ISSN: 2321-7782 (Online)
Volume 2, Issue 9, September 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Active and Actionable User Profile based Query Planning over a Network of Data Aggregators in Shared Environment

Varada Venkata SubbaRao[1]
Computer Science & Engineering
Sri Aditya Engineering College
Kakinada – India

N.Venkata Ramana[2]
Computer Science & Engineering
Sri Aditya Engineering College
Kakinada – India

Abstract: ***Now a days, software industries moving towards developing challenging applications to fulfill the business (client) needs for taking online decision making to gain their profitability. In this paper, we are developing a user profile based model which minimizes the number of messages required to satisfy the client specified incoherency bound for time varying data in a shared environment. Continuous queries are used to monitor changes to time varying data and using a network of aggregators of dynamic data items to provide results. Each data aggregator serves a set of data items at specific coherencies. our technique enhances involves decomposing a client query into subqueries and executing subqueries on judiciously chosen data aggregators with their individual subquery incoherency bounds and sharing the user results to user groups according to user profiles. Group profiles are the mechanism used to reduce the message refreshes by sharing the result data to different user profiles (clients).This reduces the required number of quires by eliminating the duplicate quires instead of previous system models. We provide a enhanced technique for getting the optimal set of subqueries with their incoherency bounds which satisfies client query's coherency requirement with least number of refresh messages sent from aggregators to the client and shared by the users of different profiles belongs to specific user groups. 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.***

Keywords-User profile, Algorithms, continuous queries, distributed query processing, data dissemination, coherency, performance.

I. INTRODUCTION

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 vi(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(t). Then, the data incoherency at the client is given by |vi(t) - ui(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., $| vi(t) - ui(t)| > C$.

Network of Data Aggregators (DA) over Group Profiles: 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 a pull-based mechanism data sources send messages to the client only when the client makes a request. We assume the push-based mechanism for data transfer between data sources and User profile (clients) which are again mapped with the Group profile. A User profile is a mechanism which specifies incoherency bound incorporate in Group profile. A Group profile is a method of accepting the User profile requests and maintains the group queries and run against the different data aggregators. Group profiles provide the results near to the client coherence bound. so, user(client) can analyze the result according to his appropriate coherence bound. In such network of data aggregators, data refreshes occur from data sources to the clients through one or more data aggregators.

User Profile P = {P1, P2, P3…..N}, Group Profile G= {G1, G2, G3….N}, PXG is network of queries run against different data aggregators. A client (User Profile) may run the query independently to get their result is a previous model, or batch process (group profile) current method to get the required result and also for the other users(clients) results according to their incoherence bounds which can be share in the group region.

In this paper , we provide shared updated results of different user profiles by running individual queries on different data aggregators, Every user(client) in group profile can view results based on the different user thresholds of different users , which can periodically reduces refresh rate against data aggregators and increases performance of the application. Pre defined user groups are defined based on constraints required viewing the results for simulation purpose, user with different profile can select one or more user groups containing set of different client query's coherency requirements nearest to the user intended coherence bound.

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 is characterized by a set of (di,ci) pairs, where di is a data item which the DA can disseminate at an incoherency bound ci. The configured incoherency bound of a data item at a data aggregator can be maintained using any of following methods: 1) 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. 2) Data aggregator(s) with tighter incoherency bound help the DA to maintain its incoherency bound in a scalable manner as explained in [5], [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) }

a1: {(d1, 1.0), (d3, 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. 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 d1 (incoherency bound of d1 at a_i is tighter than that at a_k) but the node ai gets values of another data item d2 from a_k.

1.1 Data Scheduling Over Different User Profiles:

The USER PROFILE based Query planning over network of data Aggregators architectures are shown as in Figure 1. In this paper, we present a method for allocating the results according to their user profiles by selecting group profile and this method can reduce the user intended continuous queries to run on different data aggregators. The users in any group can share the results of that group of clients queries coherency requirements and by analyzing the results the user(client) can take online decision making. The group profile management explained in the figure 1. The user group maintains the users selected by specific group profile. By using this method the user need not to query the remaining coherence bounds which can be taken dynamically by the other clients coherence bound. This method minimizes the message refreshes over different data aggregators because of selecting user group .The user group contains one or more user profiles based on the client requirement. User profiles are the clients interesting data about the user who interested in set of data result with different incoherency bounds.

