IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Distributed Continuous Query Processing over a Network of Data Aggregators

N. Dilip Kumar[1], A. Jaganath[2] and R. Ranjani[3]

[1,2,3]Department of Information Technology, Anand Institute of Higher Technology, Kazhipattur, Chennai, Tamil Nadu, India

Abstract

Processing of continuous queries that are getting updated dynamically and satisfying the client's requirement with the least number of refresh messages has always been a concern. To process such queries, we propose to execute those queries over a network of data aggregators, each having their own incoherency bound. The queries are first sub-divided into sub-queries and disseminated to various data aggregator's network, where the data item that matches client's requirement is provided to the user based on his preference already stored in the system.

Keywords: *Continuous Queries, Refresh Messages, Data Aggregators, Incoherency*

1. Introduction

When the continuous queries are given as input then the output must be planned. Here data aggregators are the data updaters. They provide the data to users according to their queries. Here query planning tells us how to sense the queries and how to disseminate the queries to target. This system is reducing the overhead of users (source of queries).

The existing system of a search engine focuses primarily on the page rank, which is given to a page based on the number of users who accessed the page. It is only based on the page rank that the result to user search gets ordered and updated.

Thus the system has some problems due to inadequacy in efficiency. Query plan selection problem is not fully rectified and analyzed. Actually user queries are having many meanings depending upon the users mind set. Though some of the queries are analyzed, problems are not fully rectified; but some of the problems are cleared. Thus the system was too flexible to handle the user queries and produce the results.

The rest of this paper is as follows: Section 2 presents the existing work of this system. Section 3 explains about the proposed system. Section 4 briefly explains about the system design. In Section 5, we conclude this system with future work of this paper.

2. Related Work

Many data intensive applications delivered over the Web suffer from performance and scalability issues. Content distribution networks solved the problem for static content using caches at the edge nodes of the networks. 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.

For static pages content requested, proximity to the client and load on the nodes are the parameters generally used to select the appropriate node. Techniques to efficiently serve fast changing data items with guaranteed incoherency bounds have been proposed in the literature. Such dynamic data dissemination networks can be used to disseminate data such as stock quotes, temperature data from sensors, traffic information, and network monitoring data.

2.1. Query Plan Selection Technique

In this section, we give a formal definition of the optimization problem described. We are given a set D of data aggregators, set S of data items and one-to-many mapping f: $D\_(S, C)$ where $C\_\_$ is a sub-set of

real number representing incoherency bounds for various data items (in the set *S*) at aggregators in *D*. Each incoming client query *q* over the data items set *Sq* _ *S* has corresponding weights given as a set *Wq*.

Thus the query can be represented as set of tuples of *_data item, weight_*, i.e., *q*= {*sq, wq*} with the query incoherency bound *Cq*.

We need to perform the following two tasks such that the number of refreshes to the client is minimum:

Task1: Divide the client query *q*= {(*sq, wq*)} into sub-queries *qk*={(*qkqsk* ,*w*)} so that *qk* = *q* i.e., although different sub-queries may be executed at different aggregators, combining their results gives the value of the client query.

Task2: Allocate each sub-query *qk*, with its incoherency bound *Ck* , to data aggregators.

While fulfilling the following conditions:

Condition1: Query incoherency bound is satisfied, i.e.,_ £*kq Ck <= Cq* .The sub-query *qk* should be assigned to a data aggregator *di* _Diff:

Condition2: The chosen aggregator should have all the data items appearing in the sub-query i.e.Õ*S* (*qk*) ÍÕ*S* (*f* (*di*)) . Here _indicates project operator in relational algebra.

Condition3: Data incoherency bounds at the selected data aggregator Õ = = ((( )) *c j C s s q* (*j*) *fdik*s should be such that *Ck*³ *Xk*

where *sk q*(*j*) is the *j*th data item appearing in the subquery *qk* and *Xk* is the tightest incoherency bound the aggregator can ensure for the given sub-query. *Xk* can be calculated as: = _ *qj Xk c jw* . Here _ indicates select operator in relational algebra.

3. Proposed System

We provide a technique 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 estimating the number of refresh messages, we build a query cost model which can be used to estimate the number of messages required to satisfy the client specified incoherency bound.

