Global Journal of Computer Science and Technology
Network, Web & Security
Volume 12 Issue 14 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Analyzing the Query Performance Over a Distributed Network of Data Aggregators

By P. Prabhakar & S. Nageswara Rao
Madanapalle Institute of Technology and Science, AP, India

Abstract - Typically a user desires to obtain the value of some aggregation function over distributed data items. We present a low-cost, scalable technique to answer continuous aggregation queries using a network of aggregators 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 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. Performance results using real-world traces show that our cost based query planning leads to queries being executed using less than one third the number of messages required by existing schemes.

Keywords : *Content distribution network, continuous query, online decision making, data dissemination, coherency, performance.*

GJCST-E Classification : *C.2.1*

Strictly as per the compliance and regulations of:

Analyzing the Query Performance Over a Distributed Network of Data Aggregators

P. Prabhakar [α] & S. Nageswara Rao [σ]

Abstract - Typically a user desires to obtain the value of some aggregation function over distributed data items. We present a low-cost, scalable technique to answer continuous aggregation queries using a network of aggregators 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 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. Performance results using real-world traces show that our cost based query planning leads to queries being executed using less than one third the number of messages required by existing schemes.

Keywords : Content distribution network, continuous query, online decision making, data dissemination, coherency, performance.

I. Introduction

Application such as auctions, personal portfolio for financial decisions, sensors based monitoring, route planning based on traffic information, etc., make extensive use of dynamic data. For such applications, data from one or more independent data sources may be aggregated to determine if some action is warranted. Given the increasing number of such applications that make use of highly dynamic data, there is significant interest in systems that can efficiently deliver the relevant updates automatically. 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 nodes 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. In data dissemination schemes proposed in literature [4, 11], a hierarchical network of data aggregators is employed such that 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. Although CDNs use page-purge [8] based coherency management, we assume that in dynamic data dissemination networks, these messages carry the new data values thereby an invalidation message becomes a refresh message. For maintaining a certain incoherency bound, a data aggregator gets data updates from the data source or some higher level data aggregator so that the data incoherency is not more than the data incoherency bound. In a hierarchical data dissemination network a higher level aggregator guarantees a tighter incoherency bound compared to a lower level aggregator. Thus, data refreshes are pushed from the data sources to the clients through the network of aggregators.

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 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 $|u_i(t)-v_i(t)|$. For a data item which needs to be refreshed at an incoherency bound C a

Author α : M.Tech Student, Dept of CSE, Madanapalle Institute of Technology and Science, AP, India.
E-mail : prabhakar.mth@gmail.com
Author σ : Assistant Professor, Dept of CSE, Madanapalle Institute of Technology and Science, AP, India. E-mail : nag_sirisala@yahoo.com

data refresh message is sent to the client as soon as data exceeds C, i.e., $|u_i(t)-v_i(t)|>C$.

Network of data aggregators: Data aggregators are one kind of secondary server it serves as data sources (data items). The data refreshes can be done using two mechanisms. *(a)Push* based mechanism data source send update messages to client on their own. *(b)Pull* based mechanism data sources send messages to the client only when client makes a request. For scalable handling of push based data dissemination, network of DA's are proposed in the literature [12,15,16].

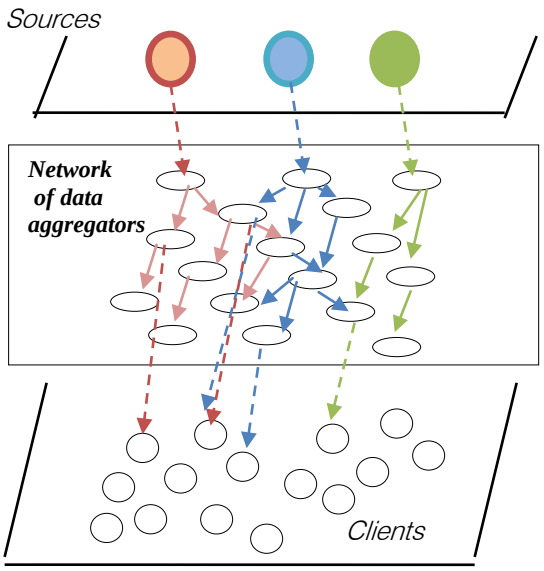

Figure 1 : Data dissemination network for multiple data items

In such network of DA's, data refreshes occur from data sources to the client through one or more DA's. In this paper we assume that each DA maintains its configured incoherency bounds for various data items. Dissemination networks for various data items can be overlaid over a single network of data aggregators as shown in Figure 1. Thus, 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 the DA can disseminate at an incoherency bound C_i. The configured incoherency bound of data item at a DA 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 is scalability problems. (b) data aggregators with tighter incoherency bound help the DA to maintain its incoherency bound in scalable manner as explained in [4,7].

