International Journal of Current Trends in Engineering & Research (IJCTER)
e-ISSN 2455–1392 Volume 2 Issue 4, April 2016 pp. 20 - 26
Scientific Journal Impact Factor : 3.468
http://www.ijcter.com

Query Processing and Query Data Aggregation for Continuous Queries

Reemi Shirsath[1], Bhawna Shivhare[2], Prachi Chaudhari[3], Pooja Kadam[4]

[1,2,3,4] Department of Information Technology
MET's Institute of Engineering,
Nashik, India

Abstract—The transformation of passive web pages into active environment through the continuous queries by providing the queries which have time varying dynamic query result useful for making online decision. Different users having diverse interest desire to obtain the value of some aggregation function over distributed data item, for example, to find the value of portfolio for The client. In these The client specifies a coherency requirement as part of query and for these queries low cost and scalable technique network of data aggregators. The entire node cannot individually determine by itself its inclusion in the query result for these different algorithmic challenges are presented like aggregators and selection queries. Each data item serve for set of data aggregators. Disseminating a client query into sub queries and then executing this sub queries on chosen data aggregator are involved in this technique. A query cost model, we build which can be used to estimate the number of refresh message which is required to satisfy client specified incoherence bound.

Keywords—Algorithms, continuous queries, distributed query processing, data dissemination, coherency, and performance

I. INTRODUCTION

Dynamic data denotes information that is asynchronously changed as further updates to the information become available, in that there is no constant flow of information. Rather, updates may come at any time, with periods of inactivity in between. The current version of the wiring schematic is considered "dynamic" (changeable and changing).The "dynamic" updated data in both cases. Stock data values from possibly different sources are required to be aggregated to be aggregated to satisfy client's requirement. While certain conditions are on hold the client is interested in notifications of the aggregated queries that are long running queries as data is continuously changing. Thus number of refreshes of queries.

In these continuous query applications, The client probably tolerate some inaccuracy in the results. The exact value might be The client will not get as per their specified accuracy requirement. For instance, an auction maker may be happy with an accuracy of $2000 as per the deal. How technology has influenced The Stock Market Technology has had a massive impact on our lives and is generally regarded to have improved our social lives, businesses, governments and education. Unless you are a broker or an investor not much thought is given as to the impact technology has had on the productivity of The Stock Market**.**

Data incoherency: Data accuracy can be specified in terms of incoherency of a data item, defined as the absolute difference in value of the data item at the data source and the value known to a client of the data. Let $v_i(t)$ denote the value of the ith data item at the data source at time t and let the value the data item known to the client be $u_i(t)$. Then the data incoherency at the client is given by $|v_i(t)-u_i(t)|$. For a data item which needs to be refreshed at an incoherency bound C a data refresh message is sent to the client as soon as data incoherency exceeds C, i.e., $|v_i(t)-u_i(t)|>C$.

Network of data aggregators: Data aggregation is the compiling of information from databases with intent to prepare combined datasets for data processing. Data refresh from data sources to clients can be done using push or pull based mechanisms. In a push based mechanism data sources send update messages to clients on their own whereas in pull based mechanism data sources send messages to the client only when the client makes a request. Push based mechanism applicable for the data transfer between data sources and clients. For the scalable handling of push based data dissemination, network of data aggregators are proposed. In such network of data aggregators, data refreshes occur from data sources to the clients through one or more data aggregators.

In this paper we assume that each data aggregator maintains its configured incoherency bounds for various data items. From a data dissemination capability point of view, each data aggregator (DA) is characterized by a set of (d_i, c_i) pairs, where d_i is a data item which DA can disseminate at an incoherency bound c_i. The configured incoherency bound of a data item at a data aggregator can be maintained using any of following methods: (a) The data source refreshes the data value of the DA whenever DA's incoherency bound is about to get violated. This method has scalability problem. (b) Data aggregator(s) with tighter incoherency bound help the DA to maintain its incoherency bound in a scalable manner.