No of users can access with network of data aggregators to get the data which are provided by them. So many users give more queries to network of data aggregators. Here our system needs to handle more than a user at a time as well as give correct results to users. Here number of refresh messages are to be reduced. Data aggregators are considered as administrators and so they can update their data.

Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. Providing a technique 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 is the main challenge.

3.1. Greedy Heuristics Algorithm

Fig. 1 gives the outline of greedy algorithm for deriving subqueries. First, we get a set of maximal subqueries (Mq) corresponding to all the data aggregators in the network. The maximal subquery for a data aggregator is defined as the largest part of the query which can be disseminated by the DA (i.e., the maximal subquery has all the query data items which the DA can disseminate).

For example, consider a client query 50d1 + 200d2 + 150d3. For the data aggregators a1 and a2 given in Example 1, the maximal subquery for a1 will be m1 = 50d1 + 150d3, whereas for a2 it will be m2 = 50d1 + 200d2. For the given client query (q) and relation consisting of data aggregators, data items, and data incoherency bounds (f(A; D;C)) maximal subqueries can be obtained for each data aggregator by forming subquery involving all data items in the intersection of query data items and those being disseminated by the DA.

Different criteria can be used to select a subquery in each iteration of various greedy heuristics. All data items covered by the selected subquery are removed from all the remaining subqueries in Mq before

performing the next iteration. It should be noted that subqueries for data aggregators can be null.

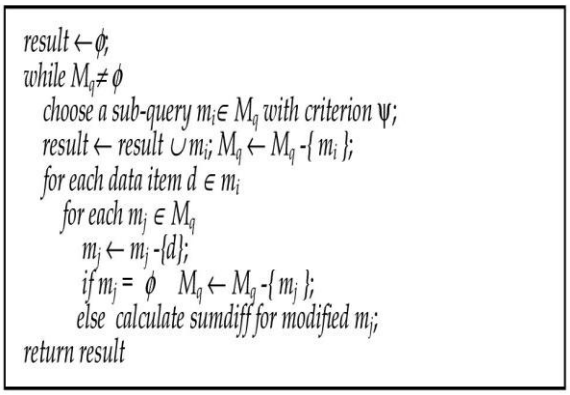

Fig. 1 Greedy algorithm for query plan selection.

Now we describe two criteria for the greedy heuristics; 1) min-cost: estimate of query execution cost is minimized, and 2) max-gain: estimated gain due to executing the query using subqueries is maximized.

3.2. Query Cost Model

Incoherency bound model is used for estimating dependency of data dissemination cost over the desired incoherency bound. As per this model, the number of data refreshes is inversely proportional to the square of the incoherency bound ($1/C^2$).

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

Data Synopsis Model is used for estimating the effect of data dynamics on number of data refreshes. We define a data dynamics measure called, *sum diff*, to obtain a synopsis of the data for predicting the dissemination cost. 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. Thus we hypothesize that cost of data dissemination for a data item will be proportional to *sum diff* where *si* and *si-1* are the sampled values of the data item at *ith* and (*i-1*)*th* time instances (consecutive ticks).

Pearson product moment correlation coefficient (PPMCC) values, used for quantifying linearity between data *sum diff* and number of refreshes required to maintain a fixed incoherency bound, were found to be between 0.90 and 0.96 for various values of incoherency bounds. *Sum diff* value for a data item can be calculated at the data source by taking running average of difference between data values at the consecutive ticks.

A data aggregator can also estimate the *sum diff* value by interpolating the disseminated values. Thus, the estimated dissemination cost for data item *S*, disseminated with an incoherency bound *C*, is proportional to Rs/C^2. Next we use this result for developing the query cost model.

A query consists of two data items *P* and *Q* with weights *wp* and *wq*.

If the data items are disseminated separately, the dissemination cost is

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

If client is interested in a query over P and Q it creates and pushes a composite data item ($w_p p + w_q q$)

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

4. System Model

The architecture of this system as shown in Fig. 2 would explain the system description of this work. User gives his query and it gets converted into sub queries and sent through the data dissemination network to several data aggregators. Results are to be showed to the user by data aggregators.

