount more than C

In both these cases, value at the source can be modeled as a Random process with average as the value known at the client. In case 2, the client and the server estimate the data value as the mean of the modeled random process, whereas in case 1 deviation from the last pushed value can be modeled as zero mean process. Using Chebyshev's inequality Thus, we hypothesize that the number of data refresh messages is inversely proportional to the square of the incoherency bound. A similar result was reported in where data dynamics were modeled as random walks. Validating the analytical model to corroborate the above analytical result we simulated data sources by reading values from the sensor and stock data traces, at periodic instances. For these experiments, every data rate at the first indicate is sent to the client. Data sources maintain last sent value for each client. The sources read new value from the trace and send the value to its clients if and only if not sending it will violate the client's incoherency bound C. For each data item the incoherency bound was varied and refresh messages, to ensure that incoherency bound, were counted. Fig. 1 shows the curves for the number of push messages, for four representative share price data items, as their corresponding incoherency bounds, and hence 1 is C_2, are varied. Besides validating the analytical model, these results give one important insight into the distribution mechanism. As the incoherency bound reduces, the number of messages

increases as per analytical model, but there is a saturation effect for very low values of the incoherency bound (i.e., right part of the curve). This is due to the fact that the data items have limited number of discrete changes in the value. For example, if the sensitivity of a temperature sensor is one degree then number of dissemination messages will not increase even if incoherency bound is decreased below one degree.

B. Data Dynamics Model

Two possible options to model data dynamics, as a first option, the data dynamics can be quantified based on standard deviation of the data item values. Suppose both data items are disseminated with an incoherency bound of 3. It can be seen that the number of messages required for maintaining the incoherency bound will be 7 and 1 for data items d1 and d2, respectively, whereas both data items have the same standard deviation. Thus, we need a measure which captures data changes along with its temporal properties. This motivates us to examine the second measure. As a second option we considered Fast Fourier Trans- form (FFT) which is used in the digital signal processing domain to characterize a digital signal. FFT captures number of changes in data value, amount of changes, and their timings. Thus, FFT can be used to model data dynamics but it has a problem. To estimate the number of refreshes required to disseminate a data item we need a function over FFT coefficients which can return a scalar value. The number of FFT coefficients can be as high as the number of changes in the data value. Among FFT coefficients, 0th order coefficient identifies average value of the data item, whereas higher order coefficients represent transient changes in the value of data item. We hypothesize that the cost of data dissemination for a data item can be approximated by a function of the first FFT coefficient. Specifically, the cost of data dissemination for a data item will be proportional to data sum diff defined as where si and si1 are the sampled values of a data item S at ith time instances (i.e., consecutive ticks). In practice, sum diff value for a data item can be calculated at the data source by taking running average of difference between data values for consecutive ticks. For our experiments, we calculated the sum diff values using exponential window moving average with each window having 100 samples and giving 30 percent weight to the most recent window.

Validating the hypothesis:

We did simulations with different stocks being disseminated with incoherency bound values of $0:001, 0.01, and 0.1. This range is 0.1 to 10 times the average standard deviation of the share price values. Number of refresh messages is plotted with data sum diff (in $) the linear relationship appears to exist for all incoherency bound values. To quantify the measure of linearity we used Pearson product moment correlation coefficient (PPMCC), a widely used measure of association, measuring the degree of linearity between two variables. It is calculated by summing up the products of the deviations of the data item values from their mean. PPMCC varies between 1 and 1 with higher (absolute) values signifying that data points can be considered linear with more confidence. For three values of incoherency bounds 0.001, 0.01, and 0.1; PPMCC values were 0.94, 0.96, and 0.90, respectively, i.e., average deviation from linearity was in the range of 5 percent for low values of C and 10 percent for high values of C. Thus, we can conclude that, for lower values of the incoherency bounds, linear relation- ship between data sum diff and the number of refresh messages can be assumed with more confidence.

IV. QUERY PLANNING

A query plan is an ordered set of steps used to access or modify information in a SQL relational database management system. This is a specific case of the relational model concept of access plans. Since Oracle is declarative, there are typically a large number of alternative ways to execute a given query, with widely varying performance. When a query is submitted to the database, the query optimizer evaluates some of the different, correct possible plans for executing the query and returns what it considers the best alternative. Thus to get a query plan we need to perform following tasks.

1. Determining sub queries: For the client query get sub queries for each data aggregator.
2. Dividing incoherency bound: Divide the query incoherency bound among sub queries to get the value of sub query.

Optimization objective:

Number of refresh messages is minimized. For a sub query the estimated number of refresh messages is given by the ratio of sum diff of the sub query and the incoherency bound assigned to it and the proportionality factor k. Thus the total no of refresh messages is estimated as the summation of the ratio of the sub query of a given query and in coherency bound associated to it.