Example 1: In network of data aggregators managing data items d_1-d_4, various aggregators can be characterized as-

A_1: {(d_1, 0.5), (d_3, 0.2)}

A_2: {(d_1, 1.0),(d_2, 0.1),(d_4, 0.2)}

Aggregator A_1 can serve values of d_1 with an incoherency bound greater than or equal to 0.5 whereas A_2 can disseminate the same data item at a looser incoherency bound of 1.0 or more. In such a network of aggregators of multiple data items all the nodes can be considered as peers since a node A_1 can help another node A_k to maintain incoherency bound of the data item d_1 but the node A_i gets values of another data item d_2 from A_k.

1.1 Execution of Aggregation Queries

- In this paper we present network of data aggregators for executing continuous multi-data aggregation queries, So as. to minimize the number of refreshes from data aggregator to client. Our main aim is dividing the queries into sub-queries to obtain the optimum result with the minimum number of refreshes.
- Our method of dividing query into sub-queries and executing them at individual DAs require least hand one third of the number of refreshes required in existing schemes .
- More dynamic data items should be part of sub query having large number of data items in order to reduce number of refreshes.

In this paper we have proposed a method to answer the client query using a given network of data aggregators ;if client queries are fixed ,one can use the client queries to optimally construct a network of data aggregators as in[5,7].Our intention of minimizing the number of message between aggregators and client compliments the work of[5,7],and so these both can be used together to minimize the total number of message between data source and client.

1.2 Paper layout

The cost model for data dissemination is developed in Section 2.In Section 3, we are developing the model for the parsing aggregation queries which will get answer based on the Incoherency bound. It uses the Incoherency bound model and a measure for capturing the relationship between data dynamics. Optimal query planning for parsing queries is presented in Section 4.Result of performance evaluations algorithm described in Section 4 are presented in Section 5. Section 6 discusses optimal query planning for max queries. Most conclusions drawn for this class of queries is similar to that for parsing queries. Related work is presented in Section 7. Discussion about various aspects of our work, conclusions and future work are presented in Section 8.Table 1 gives summary of various symbols used in the paper and their descriptions.

II. DATA DISSEMINATION COST MODEL

In this section we present the model to estimate the minimum numbers of refreshes required to disseminate a data item while maintaining a certain data item while maintaining a certain incoherency bound.

2.1 Incoherency Bound Model

Consider a data item which needs to be disseminated at an incoherency bound C, i.e., new value of the data item will be pushed if the value deviates by more than C from the last pushed value. The number of dissemination message will be proportional to the probability of |v(t)- u(t)| 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[10] where each step is correlated within its previous step in push based dissemination, a data source can follow one of the following schemes.

a) Data source pushes 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 [12, 16]. 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 (b), the client and the server estimates the data value as the mean of the modeled random process where in case (a) deviation from the last pushed value can be modeled as zero mean process. Using Chebyshev’s inequality [10]:

$$P\,(|v\,(t)-u\,(t)|>C) \propto 1/C^2 \qquad (4)$$

The number of data refresh message is inversely proportional to the square of the incoherency bound.

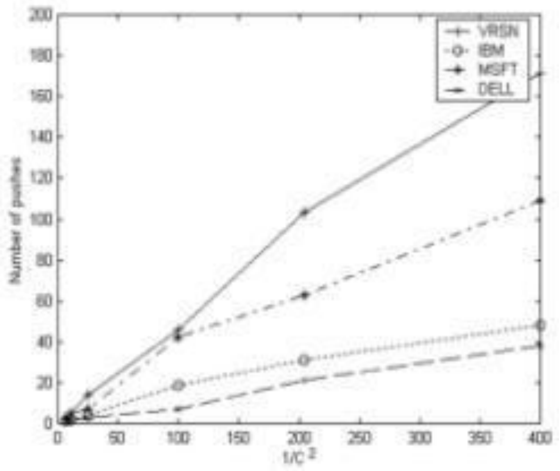

Fig 1: Number of purchases VS incoherence Bounds

III. AGREGGATION AND 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.

Constraint 1: 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 dq k is required for the sub query.

Constraint 2: Query incoherency bound is satisfied: Query incoherency should be less than or equal to the query incoherency bound. Thus, 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.

Constraint 3: 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 qk. 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.