The modules in the system are as follows:

4.1. Entry Level Design

In user login module user, we login to the user page to connect with the application.

New users can register by giving the details to registration form. Users can register using the option that is in user login design form.

4.2. Query Plan

In this module 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. We find the optimal query execution plan which satisfies client coherency requirement with the least number of refreshes.

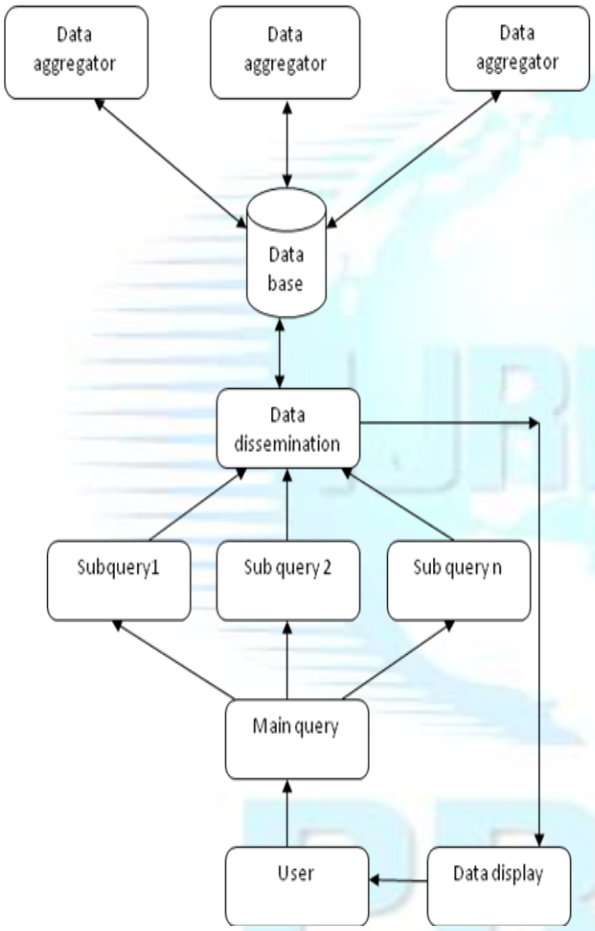

Fig. 2 Proposed system architecture

4.3. Data Dissemination

In this module, the given data of user is disseminated to various data aggregator's network. This data dissemination is pull based or push based. After data dissemination the data from aggregators will be collect and save on our database. This process helps to reach every data aggregator and to get data from data aggregators.

4.4. Data Aggregators

In this module each data aggregator serves the data item at some guaranteed incoherency bound. 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.

Data aggregators update their data using data aggregators page. Finally data aggregator returns the data to user.

4.5. Data Presentation

In this module the data from each data aggregators for the sub queries are combined together and presented to user to satisfy the user and incoherency value is measured for future analysis. So, we present data to users according to their satisfaction criteria. This helps us to generate an efficient answering system.

5. Conclusion and Future Work

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

In future we will implement the different database models such as excel and data are to be accessed by different data sources. Refresh timings will be shown in our future. The refresh concept between user and data aggregator will be shown to understand the concepts. Performance analyze will be calculated for both single user as well as all users. This will be helpful for admin to understand the users. Performance analyze is based on the number of links used by users provided by data aggregator`s results.

Using the cost model for other applications and developing the cost model for more complex queries is our future work.

References

[1] 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.

[2] "Query Cost Model Validation for Sensor Data",www.cse.iitb.ac.in/~grajeev/sumdiff/RaviVijay_ BTP06.pdf, 2011.

[3] R. Gupta, A. Puri, and K. Ramamritham, "Executing Incoherency Bounded Continuous Queries at Web Data Aggregators," Proc.14th Int'l Conf. World Wide Web (WWW), 2005.

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

[5] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y.Zhang, "STAR: Self-Tuning Aggregation for Scalable Monitoring,"Proc. Int'l Conf. Very Large Data Bases (VLDB), 2007.

[6] R. Gupta and K. Ramamritham, "Optimized Query Planning of Continuous Aggregation Queries in Dynamic Data Dissemination Networks," Proc. 16th Int'l Conf. World Wide Web (WWW) 2007.