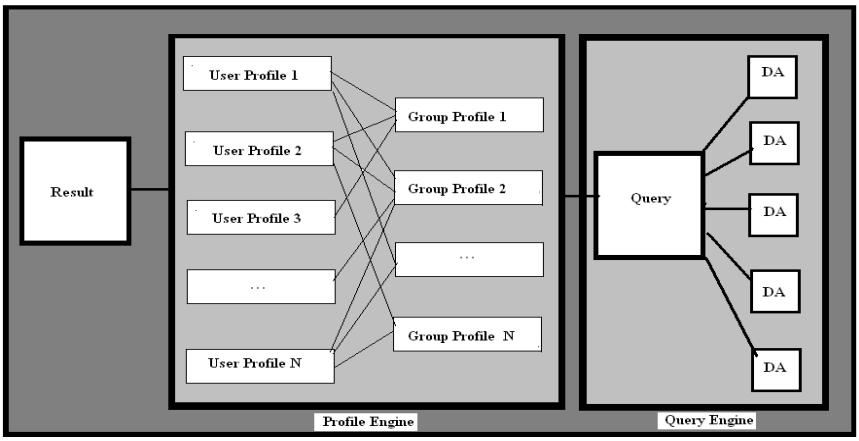


Figure 1. User Profile Based Query Planning over a Network of Data Aggregators

Users can select required specified incoherency bound; the query can select the path of the group profile to decomposing a client query into subqueries and executing subqueries on judiciously chosen data aggregators with their individual subquery incoherency bounds and sharing the user results to user groups according to user profiles.

In this paper, we present a method for executing continuous multidata aggregation queries, using a network of data aggregators, with the objective of minimizing the number of refreshes from data aggregators to the client by sharing the information in a shared environment to specific user groups containing different users (clients) users. First, we give two motivating scenarios where there are various options for executing a multidata aggregation query and one must select a particular option to minimize the number of messages.

1.2 Problem Statement and Contributions

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

$$V_q(t) = |\sum (vqi(t) - u_{qi}(t)) \times w_{qi}) \quad (1)$$

where Vq is the value of a client query q involving nq data items with the weight of the ith data item being $w_{qi}, 1<i<n_q$. 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. Suppose the result for the query given by (1) needs to be continuously provided to a user at the query incoherency bound C_q. Then, the dissemination network has to ensure

$$|\sum_{i=1}^{n_q} (v_{qi}(t) - u_{qi}(t)) \times w_{qi}| \leq C_q \quad (2)$$

Whenever data values at sources change such that query incoherency bound is violated, the updated value should be refreshed to the client. If the network of aggregators can ensure that the ith data item has incoherency bound Cqi, then the following condition ensures that the query incoherency bound Cq is satisfied

$$\sum_{i=1}^{n_q} (C_{qi} \times w_{qi}) \leq C_q \quad (3)$$

The client specified query incoherency bound needs to be translated into incoherency bounds for individual data items or subqueries such that (3) is satisfied. All these refreshes are minimized in user shared environment to fulfill to increase the performance of data aggregators to serve for other clients quires.

1.3 Outline of the Paper

The cost model for data dissemination is developed and we present the query cost model for the additive aggregation queries over shared network in Section 2. It uses the data dissemination model and a measure for capturing correlation between data dynamics. Optimal query planning for additive queries is presented in Section 3. Results of performance evaluations of algorithms described in Section 4 and also optimal query planning for MAX queries. Most conclusions drawn for this class of queries are similar to that for additive aggregation queries. Related work is presented in Section 5. 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 SHARED REGION

In this section, we present the model to estimate the number of refreshes required to disseminate a data item while maintaining a certain incoherency bound in a group of user profile. There are two primary factors affecting the number of messages that are needed to maintain the coherency requirement: 1) the coherency requirement itself and 2) dynamics of the data.

Step 1. Incoherency Bound Model in Clients Group

Consider a data item which need 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.

Thus, the number of dissemination messages will be proportional to the probability of |v(t) –u(t)|greater 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 with its previous step. In a push-based dissemination, a data source can follow one of the following schemes.

Thus, the number of dissemination messages will be proportional to the probability of |v(t) –u(t)|greater 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 with its previous step. In a push-based dissemination, a data source can follow one of the following schemes.