Example 1: In a network of data aggregators managing data items *D1-D4*, various aggregators can be characterized as-

A1: {(*D1*, 0.5), (*D3*, 0.2)}
A2: {(*D1*, 1.0), (*D2*, 0.1), (*D4*, 0.2)}

Aggregator *A1* can serve values of *D1* with an incoherency bound greater than or equal to 0.5 whereas *A2* can disseminate the same data item at a looser incoherency bound of 1.0 or more. Usually, client is interested in an aggregation of these dynamic data items at a certain incoherency bound. These continuous queries are used to monitor changes in dynamic data and provide results useful for online decision-making. For generating the result of a query, data from multiple sources is required. As a result, the query has to be evaluated either at data aggregators or at the client.

In this work, our aim is to satisfy the client's query requirements while minimizing the query execution cost in terms of number of dissemination messages. Towards that end, we have achieved the following:

1. Developed techniques for estimating the cost of disseminating a data item, at specified incoherency bound.
2. Using the estimated data dissemination cost, we developed query cost model for estimating the cost of executing an incoherency bounded continuous query.
3. Used the query cost model for assigning a client query to one or more data aggregators so that the query can be executed with the least number of messages.

Our work involves dividing the client query into sub-queries and allocating it to different data aggregators for optimal execution. In comparison, all the related work in literature [3,5] propose getting individual data items from the aggregators which, as we show in this report, leads to large number of dissemination messages. In the rest of the Introduction, we present basic concepts underlying incoherency bounded continuous query execution using a distributed network of data aggregators.

a) Problem Statement and Contributions

In this paper, we develop query cost model for aggregation query involving multiple data items:
- Additive aggregation with each data item possibly different weights, and - MIN/MAX aggregation queries. The weighted aggregation query can be mathematically written as:

$$v_s^q(t) = \sum_{i=1}^{i=n^q} s_i(t) \times w_i^q \qquad (1)$$

V_s^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_i^q, $1< i< n^q$. $s_i(t)$ is the value of the i^{th} data item at the data source at time t. Such a query encompasses SQL aggregation operators SUM and AVG besides general weighted aggregation queries such as portfolio queries, involving aggregation of stock prices, weighted with number of shares of stocks in the portfolio. Due to

space limitations, we are not presenting execution schemes for other aggregation queries such as MIN/MAX. Interested readers are referred to [13] for the extended version of this paper.

Let the value of i^{th} data item, in Equation (1), known to the client/DA be $d_i(t)$. Then the data incoherency is given by $|s_i(t)-d_i(t)|$. For a data item which needs to be disseminated at an incoherency bound C the data refresh is sent to the client or lower level DA, if the $|s_i(t)- d_i(t)|$ is more than C. If user specified incoherency bound for the query q is C^q, then the dissemination network has to ensure that:

$$\sum_{i=1}^{n^q}\left(s_i(t)-d_i\right)\times w_i^q \le c^q \qquad (2)$$

Whenever data values at sources change such that query incoherency bound is violated, the updated value(s) is disseminated to the client. If the network of aggregators can ensure that the i^{th} data item has incoherency bound C_i then the following condition ensure that the query incoherency bound C^q is satisfied:

$$\sum_{i=1}^{n^q} c_i \times w_i^q \le c^q \qquad (3)$$

For additive aggregation queries, a client specified query incoherency bound needs to be translated into incoherency bounds for individual data items or sub-queries such that these satisfy Equation (3).

MIN/MAX queries involve set of data items, whose extremes are the required result, and its incoherency bound. In a MIN (MAX) query, even if one data value changes it is possible that that value is minimum (maximum) thus individual data incoherency bound cannot be more than query incoherency bound. Thus in case of MIN/MAX queries the dissemination network has to ensure that $Ci \le Cq$ for all the data items appearing in the query.

b) Summary of Distributed Execution approach

Consider a client query Q=50 $D1$ + 200 $D2$ + 150 $D3$ with a required incoherency bound of 80 (in a stock portfolio $D1$, $D2$, $D3$ 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 $D1$, $D2$ and $D3$ separately. The query incoherency bounds can be divided among data items in various ways while satisfying Equation 3. In this report, 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 $D1$ + 200 $D2$+ 150 $D3$) 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 Q can be divided in two alternative ways:

Plan1: $A1$ {50 $D1$ + 150 $D3$}; $D2$ {$D2$}

Plan2: $A1$ {$D3$}; $D2$ {50 $D1$, + 200 $D2$}

i.e., in plan1 result of sub-query 50 $D1$ + 150 $D3$ is served by $A1$ whereas value of (or 200 $D2$) by $D2$ is served by $A2$. In plan2, value of $D3$ is served by $A1$ whereas result of sub-query 50 $D1$ + 200 $D2$ is served by $A2$. 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 cancelling each other as that will reduce the need for refresh to the client (Equation 2). In the above example, if updates to $D1$ and $D3$ are such that when $D1$ increases, $D3$ 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 $D1$ + 200 $D2$ + 150 $D3$ is divided into two sub-queries 50 $D1$ + 130 $D2$ and 70 $D2$ + 150 $D3$); 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 $50D1 + 150D3$ 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. And (d) total query execution cost, in terms of number of refreshes, is minimized.

II. Data Dissemination Cost Model

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

a) Incoherency bound model

The 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 [13] that the number of data refreshes is inversely proportional to the square of the incoherency bound ($1/C2$). Similar result was earlier reported in [4] where the data dynamics was modeled as a random walk process.

$$\textit{Data dissemination cost} \propto 1/C2 \tag{4}$$

b) Data synopsis Model

The 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 data synopsis, called sumdiff, defined as:

$$R_s = \sum_i \left(s_i - s_{i-1} \right) \tag{5}$$

Where S_i and S_{i-1} are the sampled values of the data item at i^{th} and $(i-1)^{th}$ time instances (consecutive ticks). Data sumdiff can be maintained at the source or aggregators. For calculating this quantity, the data source can accumulate the absolute value of changes in data items or the data aggregator can estimate this quantity using changes in pushed values. Next we use this result for developing the query cost model.

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 query *sumdiff* will be:

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

Instead, if the aggregator uses the information 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 $w_{pP} + w_q q$ and disseminates that data item then the query *sumdiff* will be:

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

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.

III. Query Cost Model

For getting an estimation of the query dissemination cost what we need is R_{query} whereas we know R_p and R_q (in Equation (6) and (7). As different data items may be disseminated by different servers, R_{query} can be calculated only at data aggregators. If two data items are correlated such that if value of one data item increases other also increases, then R_{query} will be closer R_{data} ; whereas if the data items are inversely correlated then R_{query} will be much less than R_{data}. Thus, institutively, we can represent the relationship between R_{query} and *sumdiff* of individual data items involved using a correlation measure between data items. Specifically, if ρ is the correlation measure then R_{query} can be written as:

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

The correlation measure is defined such that– $1 \leq \rho \leq +1$, so, *Rquery* will always be less than $| w_p R_p + w_q R_q |$ (as explained earlier) and always be more than $| w_p R_p - w_q R_q |$.

The correlation measure ρ can be interpreted as cosine similarity [19] 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:

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

a) Executing queries using sub queries

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. What we need is a mechanism to:

Task 1: Divide the aggregation query into sub-queries; and

Task 2: Allocate the query incoherency bound among them.

While satisfying the following conditions identified in Section 1.2:

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.

b) Minimum Cost

Figure 2 shows the outline of greedy heuristics where different criteria (ψ) can be used to select sub-queries. In this section we describe the case where the estimate of query execution cost is minimized in each step of the algorithm (min-cost) whereas in the next section we present the case where gain due to executing a query using sub-queries is maximized (max-gain).

c) Query Plan with Pre-decided Incoherency Bound Allocation

For the given client query (*q*) and mapping between data aggregators and the corresponding {*data-item*, *data incoherency bound*} pairs ($f: D \rightarrow (S, C)$) maximal sub-queries can be obtained for each data aggregator. Let *A* be the set of such maximal sub queries. In this set, each query $a \in A$ can be disseminated by a designated data aggregator at the assigned incoherency bound. For each sub-query $a \in A$, its *sumdiff Ra* is calculated. Using the set *A* and sub-query *sumdiffs*, we use the algorithm outlined in Figure 2 to get the set of sub-queries minimizing the query cost. In this Figure each sub-query $a \in A$ is represented by the set of data items covered by it. As we need to minimize the query cost, a sub-query with *minimum cost per data item* is chosen in each iteration of the algorithm i.e., criteria $\psi \equiv$ minimize ($Ra/Ca^2/a/$).

