Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 9; April-June, 2015 pp. 66-70

http://www.krishisanskriti.org/acsit.html

Query Planning for Dynamic Queries using Data Aggregators

Nadia Nisar

Department of Computer Science and Engineering Al-Falah University, Faridabad, Haryana-India
E-mail: nadia.nisar786@gmail.com

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 value of portfolio for a client; or the AVG of temperatures sensed by a set of sensors. In these queries a client specifies a coherency requirement as part of the query. 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. Just as various fragments of a dynamic webpage are served by one or more nodes of a content distribution network, our technique involves decomposing a client query into subqueries and executing subqueries on judiciously chosen data aggregators with their individual subquery incoherency bounds. We provide a 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. 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.*

1. INTRODUCTION

Applications such as auctions, personal portfolio valuations 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. As an example, consider a user who wants to track a portfolio of stocks in different (brokerage) accounts. Stock data values from possibly different sources are required to be aggregated to satisfy user's requirement.

These aggregation queries are long running queries as data are continuously changing and the user is interested in notifications when certain conditions hold. Thus, responses to these queries are refreshed continuously. In these continuous query applications, users are likely to tolerate some inaccuracy in the results. That is, the exact data values at the corresponding data sources need not be reported as long as the query results satisfy user specified accuracy requirements. For instance, a portfolio tracker may be happy with an accuracy of $10.

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 (DA): 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 clients. For scalable handling of push based data dissemination, network of data aggregators are proposed in the literature [5], [7], [22]. 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 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 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 d_1 - d_4, various aggregators can be characterized as

$$a_1 : \{(d_1, 0.5), (d_3, 0.2)\}$$
$$a_2 : \{(d_1, 1.0), (d_2, 0.1), (d_4, 0.2)\}$$

Aggregator a_1 can serve values of d_1 with an incoherency bound greater than or equal to 0.5 whereas a_2 can disseminate the same data item at a looser incoherency bound of or more. In such a network of aggregators of multiple data items all the nodes can be considered as peers since a node ai can help another node ak to maintain incoherency bound of the data item d1 (incoherency bound of d1 at ai is tighter than that at ak) but the node ai gets values of another data item d2 from ak.

2. AGGREGATE QUERIES AND THEIR EXECUTION

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. 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.

Consider a client query Q=50d_1) 200d_2)150d_3, where d_1, d_2, d_3 are different stocks in a portfolio, with a required incoherency bound of $80. We want to execute this query over the data aggregators given in Example 1, minimizing the number of refreshes.

In the above case a limited number of options are available for executing the aggregation query. Specifically we answer the question: Given a client query posed over a hypothetical database of multiple data sources, what subqueries should be posed at various data aggregators so that the number of refreshes from these aggregators to the client can be minimized? We use additive aggregation queries to develop our approach in detail and toward the end of the paper describe how max/min queries can be handled.

For answering the multidata aggregation query, there are three options for the client to get the query results. First, the client may get the data items d_1, d_2, and d_3 separately. The query incoherency bound can be divided among data items in various ways ensuring that query incoherency is below the incoherency bound. In this paper, we show that getting data items independently is a costly option. This strategy ignores the fact that the client is interested only in the aggregated value of the data items and various aggregators can disseminate more than one data item.

Second, 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 (50d_1) 200d_2)150d_3) 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 data item updates may cancel out each other, thereby maintaining the query results 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 Example 1. Further, even if an aggregator can refresh 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.

A third option is to divide the query into a number of subqueries and get their values from individual DAs. In that case, the client query result is obtained by combining the results of multiple subqueries. For the DAs given in Example1, the query Q can be divided in two alternative ways:

Plan 1. Result of subquery 50d_1) 150d_3 is served by a_1, whereas value of d_2 is served

by a_2.

Plan 2. Value of d_3 is served by a_1, whereas result of subquery 50d_1) 200d_2 is served by a_2.