Table.1 Important Symbols and Their Meaning

Symbols	Description
A	Set of aggregators in the network
N	Number of data aggregators(Das)
D	Set of data items disseminated by the network
C	Incoherency bounds of data items
ak	Kth data aggregator 1≤ki≤N
Dk	Set of data items disseminated by the kth DA
dkj	Jth data item disseminated by the kth DA
tkj	Incoherence bound which ak can ensure
q	Client query
Cq	Incoherence bound for q
nq	Number of data items in q
dqi	Ith data item of the query q
Vqi(t)	Value of the query q at time t
qk	Sub-query of q to be executed at ak
cqk	Incoherence bound of qk
Rq	Sumdiff of the query q
p	Correlation measure between data items

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

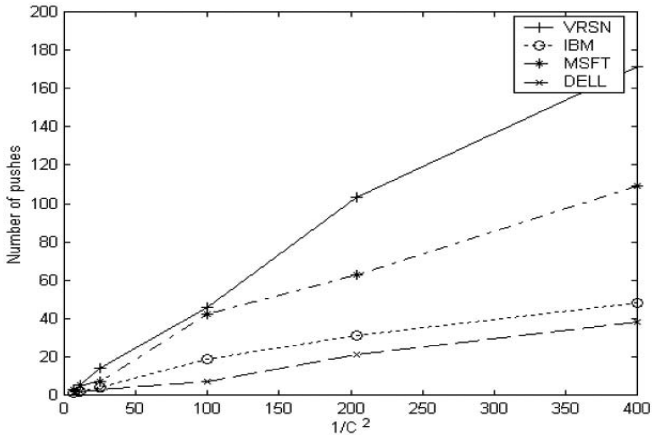


Fig 2..Number of pushes versus incoherency bounds.

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 user share environment. In case 2, the client and the server estimate the data value as the mean of the modeled random process, whereas in case 1 deviation from the last pushed value can be modeled as zero mean process. Using Chebyshev's inequality [10].

P (|v(t) – u(t)| >C)α1/C2 (4)

Step2. Data Dynamics Model In Shared Region

Suppose data values in consecutive instances for a data item d1 are { 0,4,0,4,0,4,0,4}. whereas for another data item d2 values are {0,0,0,0,4,4,4,,4} Suppose both data items are disseminated with an incoherency bound of 3. It can be seen that the number of messages required for maintaining the incoherency bound will be 7 and 1 for data items d1 and d2, respectively, whereas both data items have the same standard deviation (=2.138). Thus, we need a measure which captures data changes along with its temporal properties. This motivates us to examine the second measure.

As a second option we considered Fast Fourier Transform (FFT) which is used in the digital signal processing domain to characterize a digital signal. FFT captures number of changes in data value, amount of changes, and their timings. We hypothesize that the cost of data dissemination for a data item can be approximated by a function of the first FFT coefficient. Specifically, the cost of data dissemination for a data item will be proportional to data *sumdiff* defined as

Rs = ∑|si – si-1| (5)

where s_i and s_{i-1} are the sampled values of a data item S at i^{th} and $(i-1)^{th}$ time instances (i.e., consecutive ticks). In practice, *sumdiff* value for a data item can be calculated at the data source by taking running average of difference between data values for consecutive ticks. For our experiments, we calculated the *sumdiff* values using exponential window moving average with each window having 100 samples and giving 30 percent weight to the most recent window. Validating the hypothesis. We did simulations with different stocks being disseminated with incoherency bound values of [1].

Step 3. Combining Data Dissemination Models in shared model

Number of refresh messages is proportional to data sumdiff Rs and inversely proportional to square of the incoherency bound (C2). Further, we can see that we need not disseminate any message when either data value is not changing (RS=0)or incoherency bound is unlimited (1/C2=0). Thus, for a given data item S, disseminated with an incoherency bound C, the data dissemination cost is proportional to Rs/C2 In the next section, we use this data dissemination cost model for developing cost model for additive aggregation queries.

