International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 12, December - 2013

Optimized Query Planning for Continuous Aggregation Queries Using Query Cost Model

Pranali Sardare[1], Pratap Singh Patwal[2]
[1]*Department of Information Technology, IET, Alwar, Rajasthan, India*
[2]*Department of Information Technology, IET, Alwar, Rajasthan, India*

Abstract

Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. In these queries, at specifies coherency a client requirement as part of the query. For continuous queries, network of data aggregators a low cost and scalable technique used. Individual node cannot be determine its inclusion by itself 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. In this technique disseminating query into sub query and sub queries are executed on the chosen data aggregator are involves. We build a query cost model, which can be used to estimate the number of refresh messages which is required to satisfy client specified incoherency bound. Each data aggregator serve a set of data items at specific coherencies. Performance show that, our query cost model can be executed using less than one third the number of refresh messages required by existing schemes in result. Distributed decisions made by the distributed servers independently based on localized statistics collected by each server at runtime by our adaptive strategy.

1. Introduction

Applications such as personal portfolio valuations for financial decisions, weather prediction website, and stock market exchange Website such as BSE and NSE, student evaluation, auctions, etc. make the extensive use of active data. For such appliances data from one or more independent sources may be aggregated to determine the data with less number of refreshes. The given increasing number of such applications that make use of highly dynamic data is important concern in systems that can efficiently updates mechanically deliver the relevant. Consider a user who wants to student evaluation of performance stock in college in different subjects, attendance, marks, and extracurricular activity. User requirements are required to satisfy the stock data values from possibly unlike sources in database. The user is interested in notifications when confident conditions hold while aggregation queries are long running queries as data are changing continuously. Thus, responses to these queries are continuously revived for processing. 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 the value of portfolio for a client; or the AVG of temperatures sensed by a situate of sensors [1]. In these queries, a client specifies a coherency requirement as part of the query. While protecting their autonomy, a 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. A framework with these properties of continuous queries can be useful in different application contexts. Assume the case of a worldwide virtual organization users interested in real association on a network activity as well as case of biological data. In both cases, even users who are not continuously interested in performing data analysis can make a part of their resources available for supporting task analysis needed by others, if their own ability performing their local task is conserved. This is equivalent to the idea on public computing resources are based on several popular applications. Members of a worldwide society are using their CPU when it is inactive to scrutinize radio telescope readings in search of non random patterns. In order to make participants independent, computational resources to be shared on the reliability of their network connection with no forced on storage constraint. For answering multi data aggregation queries in

circumstances, three alternatives are for the client to get the query result. At first, the client may get the data items d1, d2 and d3 separately [2]. The query incoherency bound can be divided among data items in various ways ensuring that query incoherency is below the incoherency bound.

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. If a single Data Aggregator (DA) can disseminate all three data items required to answer the client query then the DA can construct a compound data item corresponding to the client query and disseminate the result to the client query so that incoherency bound is not violated [3]. It is noticeable that if we get the query result from single DAs, the number of refreshes will be (May data item updates cancel out each other by maintaining the incoherency bound within the query result). Further, aggregators can refresh all the data items; it may not be able to convince the necessities of query coherency. In such cases, the query has to be executed with data from multiple aggregators.

2. Related Work

Value of a continuous weighted additive aggregation query at time t can be calculated as first; 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 distributing 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 [3]. 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 aggregator makes sure that the i^{th} data item has incoherency bound C_q, following condition ensures that the query incoherency bound C_q satisfied X. The client is specified query incoherency bound needs to be translated into incoherency bounds for individual data items or sub queries such that (3) is satisfied as in [4]. It should be noted that a sufficient condition for satisfying the query incoherency bound that is not essential. Way of converting the query into sub query incoherency bounds is required, if the data is transferred between various nodes using only push-based mechanism.

We need a method for

a) Optimally we divide a client query into sub queries.

b) Transfer incoherency bounds to them.

c) Derived sub queries can be executed at chosen data aggregators.

