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

Continuous Aggregation Queries With Minimized Number Of Refreshes Over A Network Of Data Aggregators

[1]K.Gowtham Nagu, [2]V.Kathiresan- Student
[3]A.Gomathi – Head of the Department
Department of Computer Science and Engineering
Narasu's Sarathy Institute of Technology, Salem.

ABSTRACT

Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. Typically a user desires to obtain the value of some aggregation function over distributed data items, for example, to know (a) the average of temperatures sensed by a set of sensors (b) the value of index of mid-cap stocks. In these queries a client specifies a coherency requirement as part of the query. In this paper we present a low-cost, scalable technique to answer continuous aggregation queries using a content distribution network of dynamic data items. In such a network of data aggregators, each data aggregator serves a set of data items at specific coherencies. Our technique involves decomposing a client query into sub-queries and executing sub-queries on judiciously chosen data aggregators with their individual sub-query incoherency bounds. We provide a technique of getting the optimal query plan (i.e., set of sub-queries and their chosen data aggregators) which satisfies client query's coherency requirement with least cost, measured in terms of the number of refresh messages sent from aggregators to the client. For estimating query execution cost, we build a continuous query cost model which can be used to estimate the number of messages required to satisfy the client specified incoherency bound.

Index terms – Continuous queries, distributed query processing, data dissemination, coherency.

1. INTRODUCTION

Many data intensive applications delivered over the Web suffer 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 [1, 2]. 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 [3]. 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. Techniques to efficiently serve fast changing data items with guaranteed incoherency bounds have been proposed in the literature [4,5]. 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. In this paper we propose a method to efficiently answer aggregation queries involving such data items.

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 i^{th} 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 refresh from data sources to clients can be done using push or pull based mechanism data sources and messages to the client only when the client makes a request. We assume the push based mechanism for data transfer between data sources and clients. For Scalable handling of push based data dissemination, network of data aggregators are proposed in the literature. 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 a 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 problems. (b) Data aggregator(s) with tighter incoherency bound in a scalable manner.

Example 1: In a network of data aggregators managing data items s_1-s_4, various aggregators can be characterized as-

a_1: {(s_1,0.5), (s_3,0.2)}

a_2:{(s_1,1.0), (s_2,0.1), (s_4, 0.2)}

Aggregator a_1 can serve values of s_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_i can help another node a_k to maintain incoherency bound of the data item s_1 (incoherency bound of d_1 at a_i is tighter than that at a_k) but the node a_i gets values of another data item s_2 from a_k.

1.1 Summary of Approach and Contributions

Consider a client query *Q1*=50 *S1* + 200 *S2* + 150 *S3* with a required incoherency bound of 80 (in a stock portfolio *S1, S2, S3* can be different stocks and incoherency bound can be $80).We want to execute this query over data aggregators given in *Example1* minimizing number of refreshes. There are various options for the client to get the data items:

- The client may get the data items *S1*, *S2* and *S3* separately. The query incoherency

bounds can be divided among data items in various ways while satisfying Equation 3. In this paper, we show that getting data items independently is a costly option. This strategy ignores facts 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 DA can disseminate all three data items required to answer the client query, the DA can construct a composite data item corresponding to the client query (Sq=50 $S1$ + 200 $S2$ + 150 $S3$) and disseminate the result to the client so that the query incoherency bound is not violated. It is obvious that if we get the query result from a single DA, the number of refreshes will be minimum (as in this case data item updates may cancel out each other, thereby keeping the query result within the incoherency bound). As different data aggregators disseminate different subsets of data items, no data aggregator may have all the data items required to execute the client query which is indeed the case in *Example1*. Further, even if an aggregator can disseminate all the data items, it may not be able to satisfy the query coherency requirements. In such cases the query has to be executed with data from multiple aggregators.
- Another option is to divide the query into a number of sub-queries and get their values from individual DAs. In that case, the client query result is obtained by combining the results of more than one sub-query. For the DAs given in *Example1*, the query *Q1* can be divided in two alternative ways:

  plan1: *D1* {50 *S1* + 150 *S3*}; *D2* {*S2*}

  plan2: *D1* {*S3*}; *D2* {50 *S1*, + 200 *S2*}

  i.e., in *plan1* result of sub-query 50 *S1* + 150 *S3* is disseminated by *D1* and that of *S2* (or 200 *S2*) by *D2*. Combining them at the client gives the query result.
- Selecting the optimal plan among various options is not-trivial. As a thumb-rule, we should be selecting the plan with lesser number of sub-queries. But that is not guaranteed to be the plan with the least number of messages. Further, we should select the sub-queries such that updates to various data items appearing in a sub-query have more chances of canceling each other as that will reduce the need for refresh to the client (Equation 2). In the above example, if updates to *S1* and *S3* are such that when *S1* increases, *S3* decreases, and vice-versa, then selecting *plan1* may be beneficial. We give an algorithm to select the query plan based on these observations.
- While solving the above problem of selecting the optimal plan we 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 DAs (e.g., 50 *S1* + 200 *S2* + 150 *S3* is divided into two sub-queries 50 *S1* + 130 *S2* and 70 *S2* + 150 *S3*); 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 we ensure that a data item update is processed only once for each query (For example, in case of paid data subscriptions it is not prudent to get the same data item from the multiple sources).