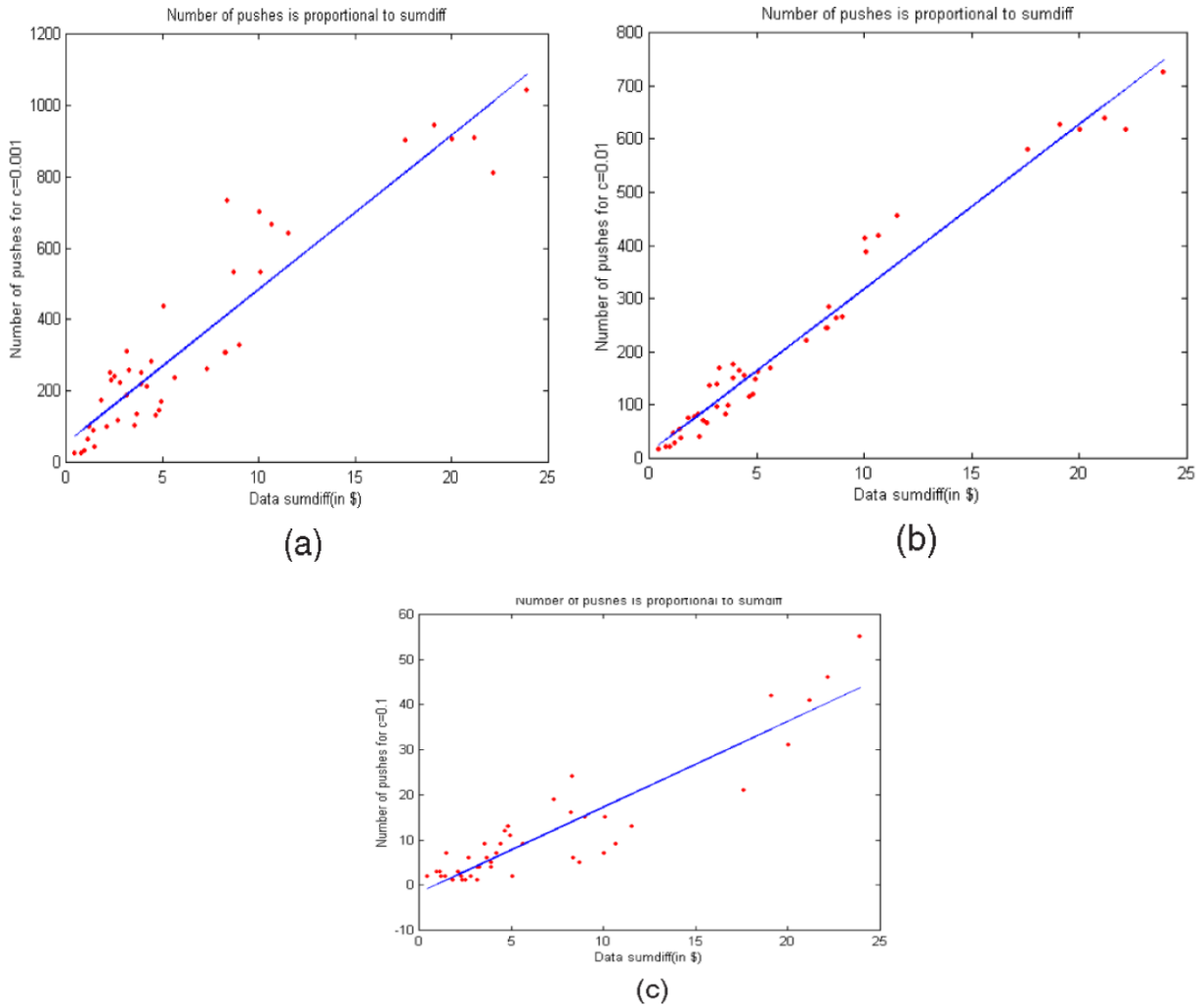


Fig. 3. Number of pushes versus data sumdiff (a) C = 0.001, (b) C = 0.01, and (c) C = 0.1.

Step 4. Cost Model for additive aggregation queries over different user profiles

Consider an additive query over two data items P and Q with weights wp and wq, respectively; and we want to estimate its dissemination cost. If data items are disseminated separately, the query sumdiff will be

Rdata = wpRp + wqRq = wp∑|pi-pi-1| + wq∑|qi-qi-1| (6)

Rquery= ∑|wp (pi-pi-1) + wq (qi-qi-1)|, (7)

III. Query Planning For Weighted Additive Aggregation Queries In Shared Region

For executing an incoherency bounded continuous query in shared region, a query plan is required. The query planning problem can be stated as:

Step 1. Inputs. 1. A network of data aggregators in the form of a relation F(A,D,C,SR) specifying the N data aggregators ak ε (A1≤K ≤N) set Dk subset D of data items disseminated by the data aggregator ak and incoherency bound tkj and shared region SR which the aggregator ak can ensure for each data item dkj ε Dk.

2. Client query q and its incoherency bound Cq. An additive aggregation query q can be represented as $\sum wq_i dq_i$, where wqi is the weight of the data item dq_i for $1 \leq i \leq n_q$.