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 present efficient algorithm 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 [3], [5] and [6] proposed getting individual data items from the aggregators which leads to large number of refreshes. To solve the above problem for best feasible result, dividing the client query into sub queries, we first need a method to estimate the query execution cost for various alternatives. The method for approximately calculations, query is another important contribution. As we divide the client query into sub queries such that each sub query get executed at different aggregator nodes, query execution cost (i.e. number of refreshes) is the sum of the execution costs of its sub queries. The model of the sub query execution cost as a function of dissemination costs of data items involved.

3. 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) Coherency requirement itself and

2) Dynamics of the data.

3.1. 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 as discrete as time random process [7] where each step is correlated with its earlier step. In a push-based distribution method, data source can follow the following schemes:

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

ii. Client estimates data value based on server specified parameters. Source pushes the new data value whenever it differs from the (client), as estimated value amount is more than C.

In both these cases, value at the source can be modeled as a Random process with average as the

known value to the client. In case 2, the client and the server estimate the data as the mean of the random process model, whereas in case 1, deviation from the last pushed value can be as zero mean model process. Using Chebyshev's inequality [7], we hypothesize; the number of data refresh messages is inversely proportional to the square of the incoherency bounds. A similar result was reported in where data dynamics were model 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 indicates is sent to the client. Data sources maintain value last sent for each client. Sources read new values 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. To ensure incoherency bound, each data item the incoherency bound was varied and refresh messages were counted.

3.2. Data Dynamics

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. Assume both data items are disseminated with an incoherency bound of 3. It can be seen that for maintaining the incoherency bound, the number of messages will be required 7 and 1 for data item d1 and d2 respectively, whereas both data items have the same standard deviation. This motivates us to examine the second measure. As second option, we considered Fast Fourier Transform (FFT) which is used in the digital signal processing domain to characterize a digital signal. Thus, FFT captures number of changes in data amount of changes, and their timings. Using FFT to model data dynamics but it has a problem. Estimating the number of refreshes required to disseminate a data item, we need a function over FFT coefficients which can return a scalar value. Number of FFT coefficients can be as high as the number of changes in data value. 0th order coefficient identifies average value of the data item among FFT coefficients, whereas higher order coefficients represent transient changes in the value of data item. Thus, we hypothesize that, the cost of data dissemination for a data item can be approximated by a function of the first FFT coefficient. The cost of data dissemination for a data item will be proportional to data sum diff defined. 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.

Comparison of Algorithms:

We consider various other query plan options for comparison of algorithms. Each query can be executed by disseminating individual data items or by getting sub query values from data aggregators. Various combinations of these dimensions of algorithm are as follows:

1. No subquery, equal incoherency bound
2. No subquery, optimal incoherency bound
3. Random subquery selection
4. Subquery selection while minimizing sumdiff
5. Subquery selection while maximizing gain

.

Validating the hypothesis:

We did simulations with different stocks being disseminated with incoherency bound values of $0:001, 0.01, and 0.1. Range is 0.1 to 10 times the average standard deviation of the student performance 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), measuring the degree of linearity between two variables, a widely used measure of association. This 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 confidence. For three values of incoherency bound such as 0.001, 0.01 and 0.1; PPMCC values were 0.94, 0.96 and 0.90 respectively. 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, linear relationship between data sum diff and the number of refresh messages can be assumed with more confidence for lower values of the incoherency bound.

4. Query panning for weighted Query

A query plan is an ordered set of steps used to access or modify information in relational database system. This is a specific case of the relational model which is used as concept of access plans. Since Oracle is declarative with database, there are typically a large number of options to execute a given query, with widely varying performance. When query is submitted to database, the query optimizer evaluates some of the correct possible different plans for executing the query and returns best alternative options. 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.