In both the plans, combining the subquery values at the client gives the query result. But, selecting the optimal plan among various options is not trivial. Intuitively, 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 subqueries such that updates to various data items appearing in a subquery have more chances of canceling each other as that will reduce the need for refresh to the client. In the above example, if updates to d_1 and d_3 are such that when d_1 increases, d_3 decreases, and vice versa, then selecting plan1 may be beneficial. We give a method to select the query plan based on these observations. While solving the above problem, 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., $50d_1$) $200d_2$) $150d_3$ is divided into two subqueries $50d_1$) $130d_2$ and $70d_2$) $150d_3$), in doing so the same data item is processed at multiple aggregators, increasing the unnecessary processing load (further, in case of paid data subscriptions it is not prudent to get the same data item from multiple sources). By dividing the client query into disjoint subqueries we ensure that a data item update is processed only once for each query.

Subquery incoherency bounds are required to be derived using the query incoherency bounds such that, besides satisfying the client coherency requirements, the chosen DA (where the subquery is to be executed) is capable of satisfying the allocated subquery incoherency bound. For example, in plan1, incoherency bound allocated to the subquery $50d_1$) $150d_3$ should be greater than 55(=50*0.5)150*0.2)as that is the tightest incoherency bound which the aggregator a_1 can satisfy for the subquery $50d_1$) $150d_3$. We show that the number of refreshes also depends on the division of the query incoherency bounds among subquery incoherency bounds. A similar result was reported for data incoherency bounds in [11].

3. DATA DISSEMINATION COST MODEL

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. 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.

2.1 Incoherency Bound Model

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 jv(t) ≤ u(t)j 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:

Fig. 1: Number of pushes versus incoherency bounds.

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

1. Client estimates data value based on server specified parameters [12], [16]. The source pushes the new data value whenever it differs from the (client) estimated value by an amount more than C.

In both these cases, value at the source can be modeled as a random process with average as the value known at the client. In case 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]

$$P\ (|v(t) \leq u(t)| > C)\ \alpha\ 1=C^2 \qquad (4)$$

Thus, we hypothesize that the number of data refresh messages is inversely proportional to the square of the incoherency bound. A similar result was reported in [5] where data dynamics were modeled as random walks.

2.2 Data Dynamics Model

We considered two possible options to model data dynamics. As a first option, the data dynamics can be quantified based on standard deviation of the data item values.

Fig. 2: Number of pushes versus data sumdiff (a) C = 0:001, (b) C = 0:01, and (c) C = 0:1.

	whereas higher order coefficients represent	
We take an example to show why standard	transient changes in the value of data item.	
deviation is not a good measure of data	We hypothesize that the cost of data	
dynamics in our case: Suppose data values	dissemination for a data item can be	
in consecutive instances for a data item d1	approximated by a function of the first FFT	
are {0, 4, 0, 4, 0, 4, 0, 4} whereas for	coefficient. Specifically, the cost of data	
another data item d2 values are {0, 0, 0, 0, 4,	dissemination for a data item will be	
4, 4, 4}. Suppose both data items are	proportional to data sumdiff defined as:	
disseminated with an incoherency bound of		
3. It can be seen that the number of	$Rs = \sum \|si - si\text{-}1\|,$	(5)
messages required for	i	
maintaining the		
incoherency bound will be 7 and 1 for data		
items d1 and d2, respectively, whereas both	where si and si-1 are the sampled values of a	

data items have the same standard deviation	data item S at ith and (i - 1)th time instances	
(=2:138). Thus, we need a measure which	(i.e., consecutive ticks). In practice, sumdiff	
captures data changes along with its	value for a data item can be calculated at the	
temporal properties. This motivates us to	data source by taking running average of	
examine the second measure.	difference between data values for	
	consecutive ticks. For our experiments, we	
As a second option we considered Fast	calculated the sumdiff values using	
Fourier Trans-form (FFT) which is used in	exponential window moving average with	
the digital signal processing domain to	each window having 100 samples and giving	
characterize a digital signal. FFT captures	30 percent weight to the most recent	
number of changes in data value, amount of	window.	
changes, and their timings. Thus, FFT can		
be used to model data dynamics but it has a	2.3 Combining Data	Dissemination
problem. To estimate the number of	Models	
refreshes required to disseminate a data item		
we need a function over FFT coefficients	Number of refresh messages is proportional	
which can return a scalar value. The number	to data sumdiff Rs	and inversely
of FFT coefficients can be as high as the	proportional to square of the incoherency	
number of changes in the data value. Among	bound (C2). Further, we can see that we need	
FFT coefficients, 0th order coefficient	not disseminate any message when either	
identifies average value of the data item,	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 $R_S=C^2$. In the next section, we use this data dissemination cost model for developing cost model for additive aggregation queries.