- The query incoherency bound needs to be divided among sub-query incoherency bounds such that, besides satisfying the client coherency requirements, the chosen DA (where the sub-query is to be executed) is capable of satisfying the allocated sub-query incoherency bound. For example, in *plan1* allocated incoherency bound to the sub-query 50*S1* + 150*S3* should be greater than 55 (=50*0.5+150*0.2) as that is the tightest incoherency bound which the aggregator *D1* can satisfy. We prove that the number of refreshes depends on the division of the query incoherency bounds among sub-query incoherency bounds.

Thus, what we need is a method of (a) optimally dividing client query into sub-queries and (b) assigning incoherency bounds to them; such that (c) selected sub-queries can be executed at chosen DAs and (d) total query execution cost, in terms of *number of refreshes*, is minimized. **We prove that the problem of choosing sub-queries while minimizing query execution cost is an *NP* hard problem**. **We give efficient approximation algorithms to choose the set of sub-queries and their corresponding incoherency bounds for a given client query.** For solving the above problem of optimally dividing the client query into sub-queries, we first need a method to estimate query execution cost for various alternative options. **A method for estimating the query execution cost is another important contribution of this paper.** 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. We model the sub-query execution cost as a function of following parameters:

(a) Dissemination costs of 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.

(b) A correlation measure of data dynamics, quantifying the chance that the updates of two data items will cancel each other out such that a sub query of their sum will incur less refreshes than disseminating the individual data changes. We use *cosine similarity* between data items for this purpose. This parameter is widely used in information retrieval domain [6]. Through extensive simulations we show that:

- Our method of dividing query into sub-queries and executing them at individual DAs requires less than one third of the number of refreshes required in the existing schemes.

- For efficient execution, more dynamic data item should be part of sub-query involving larger number of data items.

Our method of executing queries over dynamic data dissemination network is practical since it can be implemented using a mechanism similar to URL-rewriting 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 can not 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. In case of a CDN, web page's division into fragments is a page design issue, whereas, for continuous aggregation queries, this issue has to be handled on per-query basis by considering *data dissemination capabilities* of data aggregators as represented in *Example* 1.

1.2 Problem Statement and Contributions

Value of a continuous weighted additive aggregation query, at time t, can be calculated as:

$$V_q(t) = \sum_{i=0}^{i=nq} (v_{qi}(t) \times w_{qi}) \quad (1)$$

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 w_{qi}, $1 \leq i \leq n_q$. Such a query encompasses SQL aggregation operators SUM and AVG besides general weighed aggregation queries such as portfolio queries, involving aggregation of stock prices, weighted with number of shares of stocks in the portfolio.

Suppose the result for the query given by Equation (1) needs to be continuously provided to a user at the query incoherency bound Cq. Then, the dissemination network has to ensure that:

$$\left| \sum_{i=0}^{n_q} \left(v_{qi}(t) - u_{qi}(t)\right) \times w_{qi} \right| \leq C_q \quad (2)$$

Whenever data values at sources change such that query incoherency bound is violated, the update value should be refreshed to the client. If the network of aggregators can ensure that the ith data item has incoherency bound cq is satisfied:

$$\sum_{i=0}^{n_q} (C_{qi} \times w_{qi}) \leq C_q \quad (3)$$

The client specified query incoherency bound needs to be translated into incoherency bound for individual data items or sub-queries such that Equation (3) is satisfied. It should be noted that Equation (3) is a sufficient condition for satisfying the query incoherency bound but not necessary. This way of translating the query incoherency bound into sub-query incoherency bounds is required if data is transferred between various nodes using only push based mechanism.

We need a method for (a) optimally dividing client query into sub-query and (b) assigning incoherency bound to them such that (c) the derived sub-queries can be executed at chosen Das and (d) total query execution cost, in terms of number of refreshes to the client, is minimized.

2. QUERY COST MODEL

Before developing 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. For simulation experiments we use data items from sensor network and stock data domains as explained in our previous work [7]. Stock traces of 45 stocks were obtained by periodically polling *http://finance.yahoo.com*. Sensor network data used were temperature and wind sensor data from Georges Bank Cruises Albatross Shipboard [8]. Due to paucity of space we present results using stock data only but similar results were obtained for sensor data as well [9]. For detailed analysis and simulation results, readers can refer to the extended version of the paper [10].