4.1. Optimization objective

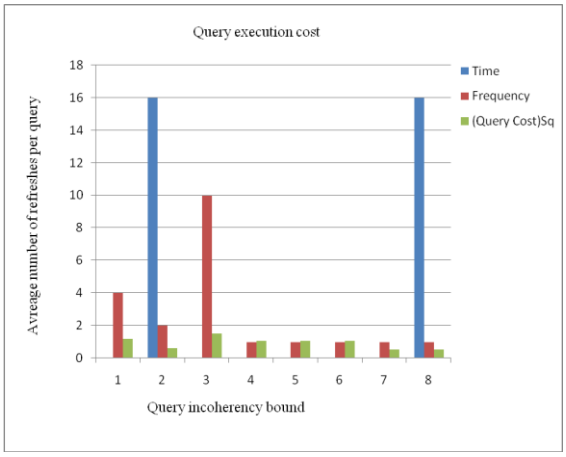
The numbers of refresh messages are minimized. For a sub query the estimated number of refresh messages is given by the ratio of sum diff of the sub query and the incoherency bound assigned to it and the proportionality factor k. Thus, the total number of refresh messages is estimated as the summation of the ratio of the sub query of a given query and in coherency bound associated to it.

4.1. Cost based query planning

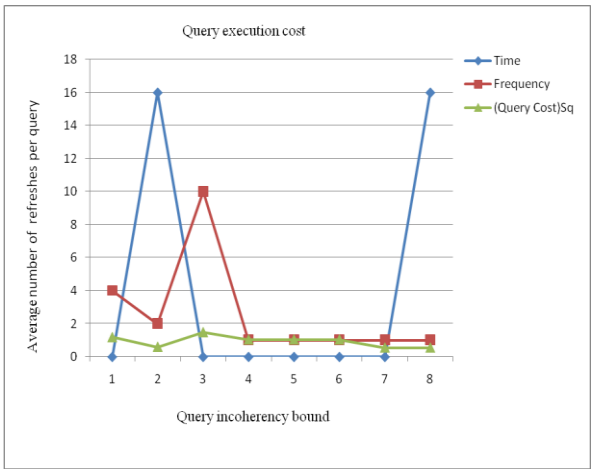
Conversion of logical query plan to physical query plan:

• Obtaining a physical query plan we need to assign each node in the logical query plans a physical operator.
• Need to obtain the physical plan with the smallest total execution cost.
• We need to compare for every node and every applicable physical operator, its cost.

To execute the MAX query using a network of data aggregators; we just assign sub queries to different DAs. Each subquery is a MAX query over a subset of query in incoherency bound. For optimal planning, we need to minimize the sum of sub query execution cost. We assign same incoherency bound to all the subqueries (equals to the query incoherency bound as per (19)) [2], we need to minimize sum of subquery sumdiff values.



(a)



(b)

Fig 3: Performance of MAX queries (a) Comparison of algorithm and (b) Effect of data dynamics order on performance

5. Conclusion

Here we presents a cost-based approach to minimize the number of refreshes required to execute an incoherency bounded uninterrupted query. As 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. Optimal query implementation, we divide the query into sub queries and evaluate each sub query at a judiciously chosen data aggregator.

10. References

[1] P. Edara, A. Limaye, and K. Ramamritham, "Asynchronous In-Network Prediction: Efficient Aggregation in Sensor Networks," ACM Trans. Sensor Networks, vol. 4, no. 4, pp. 1-34, Aug. 2008.

[2] Rajeev Gupta, Kirthi Ramaritham, " Query Planning For Continuous Aggregation Queries Over A Network Of data Aggregators" vol 24, No. 6. June 2012.

[3] C. Olston, J. Jiang, and J. Widom, "Adaptive Filter for Continuous Queries over Distributed Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2003.

[4]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.

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

[6] S. Shah, K. Ramamritham, and C. Ravishankar, "Client Assignment in Content Dissemination Networks for Dynamic Data,"Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005.

[7]A. Populis, Probability, Random Variable and Stochastic Process. Mc. Graw-Hill, 1991.