Constraint1: q_k is executable at ak: Each DA has the data items required to execute the sub query allocated to it, i.e., for each data item d_q k_i required for the sub query

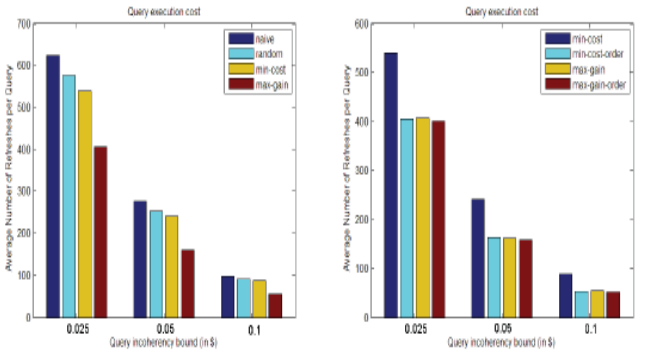
Constraint2: Query incoherency bound is satisfied: Query incoherency should be less than or equal to the query incoherency bound. For additive aggregation queries, value of the client query is the sum of sub query values. As different sub queries are disseminated by different data aggregators, we need to ensure that sum of sub query incoherencies is less than or equal to the query incoherency bound.

Constraint3: Sub query incoherency bound is satisfied: Data incoherency bounds at ak should be such that the sub query incoherency bound can be satisfied at that DA. The tightest incoherency bound, which the data aggregator ak can satisfy for the given sub query q_k. For satisfying this constraint we ensure the following is the outline of our approach for solving this constraint Optimization problem we prove that determining sub queries while minimizing Zq, as given by is NPhard.

If the set of sub queries is already given, sub query incoherency bounds can be optimally determined to minimize. As optimally dividing the query into sub queries is NP-hard and there is no known approximation algorithm, in, we present two heuristics for determining sub queries while satisfying as many constraints as possible. Then, we present variation of the two heuristics for ensuring that sub query incoherency bound is satisfied. In particular, to get a solution of the query planning problem, the heuristics presented in are used for determining sub queries. Then, using the set of sub queries, the method outlined in is used for dividing incoherency bound.

V. PERFORMANCE EVALUATION

For performance evaluation we simulated a network of data aggregators of 200 stock data items over 100 aggregator nodes such that each aggregator can disseminate combinations of 25 to 50 data items. Data items were assigned to different aggregators using zipf distribution skew assuming that some popular data items will be disseminated by more DAs. Data incoherency bounds, for various aggregator data items, were chosen uniformly between $0:005 and 0.02. We created 500 portfolio queries such that each query has 10 to 25 randomly (using zipf distribution with the same default skew) selected data items with weights varying between 2 and 10.



(a) Comparison of algorithm (b) Effect of *data dynamics order* on performance

Fig.2: Performance on MAX queries

These queries were executed with incoherency bounds between 1.0 and 3.0 (i.e., 0.02-0.07 percent of the query value). Although here we present results for stock traces (man-made data), similar results were obtained for sensor traces (natural data) as well. In the first set of experiments,

we kept data incoherency bounds at the data aggregators very low so that query can be ensured while keeping default value of as 0.

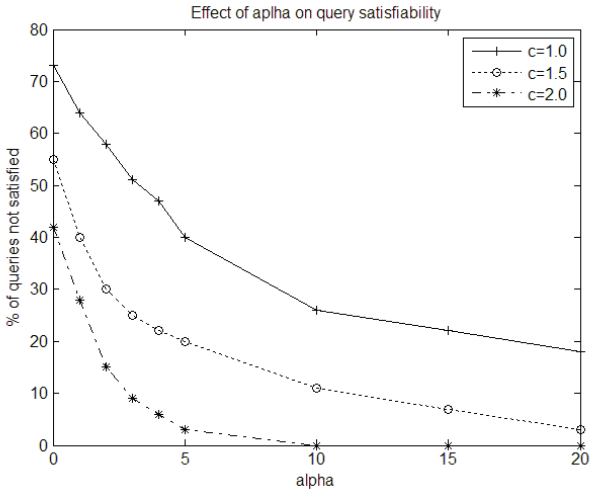


Fig 3: Effect of α on query satisfiability

VI. CONCLUSION

Here presents a cost-based approach to minimize the number of refreshes required to execute an incoherency bounded uninterrupted query. We assume the existence of a network of data aggregators, where each DA is accomplished of disseminating a set of data items at their pre specified incoherency bounds. We developed an important measure for data dynamics in the form of sum diff which is a more appropriate measure compared to the widely used standard deviation based measures. For optimal query implementation we divide the query into sub queries and evaluate each sub query at a judiciously chosen data aggregator.

VII. REFERENCES

[1] "Query Cost Model Validation for Sensor Data," www.cse.iitb.ac.in/~grajeev/sumdiff/RaviVijay _BTP06. pdf, 2011.

[2] . Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M., Chandra, T.D.: Matching events in a content-based subscription system. In: PODC, pp. 53–61 (1999)

[3] Y. Zhou, B. Chin Ooi, and K.-L. Tan, "Disseminating Streaming Data in a Dynamic Environment: An Adaptive and Cost Based Approach," The Int'l J. Very Large Data Bases, vol. 17, pp. 1465-1483, 2008.

[4] S. Rangarajan, S. Mukerjee, and P. Rodriguez, "UserSpecific Request Redirection in a Content Delivery Network," Proc. Eighth Int'l Workshop Web Content Caching and Distribution (IWCW), 2003

[5] S. Shah, K. Ramamritham, and P. Shenoy, "Maintaining Coher- ency of Dynamic Data in Cooperating Repositories," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB), 2002.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press and McGraw- Hill 2001.