Outputs. 1. *qk* for *1 ≤k ≤N,* i.e., subquery for each data aggregator ak. C_{qk} for $1 \leq k \leq N$, i.e., incoherency bounds for all the subqueries. Thus, to get a query plan we need to perform following tasks:

1. Determining subqueries: For the client query q get subqueries qks for each data aggregator[1].

2. Dividing incoherency bound: Divide the query incoherency bound Cq among subqueries to get Cqk s[1]. For optimal query planning, above tasks are to be performed with the following objective and constraints[1]:

1. Determining subqueries: For the client query q get subqueries qks for each data aggregator.

2. Dividing incoherency bound: Divide the query incoherency bound Cq among subqueries to get Cqk s.

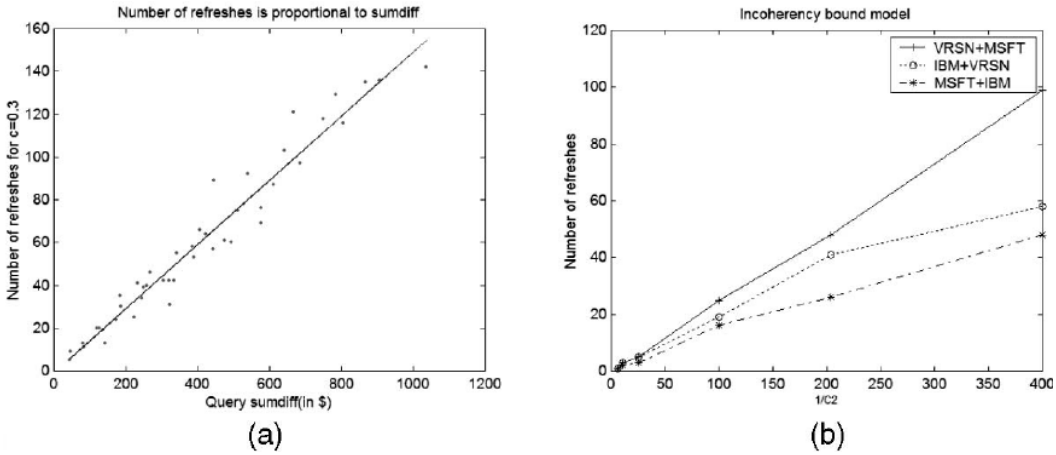


Fig. 4. Query cost validation with varying (a) Sumdiff and (b) Incoherency bound.

*Step2.*Greedy Heuristics for Deriving the Subqueries in shared region as follows:

The following algorithm 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 $=50d_1 + 150d_3$, 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,SR)) 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. For each subquery m belongs to Mq, its sumdiff Rm can be calculated using (12). Different criteria (Ψ) 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 DAs can be null..

Now we describe two criteria (Ψ) for the greedy heuristics; 1) min-cost: estimate of query execution cost is minimized[1], and 2) max-gain: estimated gain due to executing the query using subqueries is maximized[1]. Minimum Cost Heuristic, Satisfiability of Subquery Incoherency Bound, Maximum Gain Heuristic.

Greedy algorithm for query plan selection:

Result ← Φ;

GP← Φ;

UP← Φ;

while Mq≠ Φ;

SRMq ≠ Φ;

choose a sub-query mi ε Mq with criteria Ψ;

result← result mi; Mq ← Mq – {mi};

for each data item d ε mi;

for each mj ε Mq

mj←mj-{d};

if mj = Φ Mq←Mq-{mj};

else calculate sumdiff for modified mj;

return result

SRMq← result;

GP←SRMq;

UP ←GP;

IV. PERFORMANCE EVALUATION

For performance evaluation we simulated a network of data aggregators of 200 stock data items over 100 aggregator nodes such that each aggregator can disseminate combinations of 25 to 50 data items and four Group profiles and 10 user profiles.

Data items were assigned to different aggregators using zipf distribution (skew = 1.0) assuming that some popular data items will be disseminated by more DAs. Data incoherency bounds, for various aggregator data items, were chosen uniformly between $0.005 and 0.02. We created 500 portfolio queries such that each query has 10 to 25 randomly (using zipf distribution with the same default skew) selected data items with weights varying between 2 and 10. These queries were executed with incoherency bounds between 1.0 and 3.0 (i.e., 0.02-0.07 percent of the query value). Although here we present results for stock traces (man-made data), similar results were obtained for sensor traces (natural data) as well [8].