All data items covered by the selected sub query are removed from all the remaining sub-queries in *A* before performing the next iteration.

Algorithm:

```
Result ←∅
 while A ≠ ∅
choose a sub-query a∈A with criteria ψ
      Result ←Result ∪a
     A←A-{a}
       for each data element e ∈ a
          for each b ∈ A
                b←b-{e}
         if b =∅
               A←A-{b}
         else
             Calculate sumdiff for modified b
Return Result
```

Figure 2 : Greedy algorithm for query plan selection

The decision is taken based on client query information. The greedy method is the most straight forward method. It is popular for obtaining the optimized solutions. In the greedy method there are some important activities. (a) A selection of solution from the given input domain is performed. (b). The feasibility of the solution is performed and then all the feasible solutions are obtained. (c) From the set of feasible solutions, the particular solution that minimizes or maximizes the given objective function is obtained. Such a solution is called optimal solution. For an algorithm that uses greedy method works in stages. At each stage only one input is considered at a time. Based on this input it is decided whether particular input gives the optimal solution or not.

d) Maximum Gain

In this section we present an algorithm which, instead of minimizing the estimated query execution cost, maximizes the estimated gains of executing client query using sub-queries. In this algorithm, for each sub-query, we calculate the relative gain of executing it by finding the *sumdiff* difference between cases when each data item is obtained separately and when all the data items are aggregated as a single sub-query. (i.e., maximal sub-query).

IV. Related Work

We divide the related work on scalable answering of aggregation queries over a network of data aggregators in to two interrelated topics.

a) Answering Incoherency bounded aggregation queries

Various mechanism for efficiently answering incoherency bounded aggregation queries over continuously changing data items are proposed in the literature [3,4,9]. i.e., in this thesis, to develop and evaluate client-pull-based techniques for refreshing data so that the results of the queries over distributed data can be correctly reported, conforming to the limited incoherency acceptable to the users. Here considered the problem of answering queries for online decision making at web data aggregators.

Our work distinguishes itself by employing sub-query based evaluation to minimize number of refreshes. Pull based data dissemination techniques, where client or data aggregators pull data items such that query requirements are met, are described in [3]. For minimizing the number of pulls, both predict data values and pull instances. In comparison, we use push based mechanism to refresh sub-query values at the client. In [4], authors propose push based scheme using data filters at the sources. i.e., distributed data sources continuously stream updates to a centralized processor that monitors continuous queries over the distributed data. Based on we specified a new approach for reducing communication cost in an environment of centralized continuous query processing over distributed data streams.

This approach hinges on specifying precision constraints for continuous queries, which are used to generate adaptive filters at remote data sources that significantly reduce update stream rates while still guaranteeing sufficient precision of query results at all times. And enables users or applications to trade precision for lower communication cost at a fine granularity by individually adjusting precision constraints of continuous queries. Imprecision of query results is bounded numerically so applications need not deal with any uncertainty. To validate our approach we performed a number of experiments using simulations and a real network monitoring implementation approach in achieving low communication overhead. According to that work can an aggregation query, the number of refresh messages can be minimized by performing incoherency bound allocation to individual data items such that the number of messages from different data sources is the same. Instead we execute more dynamic assigning incoherency bounds. And minimizing the total number of messages send by DAs. Like us ,authors of [9],also assume that dissemination tree from sensor node[data source] to root[client]already exist; and they also install error filters on partial aggregates (similar to in coherency bound assign to sub queries) but, in our work each data aggregator can only discriminates data at some pre-specified incoherency bound depending on its capability where as such a constraints does not exist for [9].further, we also be give method to select partial aggregates (sub queries)to be used to answering the query.

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, Further, we propose that more dynamic data items should be executed as part of larger sub-query. In [8], i.e., here discuss various techniques 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.

Like us, author of [9] also assume that dissemination tree from sensor nodes (data- sources) to root (clients) already exists. In-network data aggregation has been recently proposed as an effective means to reduce the number of messages exchanged in wireless sensor networks. Nodes of the network form an aggregation tree, in which parent nodes aggregate the values received from their children and propagate the result to their own parents. However, this schema provides little flexibility for the end-user to control the operation of the nodes in a data sensitive manner. For large sensor networks with severe energy constraints, the reduction (in the number of messages exchanged) obtained through the aggregation tree might not be sufficient. In this thesis we present new algorithms for obtaining approximate aggregate statistics from large sensor networks. The user specifies the maximum error that he is willing to tolerate and, in turn, our algorithms program the nodes in a way that seeks to minimize the number of messages exchanged in the network, while always guaranteeing that the produced estimate lies within the specified error from the exact answer. And they also install error filters on partial aggregates. But in our work, each data aggregators can only disseminate data some pre-specified incoherency bound depending on its capability whereas such a constraint does not exist for [9].