IV. EVOLUTION IN PERFORMANCE

For performance evaluation we simulated the data dissemination networks of 25 stock data items over 25 aggregator nodes such that each aggregator can disseminate combinations of up to 10 data items with data incoherency bounds chosen uniformly between $0.005 and 0.02. Then we created 500 portfolio queries such that each query has up to 10 randomly (uniformly) selected data items with weights varying between 2 and 10. These queries were executed with incoherency bounds between 0.3 and 1.0 (i.e., 0.03-0.1% of the query value). In the first set of experiments, we kept the data incoherency bounds at the data aggregators very low so that query satisfiability can be ensured.

Comparison of algorithms: For comparison with our algorithms, presented in the previous section, we consider various other query plan options. Each query can be executed by disseminating individual data items or by getting sub-query values from DAs. Set of sub-queries can be selected using sumdiffbased approaches or any other random selection. Sub-query (or data) incoherency bound can either be pre-decided or optimally allocated. Various combinations of these dimensions are covered in the following algorithms:

1. No sub-query, equal data incoherency bound (naïve): In this algorithm, the client query is executed with each data item being disseminated independent of other data items in the query. Incoherency bound is divided equally among the data items. This algorithm acts as a baseline algorithm.

2. No sub-query, optimal incoherency bound : In this algorithm also data items are disseminated separately but incoherency bound is divided among data items so that total number of refreshes can be minimized. This algorithm is similar to the one presented in [11]. Here, the incoherency bound is allocated dynamically.

3. Random sub-query selection (random): In this case, sub queries are generated by randomly selecting one data aggregators and allocating it the maximal sub-query consisting of query data items which the aggregator can disseminate. Then the process is repeated for the remaining data items until the whole query is covered. This algorithm is designed to see how the sub-query selection based on query sum diff works in comparison random selection of sub-queries.

4. Sub-query selection while minimizing sum diff (min-cost):

This algorithm is described in Section 4

5. Sub-query selection while maximizing gain (max-gain): This algorithm is described in Section 4.3.Figure 5 shows average number of refreshes required for query incoherency bounds of $0.3, $0.5 and $0.8. The naïve algorithm requires more than three times the number of messages compared to min-cost and max-gain algorithms. For incoherency bound of $0.8 each query requires 1024 messages if it is executed just by optimizing incoherency bound compared to 255 when we select the query plan using the max-gain algorithm. Further, although the optimization problem is similar to the covering a set of data items (query) using its sub-sets (sub-queries) for which the greedy min-cost algorithm is considered to be most efficient [7], we see that max-gain algorithm requires 20-25% less messages compared to the min-cost approach. Reasons for max-gain algorithm performing better than other algorithms are explored in the next set of experiments. Although here we presented results for stock traces (man-made data) similar results were obtained for sensor traces (natural data) as well.