A. 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 subquery values from DAs. Set of subqueries can be selected using sumdiff-based approaches or any other random selection. Subquery (or data) incoherency bound can either be predecided or optimally allocated. Various combinations of these dimensions are covered in the following algorithms:

1.No subquery, equal incoherency bound (naı¨ve), 2. No subquery, optimal incoherency bound (optc), 3. Random subquery selection (random), 4. Subquery selection while minimizing sumdiff (mincost),5. Subquery selection while maximizing gain (maxgain). Fig. 6 shows average number of refreshes required for query incoherency bounds of $1-$3.. Now we report the time overheads for various query planning operations. We measured these costs by varying the number of data items being disseminated by the network, between 40 and 200. These experiments were done on a WindowsXP machine with 2.53 GHz Intel Core- Duo CPU and 3 GB RAM. For various sumdiff-based algorithms, we need to maintain the sumdiff values of various data items (proportional to the number of data items being disseminated) and the correlation measure for each pair of data items (proportional to the square of the number of data items), in addition to the query dependent planning cost. For a trace size of 10,000 (for each data item)

Query planning cost (time required to derive subqueries and their associated incoherency bounds) for naı¨ve and optc algorithm was found to be approximately 1 microsecond per query, whereas the same for the random, minCost, and maxGain algorithms was found to be 2.5, 2.2, and 1.7 milliseconds.

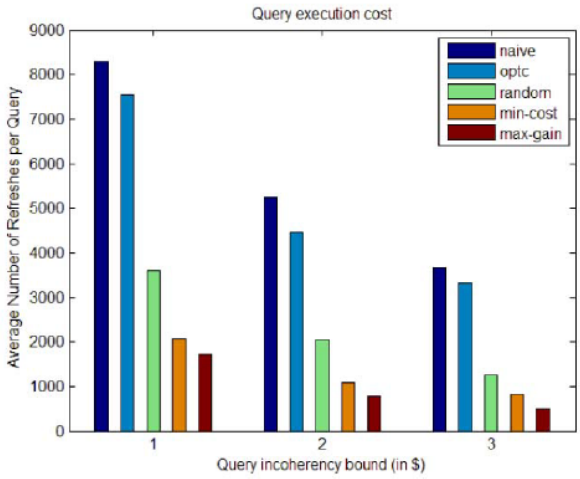


Fig.6 Performance evaluation of algorithms

B. Query planning for MAX queries:

In this section, we briefly describe the optimal query planning for MAX queries. MIN queries can be handled in the similar manner. A MAX query, where a client wants the maximum of a specified set of data item values, can be written as :

$$Vq(t) = max(vqi(t), 1 \leq I \leq nq) \quad (8)$$

the following condition ensures that the query incoherency bound Cq is satisfied

$$Ci \leq Cq, vi, 1 \leq i\ nq. \quad (9)$$

Query Cost Model:

Let us consider a query Q = max(A;,B), which is used for disseminating max of data items A and B from a data aggregator. Let the sumdiff values of A and B is Ra and Rb, respectively. For a MAX query, the query result is the maximum of data item values. Thus, the query dynamics is decided as per the dynamics of the data item with the maximum value. Hence, the query sumdiff is nothing but weighted average of data sumdiffs, weighted by fraction of time when the particular data item is maximum.

$$R_q \approx \sum_{J=1}^{n_q} R_i(\Pi\, p(xi > x_j)) \leq max(R_i | 1 \leq i \leq n_q), \quad (10)$$

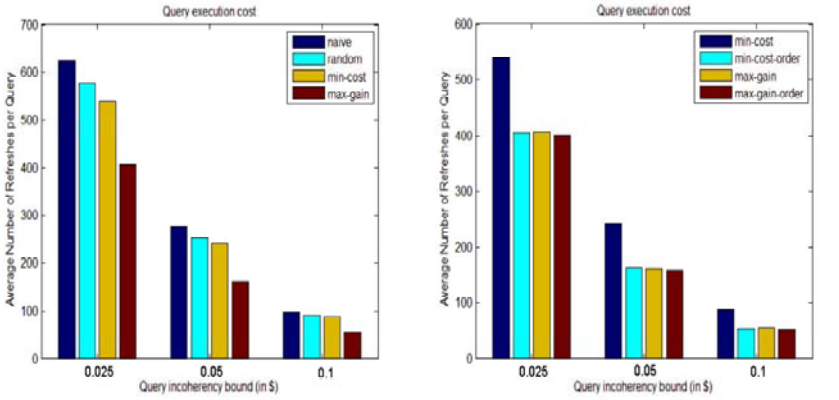
where $p(x_i > x_{ij})$ is the probability that value of ith data item is more than value of jth data item. Now we consider the optimized execution of MAX queries using the above mentioned query cost model.