Further, we also give a method to select partial aggregates (sub queries) to be used for answering the query. In [12] Pull based data dissemination techniques, where clients or data aggregators pull data items such that query requirements are met, are described in [3]. i.e., we develop and evaluate client-pull-based techniques for refreshing data so that the results of the queries over distributed data can be correctly reported, conforming to the limited incoherency acceptable to the users. 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 [5,6]. We also consider correlation in terms of cosine similarity between data items, but we use it for dividing client query into sub-queries.

b) Construction and maintenance of network of data aggregators

Authors of [1,2,8] describe Construction and maintenance of hierarchical network of data aggregators for providing scalability and fidelity in disseminating dynamic data items to large number of clients.. In these works, fidelity is defined as fraction of time when the client coherence requirements are met. Each data aggregators is given client requirements in the form of data items and their respective incoherency bounds. Instead we use such networks for efficiently answering client's aggregation queries. One can use client queries to optimally construct a network of data aggregators while, on the other hand, one can also use a given network of aggregators to efficiently answer client queries. Authors of [1,2,8] deal with the first part where as we have studied the second part. Changes in data dynamics may lead to reorganization of the network of data aggregators which, in turn necessitate changes in query plans. Whereas query plan can change more often depending on data dynamics.

Instead of optimizing fidelity of data items at data aggregators, as proposed in [2], using our work, one can optimize fidelity all the way up to client queries. Fidelity of a data item can be approximately calculated as number of dissemination messages multiplied by the total delay in the message transmission. Author of [2] assume that each client's requirements are fulfilled by a single data aggregator. But in case of data aggregators may need to disseminate a large number of data items which will lead to processing large number of refresh messages, hence increase in delay. Thus , each client getting all its data items from a single data aggregators(using a single sub-query) is optimal from number of messages point of view but not necessarily from the query fidelity point of view. By using our work, one can model expected number of messages for client query. Thus, our work can complement the of [2] for end-to-end (source-to-client) fidelity optimization.

V. Conclusion and Future Work

In this literature 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.

Developing efficient strategies for multiple invocations of our algorithm, considering hierarchy of data aggregators. Another area for future research is changing a query plan as data dynamics changes. Another area of our future work is using the cost model for these applications and developing the cost model for more complex queries.

References Références Referencias

1. A. Davis, J.Parikh and W.Weihl. "Edge Computing: Extending Enterprise Applications to the Edge of the Internet". WWW 2004.
2. D. Vander Meer, 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. Query cost model validation for sensor data.www.cse.iitb.ac.in/~ravivj/BTP06.pdf.
6. C. Olston, J. Jiang, and J. Widom. Adaptive Filter for Continuous Queries over Distributed Data Streams. SIGMOD 2003.
7. D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing, vol. 11 (3), 1982.
8. Zongming Fei. A Novel Approach to Managing Consistency in Content Distribution. WCW 2001.
9. R. Gupta, A. Puri, and K. Ramamritham. Executing Incoherency Bounded Continuous Queries at Web Data Aggregators. WWW 2005.
10. NEFSC Scientific Computer System http:// sole. wh. whoi.edu/jmanning//cruise/serve1et.cgi
11. Pearson Product moment correlation coefficient. http://www.nyx.net/~tmacfarl/STAT_TUT/correlat.ssi /.
12. S. Agrawal, K. Ramamritham and S. Shah. Construction of a Temporal Coherency Preserving Dynamic Data Dissemination networks", RTSS 2004
13. Optimized Execution of Continuous Queries, APS2006,www.cse.iitb.ac.in/~grajeev/APS06.PDF
14. S. Rangarajan, S. Mukerjee and P. Rodriguez. User Specific Request Redirection in a Content Delivery

Network, 8th Intl. Workshop on Web Content Caching and Distribution (IWCW), 2003.
15. R Guptha and K. Ramamritham, "Optimized Query Planning of Continuous Aggregation Queries in Dynamic Data Dissemination Networks", WWW 2007.
16. R Guptha and K. Ramamritham, "Query Planning for Continuous Aggregation Queries over a network of Data Aggregators", IEEE 2011.
17. S. Shah, K. Ramamritham, and C. Ravishankar. Client Assignment in Content Dissemination Networks for Dynamic Data, VLDB 2005.