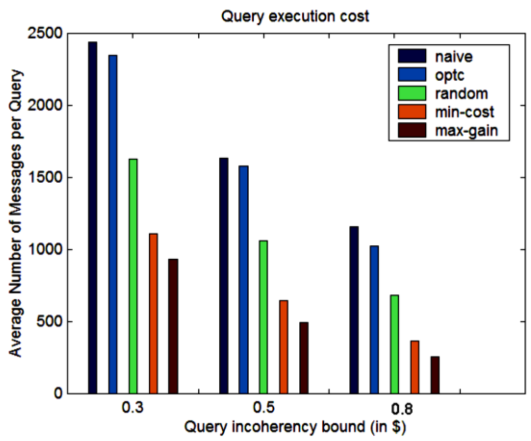

Figure 2: Performance evaluation of algorithms

V. RELATED WORK

Various mechanisms for efficiently maintaining incoherency bounded aggregation queries over continuously changing data items are proposed in the literature [11, 12, 16]. Our work distinguishes itself by being sub-query based evaluation to minimize number of refreshes. In [11], authors propose using data filters at the sources; instead we assign incoherency bounds to sub-queries which reduce the number of refreshes for query evaluation, as explained in Section 5. Further, we propose that more dynamic data items should be executed as part of larger sub-query. In [21], authors present technique of reorganizing a data dissemination network when client requirements change. Instead, we try to answer the client query using the existing network. Reorganizing aggregators is a longer term activity whereas query planning can be done for short as well as long running queries on more dynamic basis. Pull based data dissemination techniques, where clients or data aggregators pull data items such that query requirements are met, are described in [9,16]. For minimizing the number of pulls, both model the individual data items and predict data values. In comparison, we consider the situation where different sub-queries, involving multiple data items, can be evaluated at different nodes. Further, incoherency bound is applied over the sub-query rather than to individual data items, leading to efficient evaluation of the query. Spatial and temporal correlations between sensor data are used to reduce data refresh instances in [17, 18]. We also consider correlation in terms of cosine similarity between data items, but we use it for dividing client query into sub-queries. Our work can be extended by using temporal and spatial properties of data items for predicting their correlation measures. A method of assigning clients data queries to aggregators in a content distribution network is given in [12]. We do for client queries consisting of multiple data items what [12] does for client requiring individual data items.

VI. CONCLUSIONS

This paper presents a cost based approach to minimize the number of refreshes required to execute an incoherency bounded continuous query. For optimal execution we divide the query into sub-queries and evaluate each sub-query at a chosen aggregator. Performance results show that by our method the query can be executed using less than one third the messages required for existing schemes. Further we showed that by executing queries such that more dynamic data items are part of a larger sub-query we can improve performance. Our 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. Using the cost model for other applications and developing the cost model for more complex queries is our future work.

REFERENCES

[1] A. Davis, J. Parikh and W. Weihl. Edge Computing:Extending Enterprise Applications to the Edge of theInternet. WWW 2004
[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] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman and B. Weihl. Globally Distributed Content Delivery, IEEE Internet Computing Sept 2002.
[4] S. Rangarajan, S. Mukerjeeand P. Rodriguez. User Specific Request Redirection in a Content Delivery Network, 8th Intl.Workshop on Web Content Caching and Distribution (IWCW), 2003.
[5] S. Shah, K. Ramamritham, and P. Shenoy. MaintainingCoherency of Dynamic Data in Cooperating Repositories. VLDB 2002.
[6] Dynamai: Caching Technology for Dynmaic ContentRevealed. www.infoworld.com/articles.
[7] D. S. Hochbaum. Approximation algorithms for the setcovering and vertex cover problems. SIAM Journal on Computing, vol. 11 (3), 1982.
[8] ZongmingFei. A Novel Approach to Managing Consistency in Content Distribution. WCW 2001
[9] R. Gupta, A. Puri, and K. Ramamritham. ExecutingIncoherency Bounded Continuous Queries at Web Data Aggregators. WWW 2005.
[10] Optimized Execution of Continuous Queries, APS 2006, www.cse.iitb.ac.in/~grajeev/APS06.PDF

[11] S. Shah, K. Ramamritham, and C. Ravishankar. Client Assignment in Content Dissemination Networks for Dynamic Data. VLDB 2005.
[12] C. Olston, J. Jiang, and J. Widom. Adaptive Filter for Continuous Queries over Distributed Data Streams. SIGMOD 2003
[13] NEFSC Scientific Computer Systemhttp://sole.wh.whoi.edu/~jmanning//cruise/serve1.cgi
[14] Query cost model validation for sensor data. www.cse.iitb.ac.in/~ravivj/BTP06.pdf.
[15] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing, vol. 11 (3), 1982
[16] S. Zhu and C. Ravishankar. Stochastic Consistency and Scalable Pull-Based Caching for Erratic Data Sources. VLDB 2004.
[17] D. Chu, A. Deshpande, J. Hellerstein, W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models. ICDE 2006.
[18] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. VLDB, 2004.
[19] Lam, W. and Ho, C.Y. Using a Generalized Instance Set for Automatic Text Categorization. SIGIR, 1998.
[20] G. Cormode and M. Garofalakis. Sketching Streams through the Net: Distributed Approximate Query Tracking. VLDB 2005.
[21] S. Agrawal, K. Ramamritham and S. Shah. Construction of a Temporal Coherency Preserving Dynamic Data Dissemination Network. RTSS 2004.