To execute the MAX query using a network of data aggregators in a shared region contains group of group profiles, we assign subqueries to different DAs. Each subquery is a MAX query over a subset of query data items. For optimal planning we need to minimize the sum of subquery execution costs. As we assign same incoherency bound to all the subqueries (equals to the query incoherency bound as per (20)), we just need to minimize sum of subquery sumdiff values. Optimal query planning problem for MAX queries is NPhard. This can be proved by mapping the set cover problem to this optimal query planning problem[1].

V. Simulation Results

Fig. 8a shows simulation results for MAX query for various algorithms outlined in Section A. We have not used optc algorithm here as all data items have to be served at the query incoherency bound without any optimization in the incoherency

bound allocation. Naı¨ve algorithm requires more than 1.5 times messages compared to other efficient subquery-based algorithms. Other results are qualitatively similar to what we obtained for the additive queries with one difference. we modified the greedy algorithms by considering the data items in the descending order of sumdiffs. For example, in the max-gain algorithm we first calculate gains of subqueries covering the data item having maximum sumdiff. We select the one with maximum gain. We repeat the step for the next most dynamic data item and so on. Fig. 8b shows with this modified greedy approach, performance of min-cost and max-gain algorithms is almost the same.



(a) Comparison of algorithm (b) Effect of *data dynamics order* on performance

Fig.8.Performance of MAX queries.

VI. Related Works

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

1. Answering Incoherency Bounded Aggregation Queries over group of profiles
2. Construction and Maintenance of Network of Data Aggregators.

Various mechanisms for efficiently answering incoherency bounded aggregation queries over continuously changing data items are proposed in the literature [10], [12], [15], [17], [21]. Our work distinguishes itself by employing subquery based query evaluation to minimize number of refreshes. 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 predict data values and pull instances. In comparison, we use push-based mechanism to refresh subquery values at the client. In [12], authors propose push based scheme using data filters at the sources. According to that work, for 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 data items as part of larger subqueries while optimally assigning incoherency bounds. While this might lead to different messaging overheads for different DAs as opposed to what is proposed in [12], it does result in minimizing the total number of messages sent by DAs. In our work, each data aggregator can only disseminate data at some prespecified incoherency bound depending on its capability, whereas such a constraint does not exist for [31]. Further, we also give a method to select partial aggregates (subqueries) to be used for answering the query. In [30], authors propose cost-based methods to create innetwork- aggregation tree consisting of the query node, where an aggregation query is invoked being the root of the aggregation tree, and sensors. Each node can select subqueries based on their sumdiff values using principles outlined in this paper to minimize the number of message transfers in the network in share environment. In these works, fidelity is defined as fraction of time when the client coherency requirements are met. Each data aggregator 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 data aggregators to efficiently answer client queries. Construction and Maintenance of Network of Data Aggregators. Authors of [6], [8], [23] deal with the first part, whereas

we have studied the second part. Changes in data dynamics may lead to reorganization of the network of data aggregators which, in turn, may necessitate changes in query plans. Authors of [8] assume that each client's data requirements are fulfilled by a single data aggregator. But, in that case, 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 aggregator (using single subquery) 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 the client query. Thus, our work can complement the work of Zhou et al. [8] for end-to-end (sources-to-client) fidelity optimization.

VII. Discussions And 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 a network of data aggregators, where each DA is capable of disseminating a set of data items at their prespecified incoherency bounds in an non share region. We developed an important measure for data dynamics in the form of sumdiff is a more appropriate measure compared to the widely used standard deviation based measures.. Performance results show that by our method the query can be executed using less than one third the messages required for existing schemes. We showed that the following features of the query planning algorithms improve performance:

Developing efficient strategies for multiple invocations of our algorithm, considering hierarchy of data aggregators, is an area for future research. Another area for future research is changing a query plan as data dynamics changes. We are calculating data sumdiff in dynamic manner. If data sumdiff changes beyond a certain limit, the chosen query plan may not remain efficient.