2.1 Data Dissemination Cost

Cost of disseminating a data item at a certain given incoherency bound *C* can be estimated by combining two models:

- *Incoherency bound model* is used for estimating dependency of data dissemination cost over the desired incoherency bound. As per this model, we have shown in [10] that the number of data refreshes is inversely proportional to the square of the incoherency bound ($1/C^2$). Similar result was earlier reported in [5] where the data dynamics was modeled as a random-walk process.

$$Data\ dissemination\ cost \propto 1/C^2 \quad (4)$$

- *Data Synopsis Model* is used for estimating the effect of data dynamics on number of data refreshes. We define a data dynamics measure called, *sumdiff*, 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 *sumdiff*, defined as:

$$R_S = \sum_i |s_i - s_{i-1}| \quad (5)$$

where s_i and s_{i-1} are the sampled values of the data item at i^{th} and $(i\text{-}1)^{th}$ time instances (consecutive ticks). In [10] we corroborate the above hypothesis using simulation over a large number of data items. *Sumdiff* 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 *sumdiff* value by interpolating the disseminated values.

Thus, the estimated dissemination cost for data item *S*, disseminated with an incoherency bound *C*, is proportional to R_s/C^2. Next we use this result for developing the query cost model.

2.2 Query Dissemination Cost

Consider a case where a query consists of two data items *P* and *Q* with weights w_p and w_q respectively; and we want to estimate its dissemination cost. If data items are disseminated separately, the query *sumdiff* will be:

$$R_{data} = w_p R_p + w_q R_q = w_p \sum |p_i - p_{i-1}| + w_q \sum |q_i - q_{i-1}| \quad (6)$$

Instead, if the aggregator uses the information that client is interested in a query over *P* and *Q* (rather than their individual values), it makes a composite data item $w_p p + w_q q$ and disseminates that data item then the query *sumdiff* will be:

$$R_{query} = \sum |w_p (p_i - p_{i-1}) + w_q (q_i - q_{i-1})| \quad (7)$$

R_{query} is clearly less than or equal compared to R_{data}. Thus we need to estimate the *sumdiff* of an aggregation query (i.e., R_{query}) given the *sumdiff* values of individual data items (i.e., R_p and R_q). Only data aggregators are in position to calculate R_{query} as different data items may be from different sources. We develop the query dissemination model in two stages.

2.2.1 Quantifying correlation between dynamics of data

From Equations (6) and (7) we can see that if two data items are correlated such that if value of one data item increases, that of the other data item also increases, then R_{query} will be closer to R_{data} whereas if the data items are inversely correlated then R_{query} will be less compared to R_{data}. Thus, intuitively, we can represent the relationship between R_{query} and *sumdiff* values of the individual data items using a correlation measure associated with the pair of data items. Specifically, if is the correlation measure then R_{query} can be written as:

$$R_{query}^2 \propto (w_p^2 R_p^2 + w_q^2 R_q^2 + 2\rho w_p R_p w_q R_q) \quad (8)$$

The correlation measure is defined such that $-1 \leq \rho \leq +1$, so, *Rquery* will always be less than $|wpRp+wqRq|$ (as explained earlier) and always be more than $|wpRp–wqRq|$. The correlation measure ρ can be interpreted as *cosine similarity* between two streams represented by data items *P* and *Q*. Cosine similarity is a widely used measure in information retrieval domain where documents are represented using a vector-space model and document similarity is measured using cosine of angle between two document representations. For data streams *P* and *Q*, ρ can be

calculated as:

$$\rho = \frac{\sum (p_i - p_{i-1})(q_i - q_{i-1})}{\sqrt{\sum (p_i - p_{i-1})^2} \sqrt{\sum (q_i - q_{i-1})^2}} \quad (9)$$

2.2.2 Query normalization

Suppose we want to compare the cost of two queries: a SUM query involving two data items and an AVG query involving the same data items. Let the query incoherency bound for the SUM and the AVG queries be $C_1=2C$ and $C_2=C$, respectively. From Equation (8), *sumdiff* of the SUM query will be double that of the AVG query (as the weight of each data item in the SUM query is double of that in the AVG query). Hence, query evaluation cost (as per Ri/Ci^2) of the SUM query will be half that of the AVG query (as SUM query incoherency bound is double). But, intuitively, disseminating the SUM of two data items, at double the incoherency bound should require the same number of messages as their AVG. Thus, there is a need to *normalize* query costs. From a query execution cost point of view, a query

with weights *wi* and incoherency bound *C* is same as query with weights *awi* and incoherency bound *a.C*. So, while normalizing we need to ensure that both, query weights and incoherency bounds, are multiplied by the same factor. Normalized query *sumdiff* is given by:

$$R_{query}^2 = \frac{(w_p^2 R_p^2 + w_q^2 R_q^2 + 2\rho w_p R_p w_q R_q)}{(w_p^2 + w_q^2 + 2\rho w_p w_q)} \quad (10)$$

i.e., the value of the normalizing factor should be $1/\sqrt{(w_p^2 + w_q^2 + 2\rho w_p w_q)}$. The value of the method to calculate ρ which can also be used when the corresponding data items are not being disseminated by the same data aggregator. Equation (10) can be extended to get query *sumdiff* for any general weighted aggregation query given by Equation (1) as:

$$R_Q^2 = \frac{\sum_{i=1}^{n} w_i^2 R_i^2 + 2\sum_{i=1, j=1, j\neq i}^{n} \rho_{ij} w_i w_j R_i R_j}{\sum_{i=1}^{n} w_i^2 + 2\sum_{i=1, j=1, j\neq i}^{n} \rho_{ij} w_i w_j} \quad (11)$$

3. EXCEUCTING QUERIES USING SUBQUERIES

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 need to find the optimal query execution plan which satisfies client coherency requirement with the least number of refreshes. As explained in Section 1, what we need is a mechanism to:

incoherency bound has to be adjusted by the same factor. Normalization ensures that queries with arbitrary values of weights can be compared for execution cost estimates. From Equations (9 and 10) the value of query *sumdiff* can be estimated at a data aggregator node if it has all the required data items disseminated to it. An aggregator can use interpolated values of data items to estimate ρ as it is not (always) likely to have all the data updates. In the extended version of the paper [10] we present an efficient

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 incoherency bound can be satisfied at the chosen DA.

Objective : Number of refreshes should be minimized.

Let the client query be divided into *N* sub-queries {*qk*: *1≤k≤N*}; with R_k being *sumdiff* of k^{th} sub-query and C_k being incoherency bound assigned to it. As given is Section 3, the dissemination cost of a sub-query is estimated to be proportional to $R_k/C_k{}^2$. Thus query cost estimate is given by:

$$Z = \sum_{k=1}^{N} (R_k / C_k^2) \quad (1)$$

While allocating sub-query incoherency bounds we need to ensure that the query coherency requirement C is satisfied (*condition1)*; i.e.,

$$\sum_{k=1}^{N} C_k \leq C \qquad (13)$$

For satisfying *condition2*, sub-queries should be such that all its data items can be disseminated by the chosen DA. Let X_k be the tightest incoherency bound (defined in Section 2) the chosen DA can satisfy for q_k. For the *condition3*, we have to ensure that $C_k \leq X_k$ for each sub-query q_k and its assigned data aggregator. Z needs to be minimized for minimizing the number of refreshes as per the *objective*. Before attempting the *hard* problem of optimizing Z, let us first consider a simpler problem where values of C_k are given. In this simpler problem we divide the client query into sub-queries to minimize the estimated execution cost (Z) without considering the optimal division of the query incoherency bound into sub-query incoherency bounds. Besides working as a step towards a solution for the whole problem this case can also be used where allocation of incoherency bounds to sub-queries is done independent of the data dynamics. For example, it may be pre-decided that incoherency bounds for all data items will be the same. Thus, for a given query and its incoherency bounds, the sub-query incoherency bounds can be obtained. Next we prove that this simpler version of the problem is NP-hard

CONCLUSION

This paper presents a cost-based approach to minimize the number of refreshes required to execute an incoherency bounded continuous query. We assume the existence of network of data aggregator, where each DA is capable of disseminating a set of data items at their pre-specified incoherency bounds. For optimal query execution we divide the query into sub-queries and evaluate each sub-query at a judiciously chosen data aggregator. We developed an important measure for data dynamics in the form of *sumdiff*. Performance evaluation of these sub-query execution and using the cost model for other applications and developing 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 the Internet. 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. Shah, K. Ramamritham, and P. Shenoy. Maintaining Coherency of Dynamic Data in Cooperating Repositories.\ VLDB 2002.
[5] Dynamai: Caching Technology for Dynmaic Content Revealed. www.infoworld.com/articles.
[6] Lam, W. and Ho, C.Y. Using a Generalized Instance Set for Automatic Text Categorization. SIGIR, 1998.
[7] R. Gupta, A. Puri, and K. Ramamritham. Executing Incoherency Bounded Continuous Queries at Web Data Aggregators. WWW 2005.
[8] NEFSC Scientific Computer System http://sole.wh.whoi.edu/~jmanning//cruise/serve1.cgi
[9] Query cost model validation for sensor data. www.cse.iitb.ac.in/~ravivj/BTP06.pdf.
[10] Optimized Execution of Continuous Queries, APS 2006, www.cse.iitb.ac.in/~grajeev/APS06.PDF