4. QUERY PLANNING FOR WEIGHTED ADDITIVE AGGREGATION QUERIES

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

Inputs

1. A network of data aggregators in the form of a relation f (A, D, C) specifying the N data aggregators $a_k \in A(1 \leq k \leq N)$, set $D_k \subseteq D$ of data items disseminated by the data aggregator a_k, and incoherency bound t_{kj} which the aggregator a_k can ensure for each data item $d_{kj} \in D_k$.

2. Client query q and its incoherency bound C_q. An additive aggregation query q can be represented as $\sum w_{qi} d_{qi}$ where w_{qi} is the weight of the data item d_{qi} for $1 \leq i \leq n_q$.

Outputs

1. q_k for $1 \leq k \leq N$, i.e., subquery for each data aggregator a_k.
2. 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 q_ks for each data aggregator.
2. *Dividing incoherency bound*: Divide the query incoherency bound Cq among subqueries to get Cqk s

5. 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. We developed an important measure for data dynamics in the form of sumdiff which, as we discussed in Section 2, is a more appropriate measure compared to the widely used standard deviation based measures. For optimal query execution we divide the query into subqueries and evaluate each subquery at a judiciously chosen data aggregator.

REFERENCES

[1] A. Davis, J. Parikh, and W. Weihl, "Edge Computing: Extending Enterprise

[2] Applications to the Edge of the Internet," Proc. 13th Int'l World Wide Web Conf.

[3] Alternate Track Papers & Posters (WWW), 2004.

[4] D. VanderMeer, A. Datta, K. Dutta, H. Thomas, and K. Ramamritham, "Proxy-Based Acceleration of Dynamically Generated Content on the World Wide

[5] Web," ACM Trans. Database Systems, vol. 29, pp. 403-443, June 2004.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl,

[7] "Globally Distributed Content Delivery,"

[8] IEEE Internet Computing, vol. 6, no. 5, pp. 50-58, Sept. 2002.

[9] S. Rangarajan, S. Mukerjee, and P.

[10] Rodriguez, "User Specific Request Redirection in a Content Delivery Network," Proc. Eighth Int'l Workshop Web Content

[11] Caching and Distribution (IWCW), 2003.

[12] 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.

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press and McGraw-Hill 2001.

[14] 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.

[15] "Query Cost Model Validation for Sensor Data, "www.cse.iitb.ac.in/ ~grajeev/sumdiff/RaviVijay_BTP06.pdf, 2011.

[16] R. Gupta, A. Puri, and K. Ramamritham,

[17] "Executing Incoherency Bounded Continuous Queries at Web Data Aggregators," Proc. 14th Int'l Conf. World Wide Web (WWW), 2005.

[18] A. Populis, Probability, Random Variable and Stochastic Process. Mc. Graw-Hill, 1991.

[19] 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.

[20] S. Shah, K. Ramamritham, and C.

[21] Ravishankar, "Client Assignment in Content Dissemination Networks for Dynamic Data," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), 2005.

[22] NEFSC Scientific Computer System, http://sole.wh.whoi.edu/~jmanning//cruise/s erve1.cgi, 2011.

[23] S. Madden, M.J. Franklin, J.

[24] Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," Proc. Fifth Symp. Operating Systems Design and Implementation, 2002.

[25] D.S. Johnson and M.R. Garey, Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

[26] 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.

[27] 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.

[28] 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

[29] Large Data Bases (VLDB), 2004.

[30] Pearson Product Moment Correlation Coefficient,http://www.nyx.net/~tmacfarl/S TAT_TUT/correlat.ssi/, 2011.

[31] S. Agrawal, K. Ramamritham, and S.

[32] Shah, "Construction of a Temporal Coherency Preserving Dynamic Data Dissemination Network," Proc. IEEE 25th Int'l Real-Time Systems Symp. (RTSS), 2004.