References

1. Rajeev Gupta and Krithi Ramamritham, "Query Planning for Continuous Aggregation Queries over a Network of Data Aggregators," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 6, JUNE 2012
2. A. Davis, J. Parikh, and W. Weihl, "Edge Computing: Extending Enterprise Applications to the Edge of the Internet," Proc. 13th Int'l World Wide Web Conf. Alternate Track Papers & Posters (WWW), 2004.
3. D. VanderMeer, A. Datta, K. Dutta, H. Thomas, and K. Ramamritham, "Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web," ACM Trans. Database Systems, vol. 29, pp. 403-443, June 2004.
4. J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery," IEEE Internet Computing, vol. 6, no. 5, pp. 50-58, Sept. 2002.
5. S. Rangarajan, S. Mukerjee, and P. Rodriguez, "User Specific Request Redirection in a Content Delivery Network," Proc. Eighth Int'l Workshop Web Content Caching and Distribution (IWCW), 2003.
6. S. Shah, K. Ramamritham, and P. Shenoy, "Maintaining Coherency of Dynamic Data in Cooperating Repositories," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB), 2002.
7. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press and McGraw-Hill 2001.
8. 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.
9. "Query Cost Model Validation for Sensor Data,"www.cse.iitb.ac.in/~grajeev/sumdiff/RaviVijay_BTP06.pdf, 2011.
10. 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.
11. A. Populis, Probability, Random Variable and Stochastic Process. Mc. Graw-Hill, 1991.
12. C. Olston, J. Jiang, and J. Widom, "Adaptive Filter for Continuous Queries over Distributed Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2003.
13. S. Shah, K. Ramamritham, and C. Ravishankar, "Client Assignment in Content Dissemination Networks for Dynamic Data," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005.
14. NEFSCScientificComputerSystem,http://sole.wh.whoi.edu/~jmanning//cruise/serve1.cgi, 2011.
15. S. Madden, M.J. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," Proc. Fifth Symp. Operating Systems Design and Implementation, 2002
16. D.S. Johnson and M.R. Garey, Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.
17. S. Zhu and C. Ravishankar, "Stochastic Consistency and Scalable Pull-Based Caching for Erratic Data Sources," Proc. 30th Int'l Conf. Very Large Data Bases (VLDB) 2004.
18. D. Chu, A. Deshpande, J. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," Proc. 22nd Int'l Conf. Data Eng. (ICDE), 2006.

19. A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, and W.Hong, "Model-Driven Data Acquisition in Sensor Networks,"Proc. 30th Int'l Conf. Very Large Dat Bases (VLDB), 2004.
20. Pearson Product Moment Correlation Coefficient, http://www.nyx.net/~tmacfarl/STAT_TUT/correlat.ssi/, 2011.
21. A.Deligiannakis,Y.Kotidis,and N. Roussopoulos,"ProcessingApproximate Aggregate Queries in Wireless Sensor Networks," Information Systems, vol. 31, no. 8, pp. 770-792, 2006.
22. G. Cormode and M. Garofalakis, "Sketching Streams through the Net: Distributed Approximate Query Tracking," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005.
23. S. Agrawal, K. Ramamritham, and S. Shah, "Construction of a Temporal Coherency Preserving Dynamic Data Dissemination Network," Proc. IEEE 25th Int'l Real-Time Systems Symp. (RTSS), 2004.
24. B. Babcock and C. Olston, "Distributed Top-K Monitoring," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2003.
25. A. Silberstein, K. Munagala, and J. Yang, "Energy Efficient Monitoring of Extreme Values in Sensor Networks," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2006.
26. 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.
27. 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.
28. S. Kashyap, J. Ramamritham, R. Rastogi, and P. Shukla, "Efficient Constraint Monitoring Using Adaptive Thresholds," Proc. IEEE 24th Int'l Conf. Data Eng., 2008.
29. D.S. Hochbaum, "Approximation Algorithms for the Set Covering and Vertex Cover Problems," SIAM J. Computing, vol. 11, no. 3, pp. 555-556, 198
30. P. Edara, A. Limaye, and K. Ramamritham, "Asynchronous In- Network Prediction: Efficient Aggregation in Sensor Networks," ACM Trans. Sensor Networks, vol. 4, no. 4, pp. 1-34, Aug. 2008.

AUTHOR(S) PROFILE

ABOUT AUTHORS

Varada Venkata SubbaRao , pursuing M.tech in the Department of CSE ,Sri Aditya Engineering college,Surempalem.He completed his M.C.A from St. Mary's College For Pg Courses,Surempalem in 2005.

N.Venkata Ramana M.Tech is an Assistant Professor in the department of CSE,Sri Aditya Engineering college,surempalem,currently working towards Data mining Reasearch.

